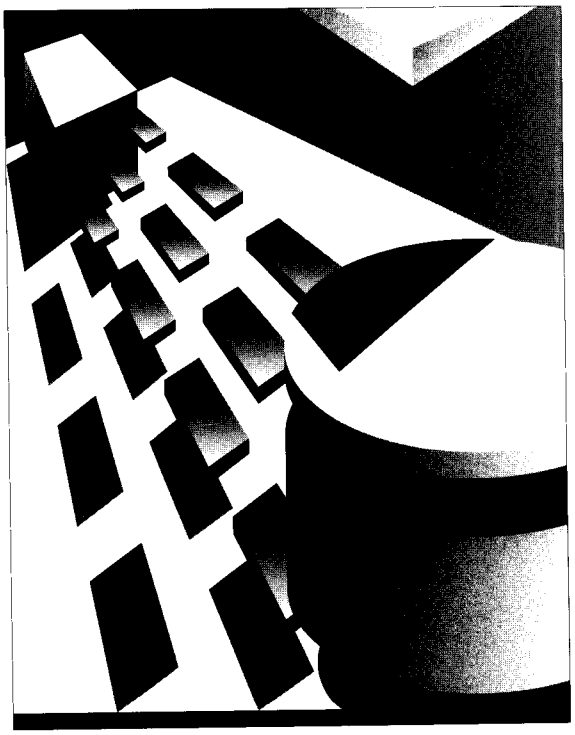
T A N D E M

SYSTEMS REVIEW

VOLUME 5, NUMBER 2

SEPTEMBER 1989



An Overview of SNAX/CDF

Network Design Considerations

NonStop SQL:
The Single Database Solution

Optimizing Batch Performance

Volume 5, Number 2, September 1989

Editorial Director
Susan Wayne Thompson

Editor
Anne Lewis

Associate Editor
Suzanne Ambiel

Editorial Advisor
Ellen Marielle-Tréhöuart

Technical Advisors
Mark Anderton
Bart Grantham

Assistant Editor
Sarah Rood

Electronic Publishing
Annie F. Valva

Art Director
Janet Stevenson

Cover Art
Cara Koenig

Production and Layout
Melanie Bell
Jaroslav Dostal

The *Tandem Systems Review* is published by Tandem Computers Incorporated.

Purpose: The *Tandem Systems Review* publishes technical information about Tandem software releases and products. Its purpose is to help programmer-analysts who use our computer systems to plan for, install, use, and tune Tandem products.

Subscription additions and changes: Subscriptions are free. To add names or make corrections to the distribution database, requests within the U.S. should be sent to Tandem Computers Incorporated, *Tandem Systems Review*, 18922 Forge Drive, LOC 216-05, Cupertino, CA 95014. *Requests outside the U.S. should be sent to the local Tandem sales office.*

Comments: The editors welcome suggestions for content and format. Please send them to the *Tandem Systems Review*, 18922 Forge Drive, LOC 216-05, Cupertino, CA 95014.

Preface

Connecting networks from different vendors is becoming increasingly important as users require access to more and larger databases. Users need the ability to quickly access other networks without taking the time to learn a new set of interfaces. The opening article by Turner describes SNAX/CDF (SNAX Cross-Domain Facility), which allows users on Systems Network Architecture (SNA)–based and Tandem networks to access each other's applications and databases. SNAX/CDF not only provides connection between Tandem and SNA networks but allows users to access both networks with the familiar operating system commands of their home system. This overview article describes the functional aspects of the product, its components, system configuration considerations, and problem management.

A properly designed data network provides companies with an effective means for reducing costs and also ensures that it has the facilities and the flexibility to meet current and future demands. Good network design must be a careful and thorough analysis of the network requirements as well as the implications of the available design alternatives. The article by Evjen presents a high-level overview of the considerations involved in the design of data networks. It discusses the issues and decisions that a network designer makes when designing or modifying a network. The article also discusses some techniques and methodologies that can be used in the network design process.

The next article in this issue describes the performance evaluation benchmark that was made for the California Department of Motor Vehicles (DMV). The purpose of this benchmark was to demonstrate to DMV that Tandem systems can simultaneously process both online and batch processing on a database of nearly 60 Gbytes while maintaining a transaction rate of 30 transactions per second and a response time of 1.5 seconds. Cassidy and Kocher explain how the benchmark system, using NonStop VLX systems and NonStop SQL, met the DMV performance, fault tolerance, linear expandability, and ease-of-use requirements. The article describes the benchmark system's configuration, tuning, and testing methods.

The final article, by Keefauver, continues a series of articles that the *Tandem Systems Review* is publishing on batch processing. Because of the inherent value of sequential processing, integrating batch processing with online transaction processing (OLTP) is important to efficient system management. Keefauver describes how recent enhancements to the GUARDIAN 90 operating system allow batch applications to take advantage of the Tandem system's parallel architecture and optimally perform with OLTP applications. The article also describes how each successive version of the software, starting with version C10, improved sequential processing techniques for disk access, memory, usage, and file management.

The last page of this issue is a reader response card. The purpose of this questionnaire is to give the *Tandem Systems Review* staff information about reader interests. Please take a few minutes to evaluate each article in this issue and indicate the area in which you would like to see more articles.

Susan Wayne Thompson
Editorial Director

An Overview of SNAX/CDF

The SNAX™ Cross-Domain Facility (SNAX/CDF) product represents the next step in the evolution of Tandem's SNA Communications Services (SNAX) family of products, which provide access between Tandem® and IBM Systems Network Architecture (SNA) networks. SNAX/CDF enables users on a Tandem system to access applications on an SNA network. Users on an IBM terminal connected to an SNA network can use the SNAX/CDF product as a way to access Tandem applications.

This overview provides a discussion focusing on the technical aspects of enhanced connectivity with SNAX/CDF, system configuration, SNAX/CDF components and processes, and problem management. Although the article is intended primarily for those readers familiar with SNA, readers who need definitions of the italicized terms can refer to the SNA glossary at the end of this article.

The Development of SNAX/CDF

As a leading supplier of online transaction processing (OLTP) solutions, Tandem recognizes interoperability with other vendor systems to be a critical component. The Tandem SNA products provide system interoperability in an IBM SNA-dominated industry.

The initial Tandem SNA product, SNAX/XF (SNA Communications Services–Extended Facility), acts as a multipurpose gateway between Tandem networks and SNA network hosts and devices. SNAX/XF capabilities allow all of the following:

- Users can build OLTP applications on Tandem systems, which can then communicate with a wide variety of SNA devices attached to a single Tandem system or a network of systems.
- Tandem applications can readily communicate with applications on IBM (or compatible) systems.
- SNA devices can continue to access existing IBM applications by way of the Tandem system.

Having developed applications on Tandem systems that are strategic to their business, users require that these OLTP applications be available to users throughout their organization. However, the majority of the network resources in the organization is often owned and managed by IBM system software, namely *Advanced Communication Function/Virtual Telecommunications Access Method* (ACF/VTAM, or simply VTAM) and *Advanced Communication Function/Network Control Program* (ACF/NCP, or simply NCP).

If this is the case, as it is for many large IBM users, SNAX/XF is restricted in its ability to make OLTP applications available to all network users. SNAX/XF is an implementation of that part of SNA to present a *physical unit (PU) Type 2.0* appearance to VTAM and NCP. Because SNA does not allow peer-to-peer connections between PU Type 2.0 nodes, terminal users (which are also represented as PU Type 2.0s) connected to the IBM system cannot communicate with the Tandem system through VTAM and NCP.

The SNAX/CDF product, developed to correct this problem, now provides complete SNA connectivity between Tandem and IBM systems. SNAX/CDF allows *cross-domain sessions* to be established between IBM-owned devices and the strategic applications executing on Tandem systems. Users can now utilize key products such as NonStop® SQL, build their critical databases and applications on the Tandem system, and have those databases and applications available to all users connected to the corporate network. SNAX/CDF widens the options available to users when they plan their distributed OLTP applications and design the networks that best suit their business needs.

Technical Overview of SNAX/CDF

SNAX/CDF is designed to provide access to SNA *backbone networks* by appearing to those networks as a peer functional component to the VTAM and NCP products. This means that SNAX/CDF appears to VTAM (an SNA host component) as a peer host; to NCP (an SNA communications controller component), SNAX/CDF appears as a peer communications controller. Architecturally, SNAX/CDF provides the same functions as both a *PU Type 4* (such as NCP) and a *PU Type 5* (such as VTAM) node.

When a Tandem system uses SNAX/CDF to connect into an SNA network, it does so using one or more Synchronous Data Link Control (SDLC) communication lines. The Tandem system appears to the SNA network as an SNA *subarea node*. (See Figure 1.) In comparison, a commonly used IBM system configuration connecting to an SNA network appears as two subarea nodes, one for VTAM and another for NCP.

Figure 1

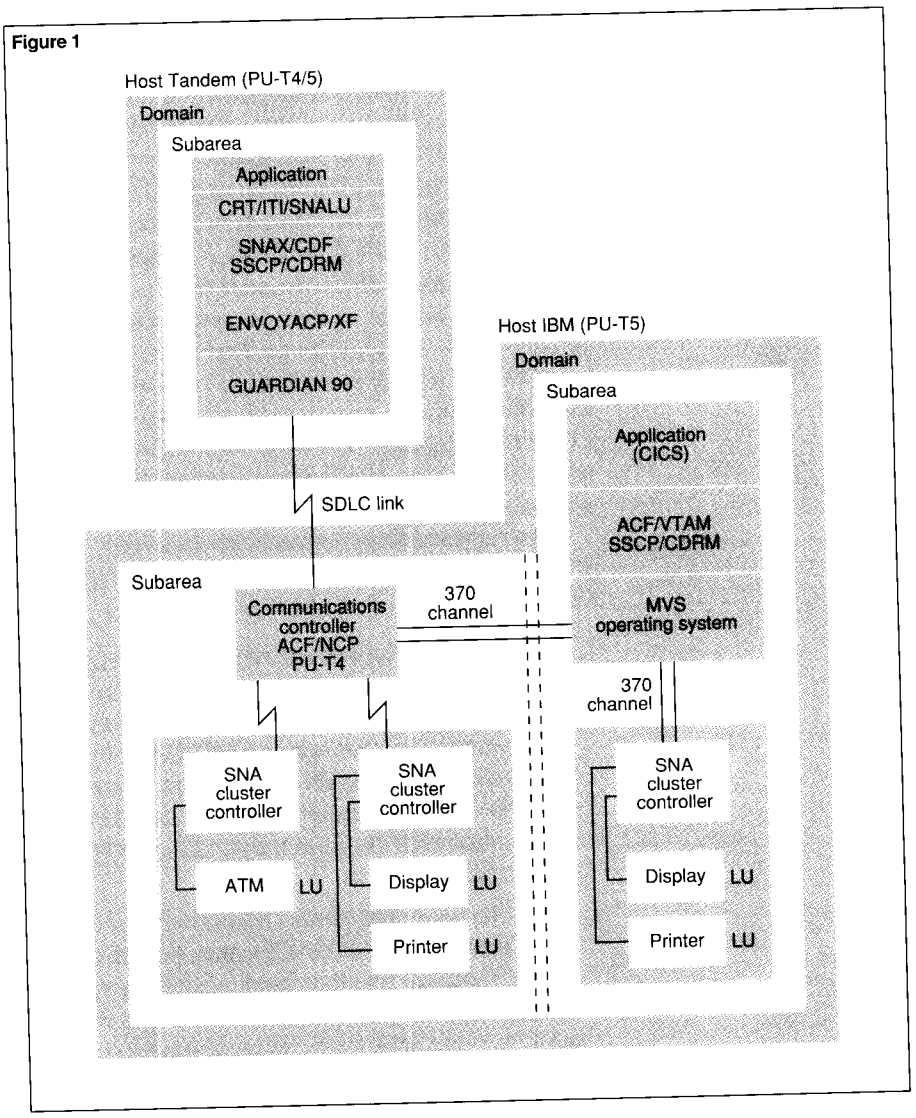


Figure 1.
Example SNAX/CDF subarea network.

Cross-Domain Session Establishment

Domains are basic organizational units within an SNA network. A domain contains one or more *subareas* and their subordinate *peripheral nodes*, which represent resources capable of acting as *logical units* (LUs). LUs can be application programs or terminals. A *system services control point* (SSCP) controls each domain and is responsible for the error recovery of its owned resources.

Once the Tandem system is connected to the SNA network, SNAX/CDF facilitates the establishment of cross-domain SSCP-SSCP (also known as CDRM-CDRM[1]) sessions. This in turn enables LUs on PU Type 2 or PU Type 5 nodes in the SNA network to access applications (represented as LUs within SNAX/CDF) on the Tandem system.

Two required steps precede the access of SNAX/CDF to the partner cross-domain SSCP and the establishment of an SSCP-SSCP session:

1. Any *transmission groups* and their associated communication links between SNAX/CDF and adjacent NCPs must be activated. This activation is accomplished through the ENVOY™ bit-oriented protocols with extended functions (ENVOYACP/XF) product, which gains knowledge of the NCP through the *exchange ID (XID) SDLC* frame.
2. *Explicit routes* and *virtual routes* defined to exist between the two SSCPs must be activated. These routes define the physical and logical path through the SNA backbone network that will be used to route SNA traffic to its destination.

With these two steps completed, SSCP-SSCP session establishment can begin. SNAX/CDF acts as a peer SSCP to set up one or more required cross-domain SSCP-SSCP sessions (one per SSCP) between SNAX/CDF and the remote SSCP domains in which the target LUs reside. Once that exists, the partnership of the two SSCPs then allows LU-LU session initiation and termination requests between Tandem application LUs and cross-domain application or terminal LUs in the SNA network.

LU-LU Session Initiation. An LU-LU session can be initiated from SNAX/CDF or VTAM using the facilities of the appropriate *application program interface* (API). The API provides a function translation layer between the application program and SNAX/CDF. Prior to the LU-LU session initiation, the explicit and virtual routes involved in the connection are activated by SNAX/CDF and VTAM along the path to be used by the LU-LU session.[2]

The SSCPs in each participating domain exchange information in a handshake sequence to initiate the LU-LU session. The information exchanged between the two SSCPs defines how the LU-LU session is set up and determines the SSCP resources needed to support the LU-LU session. Once the basic session knowledge is created by the SSCPs, they are no longer involved until session termination. The next step, an exchange of a BIND request by the two API layers, actually sets up the LU-LU session between the two applications.

LU-LU Session Termination. Termination of an LU-LU session can be invoked by the SSCP function of SNAX/CDF or VTAM, by the API layer function on behalf of the application representing an LU, or by network errors causing the explicit or virtual route to be broken. The SSCPs that control the domains in which the participating LUs reside direct the termination. Once again, a handshake sequence is exchanged between the two SSCPs. This causes all representation of the LU-LU session to be deleted, and resources used during the session are made available for other use.

Figure 2 illustrates the complexity of establishing the many routes and sessions discussed above in order to ultimately exchange data between LUs of different systems.

Additional Technical Considerations

The SNAX/CDF product appears as a standard Tandem I/O process to Tandem applications using SNAX/CDF with the CRT, ITI, or SNALU application interfaces. (These interfaces are discussed later in this overview.) Compatibility is maintained at the API level with the current Tandem SNA product set.

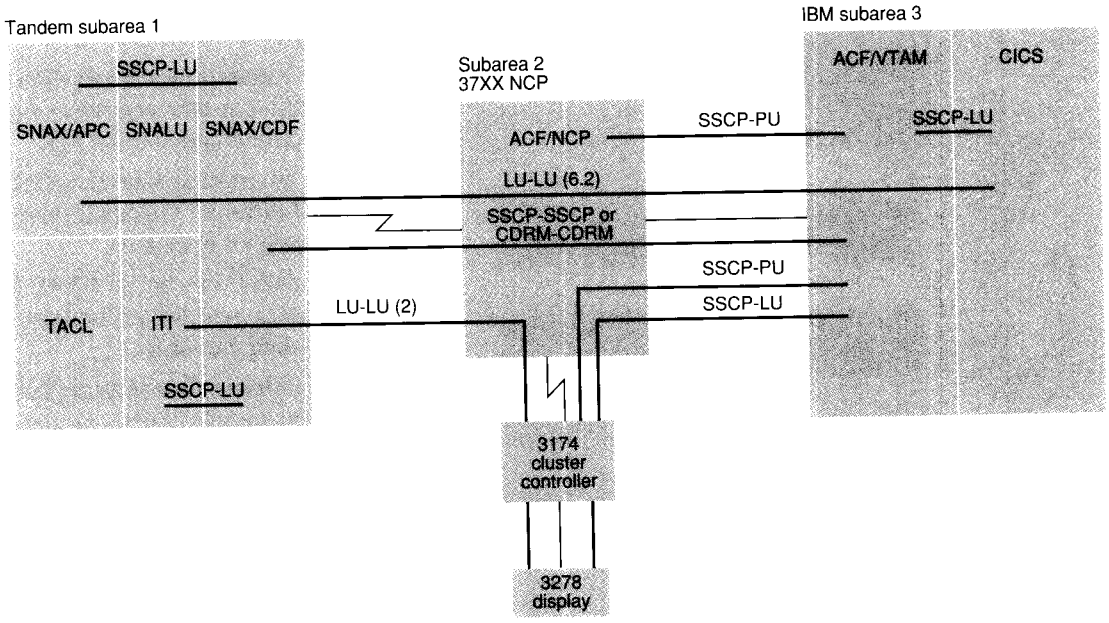
However, on the Tandem system, SNAX/CDF executes as a nonprivileged process under the control of the GUARDIAN® 90 operating system; it requires no specific SYSGEN considerations.[3] SNAX/CDF is configured online using the Subsystem Control Facility (SCF) product.

[1] Although a CDRM (cross-domain resource manager) is a component of an SSCP, the term CDRM-CDRM session is often used as a synonym for an SSCP-SSCP session.

[2] The choice of which path to activate is made based on routing information defined in the Class of Service Table (COSTAB).

[3] The GUARDIAN 90 release level must be C20 or later.

Figure 2



SNAX/CDF provides additional functions with two other processes, Virtual Route Selection Process (VRSP) and CREATOR. VRSP is a user-provided process and has the capability of altering virtual route selection for sessions. The CREATOR process, supplied by Tandem, can be configured to participate in all cross-domain logon requests from remote LUs logging onto CRT or ITI Tandem application LUs.

SNAX/CDF uses the ENVOYACP/XF product to provide SDLC protocol support and the physical connection to the SNA network. ENVOYACP/XF requires a SYSGEN. Figure 3 illustrates these different components and how they relate to SNAX/CDF.

Internal Structure

SNAX/CDF was written in strict adherence to the *IBM Systems Network Architecture Format and Protocol Reference Manual*, which defines the protocols and message formats for SNA peer subarea network components. Exceptions to adherence were allowed where the information in the IBM reference manual was incomplete, technical deficiencies needed to be corrected, or a specific Tandem implementation consideration was required.

Figure 3

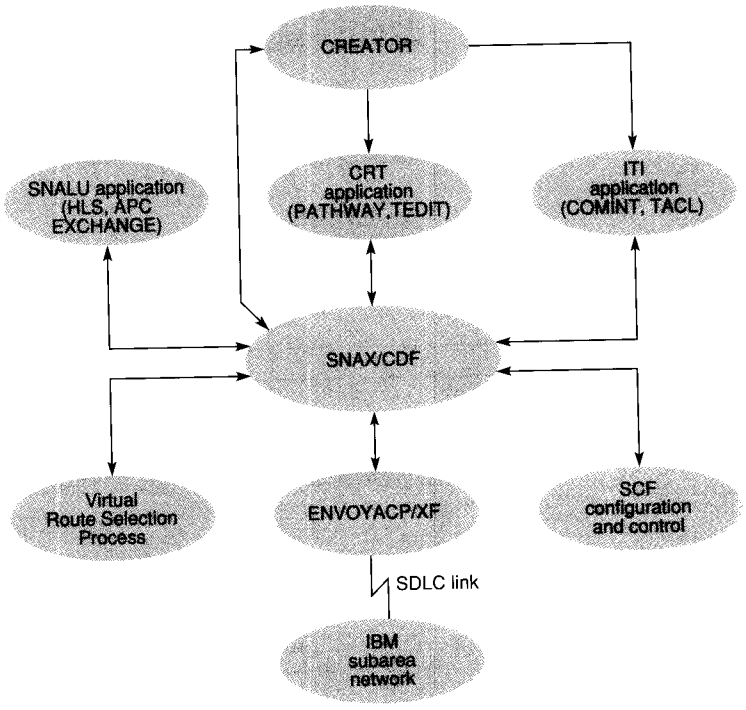
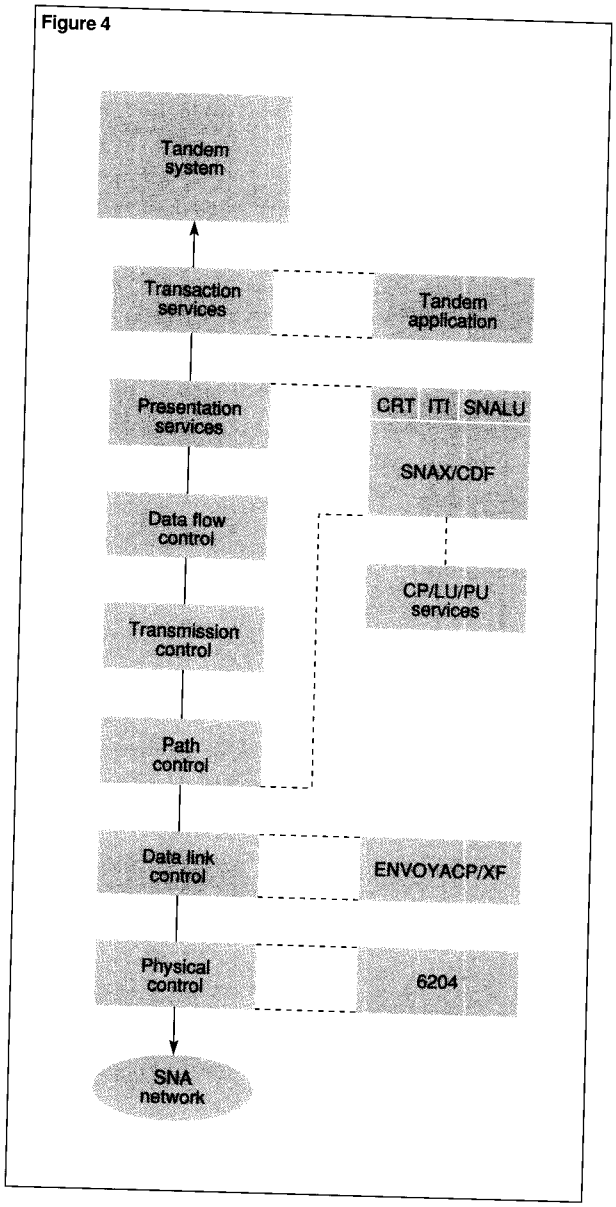


Figure 2. *Example SNAX/CDF sessions.*

Figure 3. *SNAX/CDF overview.*

Figure 4.
SNA layers and SNAX/CDF.



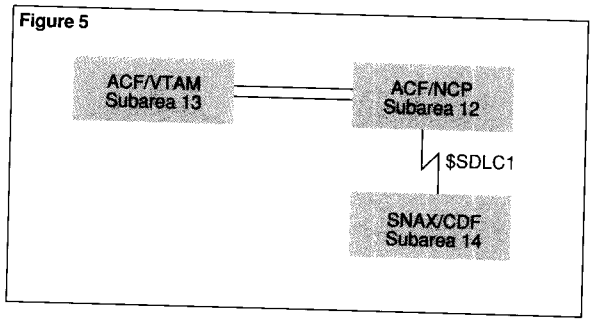
The SNAX/CDF process is, therefore, layered in accordance to the SNA model. The SNAX/CDF layers are:

- The transaction services (TS) layer provides application interface services between the application program and SNAX/CDF.
- The presentation services (PS) layer formats the data for different presentation media and handles the sharing of resources.
- The data flow control (DFC) layer[4] synchronizes data flow, correlates exchange of data, and groups related data into units.
- The transmission control (TC) layer[4] paces data exchange to help control processing capacity.
- The path control (PC) layer routes data between source and destination and controls the data traffic in the network.
- The data link control (DLC) layer is responsible for the transfer of data between adjacent nodes.
- The physical layer is responsible for the hardware control of the actual hardware communication link.

Figure 4 provides a representation of the basic layered structure of SNAX/CDF and how it maps to the SNA model. For SNAX/CDF, the DLC layer is provided by a separate process, the ENVOYACP/XF product, which in turn communicates with the 6204 controller (the physical layer). The GUARDIAN 90 file system acts as the interface between SNAX/CDF and ENVOYACP/XF. The PC layer of SNAX/CDF is able to support multiple-line transmission groups. This means that SNAX/CDF can be configured to use many copies of ENVOYACP/XF, and each copy can be controlling one or more communication lines to adjacent NCPs.

Note that, for SNAX/CDF, the PC layer only supports end-point attachment to an SNA backbone network. All communication traffic received by SNAX/CDF must terminate at SNAX/CDF. However, because of the flexibility of a GUARDIAN 90 network (perhaps combined with EXPAND™), applications can be located on other Tandem systems that are connected to the node on which SNAX/CDF is executing.

Figure 5.
Sample SNAX/CDF SNA backbone network.



[4]With SNALU, DFC and TC functions are performed by the SNALU application program.

Figure 6

```
43>  scf
SCF - T9082C11 - (15OCT89) (J15) - 10/11/89 11:18:28 System \SNAX
Copyright Tandem Computers Incorporated 1986, 1987, 1988
(Invoking $CDF.MARTY.SCFCSTM)
LOG MDT5.CDFLOG !
SCF W20052 Creating file $CDF.MDT5.CDFLOG
1->  o sts5
-> RUN CDFOBJ/name $mS5,pri 150/ &
->    HOSTSA 14,                 &
->    NETNAME M14,               &
->    COLLECTOR,                 &
->    SETTABLE $cdf.MDT5.TESTseto
11OCT89 11:18:36 $MS5: Process started SNAX-CSF (T9094C20 15FEB89)
11OCT89 11:18:48 $MS5: CDRM M14 active
-> ASSUME       $MS5
-> ADD PATH     DESTSA12,  ER0 (12,1),    VR0  0
-> ADD PATH     DESTSA13,  ER0 (12,1),    VR0  0
-> ADD LINE     LINK1,     TNDM $SDLC1
-> ADD PU       PU1,       LINE LINK1,    TGN  1, SUBAREA 12 ,PUTYPE 4
-> ADD CDRM     M13,       SUBAREA 13
-> ADD CDRSC    JES2,      CDRM M13
-> ADD CDRSC    TSO,       CDRM M13
-> ADD APPL     EXCHG01,   OPENNAME   #EX01, PROTOCOL   SNALU,DLU   JES2
-> ADD APPL     MTTSO,     OPENNAME   #MTSO,PROTOCOL   SNALU,DLU   TSO,CHARACTERSET EBCDIC
-> ADD APPL     MTA,       OPENNAME   #MTA,  PROTOCOL   CREATOR,RANGE 5, &
->                         APPLFILE   $SYSTEM.SYS07.TACL,SCREEN FAST.PASSWORD "ABC"
-> START LINE LINK1
-> START PU   PU1
11OCT88 11:19:42 $MS5:  Link LINK1 Active
-> START CDRM M13
11OCT89 11:19:43 $MS5:  Activate in progress with CDRM M13 due to ACTCDRM
11OCT89 11:19:43 $MS5:  Link station PU1 has contacted subarea 12
11OCT89 11:19:48 $MS5:  CDRM M13 active
```

Figure 6.
Sample SNAX/CDF initialization (without Cool Start File).

SNAX/CDF Configuration Overview

A primary purpose of SNAX/CDF is to provide communication between applications on the Tandem system and cross-domain resources owned by the IBM system. This requires configuration of various elements in both systems. Figure 5 illustrates the configuration elements of an SNAX/CDF SNA backbone network. In addition, communication using an API, optional control of virtual route selection through VRSP, and problem management are elements of cross-domain information exchange supported by SNAX/CDF.

Tandem System Configuration

On the Tandem side, two elements require configuration. These elements are SNAX/CDF and ENVOYACP/XF.

SNAX/CDF. SNAX/CDF is a runable, nonprivileged Tandem process and as such has no direct SYSGEN considerations. The parameters required by SNAX/CDF are documented in the *SNAX/XF Configuration and Control Manual.* These parameters, which describe how SNAX/CDF interacts with the backbone SNA network, are provided either as startup parameters as part of the RUN command for the SNAX/CDF process or by way of SCF.

SCF is used to configure the SNAX/CDF product while it is running. Configuration details are found in the *SNAX/XF Configuration and Control Manual.* The configuration parameters for SNAX/CDF are analogous to those used to define VTAM and NCP. Customer system programmers, who are used to configuring these IBM products, should have little difficulty in configuring SNAX/CDF. Through SCF, resources can be defined to SNAX/CDF as well as altered while the product is executing. Figure 6 displays a sample SNAX/CDF initialization process by way of SCF.

Figure 7

```
76>  scf
SCF - T9082C11 - (15OCT89) (J15) - 10/11/89 13:40:55 System \SNAX
Copyright Tandem Computers Incorporated 1986, 1987, 1988
(Invoking $CDF.MARTY.SCFCSTM)
LOG  MDT5.CDFLOG !
SCF  W20052 Creating file $CDF.MDT5.CDFLOG
1->  o sts5
-> RUN  CDFOBJ/name $mS5,pri 150/ &
-> HOSTSA 14,                    &
-> NETNAME M14,                  &
-> config config,                &
-> COLLECTOR,                    &
-> SETTABLE $cdf.MDT5.TESTseto
11OCT88  13:41:02  $MS5:  Process started SNAX-CDF (T9094C20 15FEB89)
11OCT88  13:41:14  $MS5:  CDRM M14 active
11OCT88  13:41:14  $MS5:  Configuration is in progress
11OCT88  13:41:21  $MS5:  Activate in progress with CDRM M13 due to ACTCDRM
11OCT88  13:41:22  $MS5:  Link LINK1 active
11OCT88  13:41:22  $MS5:  Link station PU1 has contacted subarea 12
11OCT88  13:41:24  $MS5:  Activate in progress with CDRM M13 due to ACTCDRM
11OCT88  13:41:24  $MS5:  CDRM M13 active
```

Figure 8

```
182>  cmi
CMI - T9394X06 - (01JUL88) - SYSTEM \SNAX
COPYRIGHT TANDEM COMPUTERS INCORPORATED 1982, 1983, 1984, 1985, 1986, 1987
CPU 07, PROCESS HAS NO BACKUP
>START CMP #CMP
>assume line $sdlc1
Assumed object:        LINE \SNAX.$SDLC1
>info ,config
Object: LINE  \SNAX.$SDLC1
Config %004001   %000401   %001403   %003600
       %020040   %000000   %000000   %000764
       %003253   %000764   %000062   %000620
       %140400   %000000   %000140   %000000
       %000403

>start
Address 0          Protocolid 13
>exit
```

Figure 7.
Sample SNAX/CDF initialization from Cool Start File.

Figure 8.
Sample ENVOYACP/XF configuration.

Additionally, SNAX/CDF is structured to conform to the Tandem Distributed System Management (DSM) architecture. SNAX/CDF provides a Subsystem Programmatic Interface (SPI), which enables configuration and management of SNAX/CDF by user-written programs.

As resource entities are added to the configuration by SCF, an optional Cool Start File is updated for use when SNAX/CDF is next initialized or when the backup process takes over. The Cool Start File is a disk file containing an internal representation of previously defined resources. The Cool Start File is specified as an optional initialization parameter to SNAX/CDF. SCF changes the Cool Start File when a resource is started, stopped, altered, or deleted. A sample initialization from such a Cool Start File is provided in Figure 7.

Even if a Cool Start File is not explicitly specified at product initialization, a temporary Cool Start File is created by SNAX/CDF and used in situations when SNAX/CDF is running with a backup process. This temporary Cool Start File provides the backup process with the necessary configuration information to allow it to recover and successfully complete the takeover operation. The file is deleted when SNAX/CDF is stopped.

ENVOYACP/XF. As a privileged Tandem I/O process, the ENVOYACP/XF product must be configured using the GUARDIAN 90 SYSGEN program. (See Figure 8.) The CONTROLLERS and PERIPHERALS macro definitions in SYSGEN are coded with the appropriate values to match the installed Tandem hardware and modem attributes that drive the physical connection (the SDLC line) to the adjacent NCP of the SNA network. For example, Figure 9 presents sample SYSGEN statements for a 6204 bit-synchronous controller and SDLC line.

Care must be taken to ensure that the configured ENVOYACP/XF values match their equivalents for the IBM NCP hardware and software to which the SDLC line is connected. Specifying the wrong values causes failures during activation of the SNAX/CDF PU representation of the NCP. If this is not corrected, connection to the NCP is not possible.

A summary of these configuration elements for SNAX/CDF on the Tandem portion of the SNA backbone network is illustrated in Figure 10.

IBM Configuration

Configuring the IBM portion to communicate with SNAX/CDF is the same as any configuration changes necessary for the addition of another VTAM-NCP domain to the network. All that is needed are the SNA parameters used to configure SNAX/CDF; they are all specified to SNAX/CDF in SNA terminology. The IBM systems programmer, who would be responsible for making the appropriate changes to the VTAM and NCP configurations to connect to SNAX/CDF, needs little if any knowledge of SNAX/CDF or Tandem systems.

Figures 11 and 12 present sample definitions for VTAM and NCP, respectively.

SNAX/CDF Components and Processes

SNAX/CDF works in conjunction with several components and processes. These components and processes provide SNAX/CDF the services of three application programming interfaces for cross-domain communications; the VRSP and CREATOR processes, involved in session establishment; and, for problem determination and management, Event Management Services (EMS) as well as internal and external tracing facilities.

Figure 9

```
DEFINES:

!*************************************************************
!    SDLC MACRO (for bit sync controller)                    *
!*************************************************************
  SDLC^MACRO =  SDLCXF   TYPE 11,   SUBTYPE 40,
                INTERRUPT BS^6203^INTERRUPT^XF,
                FRAMESIZE 2049
                RSIZE  2049 #;

CONTROLLERS:
  BITA          6204          05, 04          %100;

PERIPHERALS:
  $SDLC1        BITA.0,  BITA.1  SDLC^MACRO,  PRIMARY
```

Figure 10

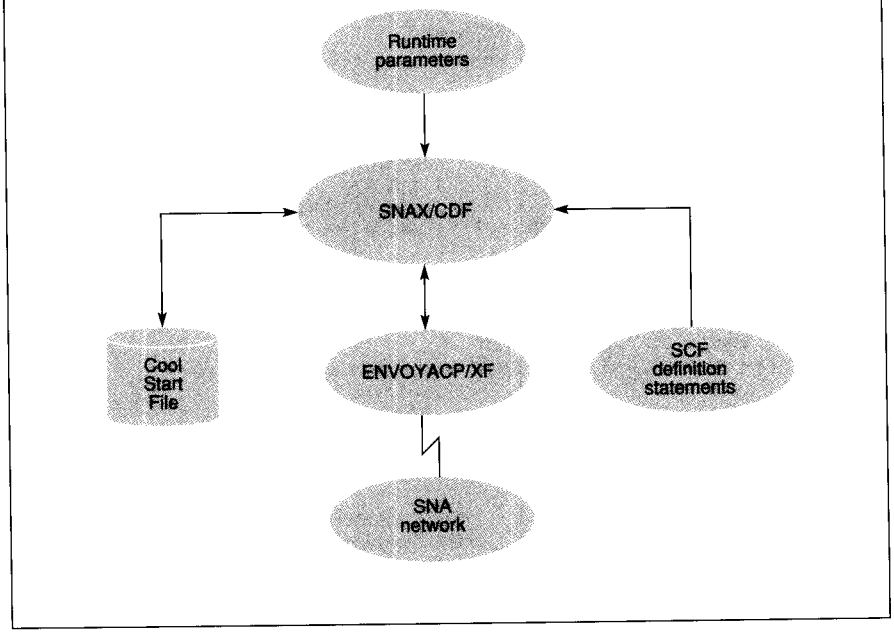


Figure 9. *Sample GUARDIAN 90 SYSGEN statements.*

Figure 10. *Configuring SNAX/CDF.*

Figure 11

```
!*************************************************
!  VTAM PATH DEFINITIONS TO TANDEM SUBAREA 14 FROM SYS1.VTAMLST  *
!*************************************************
 PATH3   PATH   DESTSA=14, ER0=(12,1) , VR0=0
```

Figure 12

```
 ! ***********************************************
1!                PATH MACROS                      *
 ! ***********************************************

APATH1   PATH   DESTSA=(13), ER0=(13,1),    PATH TO PUT5/VTAM HOST        X
                ER1=(13,1),                                               X
                ER2=(13,1)
APATH3   PATH   DESTSA=(14), ER0=(14,1)     PATH TO T16 (SA14)

 ! ***********************************************
 !        LINE GROUP FOR REMOTE 3720 NCP 1         *
 ! ***********************************************

SGROUP3  GROUP  LNCTL=SDLC,                 USE SDLC COMM. TO REMOTE NCP
                TYPE-NCP,                   SPECIFY NCP LINE GROUP
                CLOCKNG=EXT,                EXTERNAL CLOCKED MODEM
                NEWSYNC=NO,                 NEWSYNC FEATURE IN MODEM
                DUPLEX=FULL,                RTS ON
                NRZI=NO,                    NON-RETURN-TO-ZERO-MODE
                RETRIES=(5,1,3)             (M,T,N) M = RETRY COUNT
                                                    T = PAUSE TIME
                                                    N = SEQUENCE REPEAT CNT

 ! ***********************************************
1!        LINE 09  CROSS SUBAREA TG1               *
 ! ***********************************************

LINE09   LINE   ADDRESS=(09,FULL),          RELATIVE LINE #09,TWS          C
                SDLCST=(SDLPRI,SDLSEC)
*
         SERVICE ORDER=PU09C1    SETUP SERVICE ORDER TABLE
*
PU09C1   PU     PUTYPE=4,            SPECIFY PUTYPE AS NCP                 C
                MAXOUT=7,            MAX. PIU'S SENT BEFORE RESPONSE       C
                TGN=ANY              TG #1
         EJECT
```

Figure 11. *Sample ACF/VTAM path definition.*

Figure 12. *Sample ACF/NCP definitions.*

Application Programming Interfaces

Three APIs are provided by the SNAX/CDF product: the CRT, ITI, and SNA Logical Unit (SNALU) interfaces. Through the CRT API, SNAX/CDF also offers remote printer support. SNAX/CDF is backward-compatible at the application level with these three APIs, enabling current Tandem products and user applications to work unchanged with SNAX/CDF.[5] To Tandem applications, SNAX/CDF appears as a Tandem I/O process, which responds to standard GUARDIAN 90 file system requests.[6]

If required, an application can request device information from the SNAX/CDF process (for example, a GUARDIAN 90 DEVICEINFO call) to determine whether it is communicating through SNAX/CDF or SNAX/XF. If it is working through SNAX/CDF, the application then needs to know if it is communicating with CRT, ITI, or SNALU. Because each API supports a particular set of products, knowing the API allows the application to determine if it is communicating with the properly configured subdevice; for example, an application LU (SNALU) or a cross-domain terminal resource (CRT or ITI).

In addition, specific device information can be obtained about the subdevice configured through SCF. The application uses the subdevice definition to determine how to format the data to communicate with the device, since the data format exchanged between the application and the API varies with CRT, ITI, and SNALU.

CRT. The CRT API is a protocol designed to communicate with 3270-type terminal equipment (for example, LU Type 0 or Type 2 sessions) in block mode. This interface is used by block-mode Tandem products such as PATHWAY or TEDIT or by block-mode user-written applications. (See Figure 13.) Although the CRT API interfaces the application to SNAX/CDF, the Tandem or customer application is responsible for the format of the 3270 data in the buffer being written to or read from the 3270 device.

[5]Some minor differences, related to the content of certain session-related Request Units, exist in the SNALU interface between SNAX/XF and SNAX/CDF. See the *SNAX/XF and SNAX/CDF Device-Access Methods Programming Manual* for details.

[6]File system requests are documented in the two-volume *System Procedure Calls Reference Manual* and in the *GUARDIAN 90 Operating System Programmer's Guide.*

Printer Support through CRT and SPOOLER. SNAX/CDF supports remotely attached 32XX printers through the use of the CRT API and the Tandem SPOOLER product. (See Figure 13.) SPOOLER supports 32XX print devices by operating in LU Type 1 SNA character stream mode or in LU Type 3 data stream compatibility mode.

SPOOLER collects text lines written in a line-at-a-time fashion, blocks them, and sends them in block mode to the 32XX printer by way of the CRT protocol. This requires SPOOLER to be configured with a PRINT process called PSPOOLB.[7] User-written programs can also write to printers using the CRT interface, although they are responsible for managing the LU Type 1 or Type 3 protocol.

ITI. The ITI API is a protocol designed to communicate with 3270-type terminal equipment in a pseudo-conversational mode; this mode is also referred to as line-at-a-time. This interface is used by Tandem products such as TACL™ or SCF or by conversational-mode user-written applications. (See Figure 13.)

In contrast to the CRT interface, ITI manages the 3270 data stream because TACL or SCF simply deals with one line of text data to be presented at the terminal with no preference for device type. ITI also handles screen details such as 3270 screen addressing, data translation, cursor placement, data field protection, and scrolling.

Figure 13

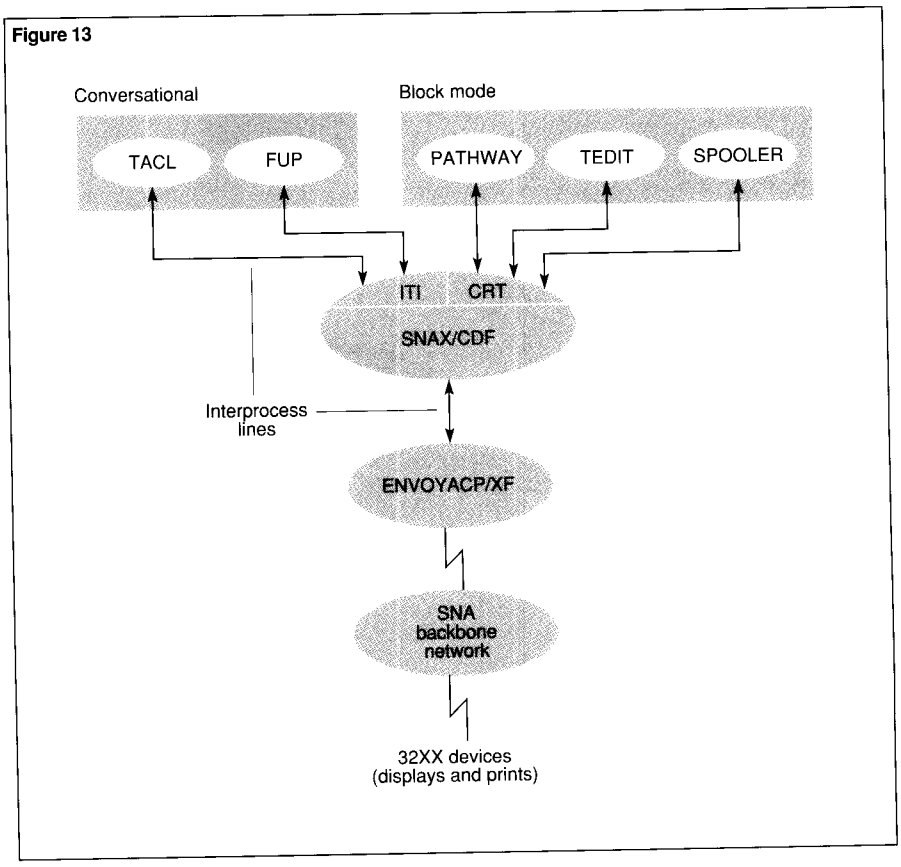
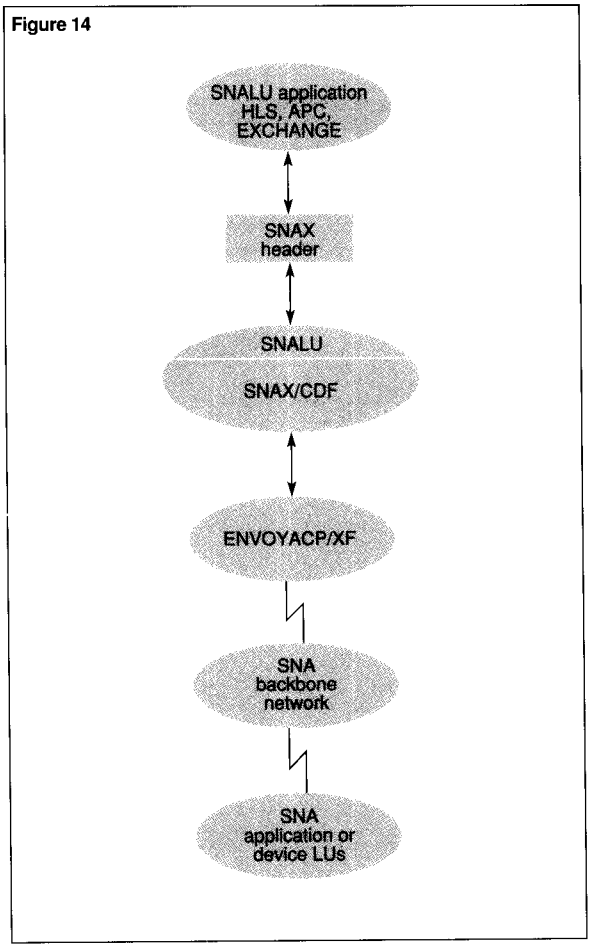


Figure 13.
SNAX/CDF CRT, ITI, and SPOOLER support.

[7] PSPOOLB is provided for the first time in the SPOOLER product included in the GUARDIAN C20 release.

Figure 14.
SNAX/CDF SNALU support.



SNALU. The SNALU interface is a general-purpose API supported by SNAX/CDF. This interface is basically analogous in function to the VTAM API supported on an IBM system. SNALU is used by Tandem products such as SNAX High-Level Support (SNAX/HLS) and SNAX Advanced Program Communication (SNAX/APC) or user-written Tandem application programs and can support a variety of LU session types. Figure 14 depicts the SNAX/CDF SNALU support.

The SNALU interface uses a data unit called a SNAX header that precedes the data in the buffer provided by the particular application. The SNAX header contains control information that defines the I/O operation to be performed by SNAX/CDF.

Processes Related to SNAX/CDF

Two processes provided by SNAX/CDF allow additional capabilities. One, the virtual route selection process, provides a method for controlling the selection of virtual routes during session establishment. The other, the CREATOR process, allows cross-domain logon requests from remote terminal users.

Virtual Route Selection Process. By default, virtual route selection is a task managed by SNAX/CDF in order for it to establish SSCP-SSCP or LU-LU sessions. However, a user can create a Virtual Route Selection Process (VRSP) when that user wants to control the virtual routes that are selected by SNAX/CDF.

A VRSP is a user-written process that can be used to alter or delete virtual route selection by SNAX/CDF. Alteration or deletion of the route selection is accomplished by using a message data unit, defined by SNAX/CDF, called a user virtual route parameter (UVRP) record. The UVRP is passed between SNAX/CDF and the VRSP. (See Figure 15.)

The UVRP is derived from the Class of Service Table (COSTAB), which is configured by the SNAX/CDF user utilizing definitions supplied to SCF. The COSTAB indicates the virtual routes that SNAX/CDF can use to establish SSCP-SSCP or LU-LU sessions. If a VRSP is not provided by the user, SNAX/CDF relies on the configured COSTAB when selecting virtual routes for sessions.

CREATOR Process. Another component of the SNAX/CDF product is the CREATOR process. This is the same process that is provided with the SNAX/XF product. CREATOR allows a terminal user to log on to SNAX/CDF from a cross-domain terminal and start a session with TACL or a PATHWAY application.

The CREATOR process uses the GUARDIAN 90 file system as its interface to SNAX/CDF. Figure 16 illustrates the relationship of the CREATOR process to SNAX/CDF.

To use CREATOR, the SNAX/CDF user defines an application LU by way of the SCF product. This allows the user to specify application usage (such as defining the range of the number of sessions that can be started) and other parameters relevant to the execution of the specific application running under control of CREATOR. The *SNAX/XF Configuration and Control Manual* documents these different parameters.

The CREATOR process must be explicitly started by the system operator as part of the SNAX/CDF startup procedure. CREATOR opens the defined application LU by opening the specific subdevice indicated on the application LU definition. When that is accomplished, CREATOR then awaits session creation and termination requests from SNAX/CDF.

Additionally, data records must be included in the IBM *Unformatted System Services* (USS) tables that define the Tandem applications available to the cross-domain terminal user.

Figure 15

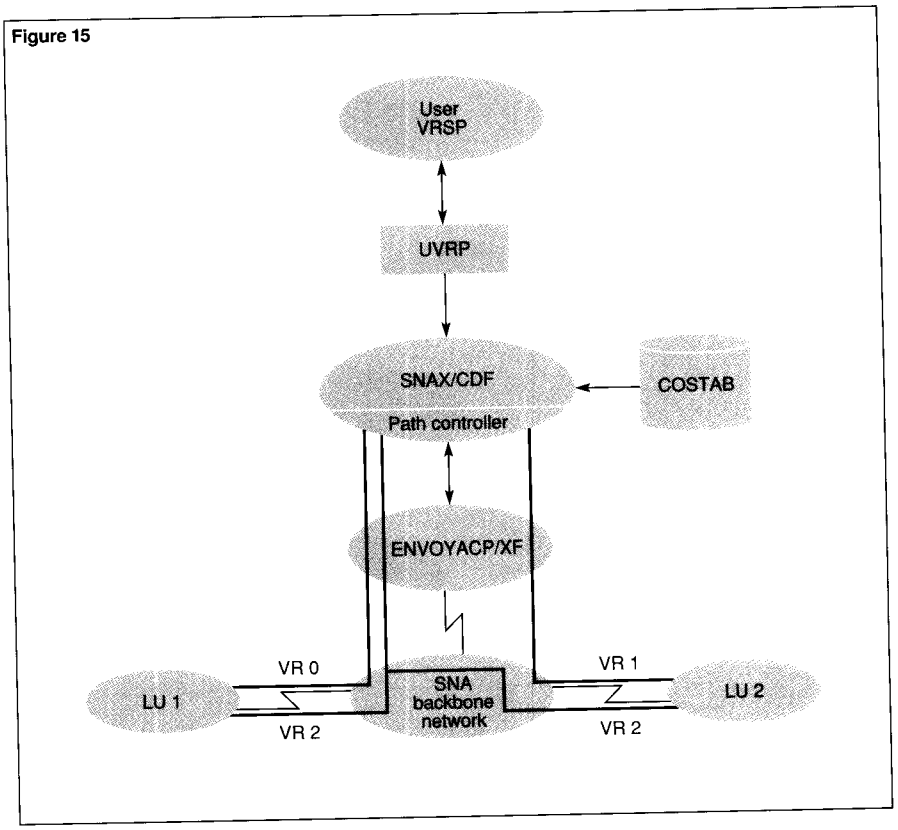
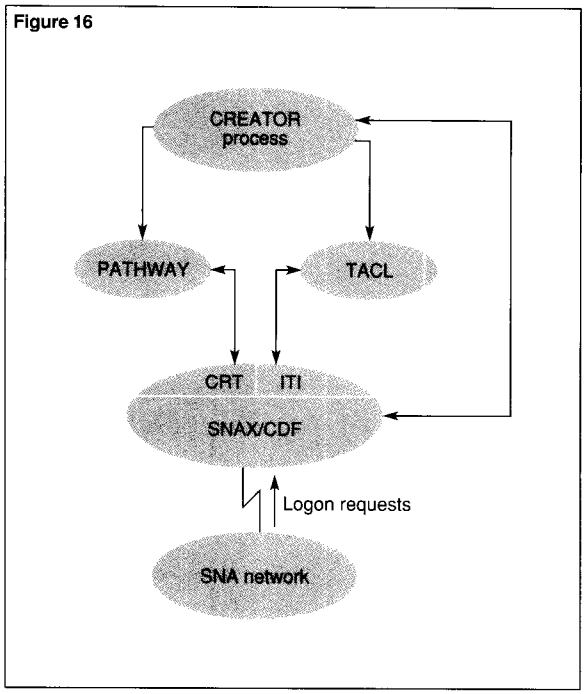


Figure 15.
SNAX/CDF virtual route selection.

Figure 16.
SNAX/CDF and CREATOR.



Alternately, parameter data supplying the same information as the USS table must be included at the time of logon from the cross-domain terminal. However they are provided, these records contain specific parameters to be used by CREATOR (for parameter information, refer to the *SNAX/XF Configuration and Control Manual*). For TACL, for example, the logon data describes that TACL is to be invoked through the GUARDIAN 90 NEWPROCESS call and specifies which optional parameters it will receive. For PATHWAY, the logon data describes which PATHWAY system to use and what PATHWAY application should be started.

The following sequence summarizes the CREATOR and SNAX/CDF relationship.

- A cross-domain logon request, destined for a TACL or PATHWAY application that is defined as having access to CREATOR, is received by SNAX/CDF.
- SNAX/CDF selects a subdevice within the specified range of subdevices defined for that TACL or PATHWAY application.
- SNAX/CDF gives the logon request to CREATOR to pass to TACL or PATHWAY. The first output request by the application starts the SNA LU-LU session.
- CREATOR is no longer involved until SNAX/CDF or the cross-domain LU terminates the session.
- Upon session termination, SNAX/CDF indicates this to CREATOR and frees the subdevice used in the session for use in a subsequent session.

Problem Determination

Management of any problems that might occur involving SNAX/CDF are handled in two ways. SNAX/CDF interfaces to the Tandem Event Management System (EMS) and also provides tracing facilities for problem analysis.

EMS. A part of the DSM architecture, EMS is used to collect and distribute events generated by a variety of Tandem products. The events generated by SNAX/CDF relate to resources under the control of SNAX/CDF, and the events generally report exception conditions or major state changes.

SNAX/CDF events are divided into three categories:

1. Configuration and startup events provide information related to the initialization of SNAX/CDF.
2. Logical error events describe software errors detected within SNAX/CDF.
3. Session events indicate activation/deactivation/errors with resources under the control of SNAX/CDF.

Examples of a configuration and startup event include SNAX/CDF process startup messages and messages indicating the progress of SNAX/CDF initialization according to the user's SCF configuration definitions. Logical processing errors or finite state machine errors are examples of logical error events. Events of this type generally indicate a serious problem within SNAX/CDF and should be reported to a Tandem representative. Session events reported for SNAX/CDF generally indicate configuration problems or trouble within the attached network.

Tracing Facilities. SNAX/CDF provides very comprehensive tracing facilities that supply both an internal and an external trace. These tracing facilities are detailed in the *SNAX/XF Configuration and Control Manual*. These traces enable analysts to more quickly isolate a problem because they provide extensive flow and component information.

While SNAX/CDF is executing, the SNAX/CDF internal trace runs constantly. Tracing is kept to a minimum to minimize performance degradation. The internal trace ensures that, in the event of a problem with SNAX/CDF, enough information is available in a program dump to enable a speedy analysis. This is especially helpful in most production situations where, typically, external traces are not enabled. The internal trace is controlled by parameter specification to SNAX/CDF at the time of initialization.

The SNAX/CDF external trace, controlled by the SCF TRACE PROCESS command, allows the specification of a significant number and types of trace points. Such a large range of trace points greatly enhances the chances of quickly identifying the cause of a particular problem. The data is traced to a disk file for later examination using the PTRACE trace formatting facility.

Conclusion

The SNAX/CDF product provides new connectivity between Tandem and IBM networks. SNAX/CDF allows IBM terminals connected to an SNA backbone network access to Tandem applications. In addition, users on their Tandem systems can use SNAX/CDF to access IBM applications on an SNA network. SNAX/CDF provides configuration in strict adherence with Tandem current SCF configuration technology, while also presenting this configuration in a fashion very similar to IBM VTAM definitions. By providing backward compatibility for existing applications, SNAX/CDF can protect investments in software development. The ability of SNAX/CDF to implement a broadened access to network resources outside the Tandem environment illustrates the Tandem commitment to SNA as a means of providing enhanced OLTP connectivity.

Advanced Communication Function (ACF)

Used in the full name for the VTAM and NCP software products. Refer to the definitions for VTAM and NCP.

Application Program Interface (API)

A pseudonym for the Transaction Services layer of SNA, an API acts as the layer that interfaces directly to the application program. An API, provided by the SNAX/XF product, is responsible for translating functional requests sent from the Tandem system or received from the SNA system into a format that the underlying SNA layers understands.

Backbone Network

The portion of a network that is the transport path between major network components controlling the routing and end point delivery of messages in a network. An SNA backbone network is the portion of the network that connects SNA host (PU Type 5) components and SNA communications controller (PU Type 4) components.

Cluster Controller

An IBM hardware control unit that contains a PU function. The cluster controller, known also as a PU Type 2, is responsible for controlling LUs attached to the PU. IBM 3272, 3274, and 3174 control units are examples of a cluster controller.

Communications Controller

An IBM front-end processor that runs the IBM Network Control Program and is the interface between the host and the subarea backbone network. Examples of communication controllers, also known as a PU Type 4, are the IBM 3705, 3720, 3725, and 3745.

Class of Service Table (COSTAB)

A user-defined table of information that defines virtual route and transmission priority information for applications. This information is encoded such that alternate virtual routes can be selected in the event of the failure of an existing active virtual route. The route selection criteria can be dynamically selected by the user using the Virtual Route Selection Exit in VTAM or the Virtual Route Selection Process in SNAX/CDF.

Cross-Domain Session

A cross-domain session is defined as a session between a resource (also referred to as a network-addressable unit, or NAU) in one domain and a resource, or NAU, in a different domain. See also NAU.

Cross-Domain Session Resource Manager (CDRM)

A subcomponent of the SSCP that is responsible for controlling cross-domain (also called xdomain or xdom) sessions between CDRMs.

Domain

A collection of subareas controlled by an SSCP. The size of the given domain is dependent on the size and configuration of the total network and is governed by the portion of the network geographically and logically assigned to the given domain.

Exchange ID (XID)

An SDLC frame exchanged between SNAX/CDF and NCP at link activation time. XID supplies communication parameters used by SNAX/CDF and NCP.

Explicit Route

A logically defined path using one or more transmission groups that defines a definite route between two subareas. An explicit route is identified by an origin subarea address, a destination subarea address, an explicit route number, and a reverse explicit route number.

Host

A processing unit that contains all or part of the user applications, the SNA access method, and an SSCP. Also known as a PU Type 5, examples of a host would be an IBM 3090 or 4381 with VTAM or a Tandem VLX with SNAX/CDF.

Logical Unit (LU)

A logical unit is a port that allows access to the SNA network for the transfer of data. Examples of an LU would be an application program, an IBM 3270 display station, or an automated teller machine. There are several types of LUs,* including:

- LU Type 0 defines an application-to-terminal LU relationship where the specifics of the exchanged data format are defined by the LU partners. This is most commonly used with non-SNA 3270 terminals. LU Type 0 is also used for application-to-application data exchange where a user-defined data exchange is defined.
- LU Type 1 defines an application-to-printer LU relationship where the SNA Character Stream (SCS) data format is used.
- LU Type 2 defines an application-to-terminal LU relationship where the 3270 data stream is used. LU Type 2 is used with SNA channel or link-attached 3270 terminals.
- LU Type 3 defines an application-to-printer LU relationship where the 3270 Data Stream Compatibility (DSC) data format is used.
- LU Type 6.2 defines an application-to-application LU relationship where the two LUs exchange data basic to the rules of LU 6.2 Advanced Program-to-Program Communication.**

Network-Addressable Unit (NAU)

A network node (SSCP, PU, LU) that can be directly addressed using the SNA routing architecture as the end point of some network-related traffic. Traffic is first routed to the subarea and then sent to the NAU within the subarea. The NAU address is a combination of subarea and element.

Network Control Program (NCP)

The IBM software that runs in an IBM communications controller unit (PU Type 4).

Peer Functional Component

In SNA, all layers of the SNA architecture exist within each network component in a peer (duplicate or mirrored) fashion. For example, in the SNA sense, SNAX/CDF has the same peer functional components in the Tandem system as does VTAM and NCP on the IBM side.

Peripheral Nodes

The actual link, PU, or LU devices that are used by end users for transmission of user data. Peripheral nodes are adjacent to and controlled by subarea nodes. Examples of peripheral nodes would be a SNALU application running on a Tandem VLX or an IBM 3174 cluster controller attached to an IBM 3725 communications controller with a 3278 display station.

Physical Unit (PU)

An SNA component that provides the services for the SSCP-PU session and is the controlling intelligence for any associated LUs. There are several types of PUs:

- PU Type 2.0 is defined as a cluster controller. (See Cluster Controller.) These devices work in a master-slave relationship, the PU Type 4 or 5 being the master and the PU 2.0, the slave.
- PU Type 2.1 is defined as an intelligent physical unit. It can communicate with other PU Type 2.1 or Type 4 nodes as a peer physical unit with a negotiated role of master or slave, which is based on Exchange ID at link connection time. (See Exchange ID.)
- PU Type 4 is defined as a communications controller. (See Communications Controller.)
- PU Type 5 is defined as a host. (See Host.)

Synchronous Data Link Control (SDLC)

A bit-oriented, link-level communication link protocol as defined by IBM. Used by ENVOYACP/XF over the Tandem 6204 controller.

Session

A logical connection between two network-addressable units that has been created by exchanging pertinent information contained in SNA requests and responses associated with network-addressable units. For example, a BIND request and its related positive response create an LU-LU session. Other logical connections include SSCP-PU, SSCP-LU, and SSCP-SSCP (CDRM-CDRM) sessions.

*The *IBM 3174 Subsystem Control Unit Functional Description* provides more details regarding SNA character string data format, SNA channel or link-attached 3270s, and data stream compatibility data format for LU Types 1, 2, and 3, respectively.

**Refer to the *IBM Format and Protocol Reference Manual: Architectural Logic for LU Type 6.2*.

Subarea

A network-addressable unit that consists of a subarea node and any attached peripheral nodes with their associated resources. The subarea node is directly addressable within the routing design of SNA, and the subarea contains resources subordinate to it that are subaddressed by specific, unique element addresses within the subarea node. This combination of the subarea and the element routing structure (also known as an NAU) comprises a subarea backbone network.

System Services Control Point (SSCP)

A network-addressable unit that resides in a host (a PU Type 5) and is responsible for the configuration, control, and error recovery of related end-user nodes in a hierarchical relationship between PU and LU nodes. SSCPs, in a peer session with each other, divide the network into domains of control.

Transmission Group (TG)

A logical association of physical communication links between adjacent subarea nodes that gives the appearance of a composite communication link. A TG can be a single- or multiple-link association.

Unformatted System Services (USS)

A facility within VTAM that defines two facilities: (1) a character string syntactical command sequence by which VTAM recognizes logon sequences to applications, either same or cross-domain to VTAM, and (2) a set of messages that can be generically displayed on VTAM-owned terminals for particular situations (for example, a good morning or terminal identifier message). The character-coded logon sequences are then translated into SNA-defined message units. These are then used to perform session initiation. USS tables define sets of these messages.

Virtual Route

A logical path mapped over one or more explicit routes that defines the virtual path from an origination subarea to a destination subarea. Virtual routes provide paths over which SNA transmission units flow. They employ a transmission priority scheme for prioritizing data flow, a virtual route pacing mechanism to control network route congestion, and a sequence number to maintain virtual route integrity.

Virtual Telecommunications Access Method (VTAM)

The name for a distinct component of IBM host-resident software that is the main IBM emphasis for SNA communications in subarea networks.

References

GUARDIAN 90 System Generation Manual, Volume 1. Part No. 84094. Tandem Computers Incorporated.

System Procedure Calls Reference Manual, Volume 1. Part No. 84118. Tandem Computers Incorporated.

System Procedure Calls Reference Manual, Volume 2. Part No. 84119. Tandem Computers Incorporated.

GUARDIAN 90 Operating System Programmer's Guide. Part No. 84083. Tandem Computers Incorporated.

ENVOYACP/XF Bit-Oriented Protocols with Extended Functions Reference Manual. Part No. 84116. Tandem Computers Incorporated.

Communications Management Interface (CMI) Operator's Guide. Part No. 84056. Tandem Computers Incorporated.

SNAX/XF and SNAX/CDF Device-Access Methods Programming Manual. Part No. 29805. Tandem Computers Incorporated. Check with your Tandem representative regarding availability during 1989.

SNAX/XF Configuration and Control Manual. Part No. 26889. Tandem Computers Incorporated. Check with your Tandem representative regarding availability during 1989.

IBM Systems Network Architecture Format and Protocol Reference Manual: Architectural Logic. SC30-3112. International Business Machines Corporation.

IBM Systems Network Architecture Technical Overview. GC30-3073. International Business Machines Corporation.

IBM ACF/VTAM Installation and Resource Definition. SC23-0111. International Business Machines Corporation.

IBM ACF/NCP Installation and Resource Definition. SC30-3253. International Business Machines Corporation.

IBM ACF/NCP Resource Definition Reference. SC30-3254. International Business Machines Corporation.

IBM Synchronous Data Link Control Concepts. GA27-3093. International Business Machines Corporation.

IBM 3174 Subsystem Control Unit Functional Description. GA23-0218. International Business Machines Corporation.

IBM Format and Protocol Reference Manual: Architectural Logic for LU Type 6.2. SC30-3269. International Business Machines Corporation.

Acknowledgments

The author wishes to thank the past and present members of the SNAX/CDF development team—Greg Bostrom, Michael Brooks, Mei Chen, Bill Dalton, Robert Haynie, Delung Lin, Bill Naylor, and Hai-Szu Yu—for their part in developing SNAX/CDF, and Margo Holen, SNAX/CDF manager, for keeping us all in line. Also, special thanks go to all those who reviewed this article and to Chris Russell, IBM section manager, for providing the section titled "The Development of SNAX/CDF."

Marty Turner has been involved in SNA since 1980 and has worked in the data processing field since 1969. He has been a member of the SNAX/CDF development team since 1985, when he joined Tandem. Prior to working at Tandem, Marty held positions in support and development with major computer hardware and software manufacturers.

Network Design Considerations

Timely access to various data is a critical, integral part of the effective operation of most organizations. Business success or failure can often depend on employees' ability to make fast, thorough, and accurate decisions based on the most current information available. The need for users to have reliable access to up-to-date information at local and remote locations gives rise to the development of computer and terminal networks.

A properly designed data network can provide an organization with an effective means for reducing costs and a more efficient operation of the organization. Conversely, an improperly planned network can inhibit the efficient use of system resources, interfere with the functioning of the organization, and even cause the loss of customers. The adverse effects of a poor network design underscore the importance of designing networks that have the facilities and flexibility to meet current and future demands of the network users.

This article presents a high-level overview of the various considerations involved in the effective design of data networks. It is intended to be a discussion of the issues and decisions that a network designer needs to make when designing or modifying a network. The importance of a careful and thorough analysis of the network requirements as well as the implications of the available design alternatives are stressed. In addition, the article discusses some techniques and methodologies that can be of use in the network design process.

Effective Network Design

In general terms, the network design process is composed of a number of tasks:

- Determining network objectives and the performance and reliability requirements.
- Gathering information regarding the network users and the activity level that the network is expected to handle.
- Sizing of the components to determine the appropriate capacities.
- Selecting the network components.
- Determining the most effective connection configuration, or topology, for linking these components together in a network.

Methods and tools are available to help the network designer arrive at the optimum design, and these are described later in detail.

The general goal of the design process is to allocate network resources in a cost-effective manner that best accommodates the requirements of the users. The desired end product is a complete description of the network, its components, and its functions.

A critical factor not to be overlooked in the network design process is the expertise of the people involved in the network planning and design. As well as having a solid background in data communications, a network designer should be up-to-date on new products and technologies in order to evaluate all technical alternatives. In addition, the designer should understand network design concepts; know the implications in trade-offs between line utilization, response times, and queueing delays; and be familiar with the different design methodologies that are appropriate for the various topologies. Finally, and most importantly for the project's success, a network designer must thoroughly understand the organization's business needs in order to design a data network that specifically accommodates those needs.

Beginning the Design Process

Designing an effective network requires that the objectives and requirements of the network, including the performance requirements and reliability standards, be defined. Then, accurate and comprehensive data describing the work that the network is to support must be obtained or estimated. Using this information, appropriate methods and techniques can be applied to arrive at a satisfactory design.

General Objectives and Requirements

The value of a data network to an organization is measured by its ability to support operational, tactical, and strategic objectives of the organization. Determining and establishing the objectives and requirements of a network is the most important step in the design process. To accomplish this first crucial step, the designer must examine, among other things:

- Why the network is being developed.
- What functions and service levels are to be expected from it.
- The defined scope of the network.
- The organizations affected by it.

Determining overall objectives and requirements necessitates frequent and thorough contact with the users of the business systems to understand their needs and the needs of the organization. Each organization has its own unique requirements and flow of information.

Several basic decisions need to be made concerning the overall structure and facilities provided by the network. For example, a group may need to decide whether voice and data networks are to be kept separate or if they are to be integrated into one network. The trend today is toward integrated networks, many of which attempt to integrate voice, data, facsimile, and video transmission over high-speed lines.

In some organizations it may be beneficial or necessary to separate certain critical applications or functions into separate networks. This may be done for a number of reasons, including cost, security, or other considerations unique to the organization. If only one network is chosen, there are the potential problems of some applications affecting other applications, especially if they share communications media. On the other hand, separate networks make the total management more complex, and the total costs of the networks may be greater than for a single network.

Cost and performance are two factors of network design.

Cost and performance goals are generally important factors in setting design requirements and objectives. However, they are not the only criteria to determine an effective design. Many other factors, such as priority, availability, and sensitivity of certain users or applications, often impose serious constraints on the network design. The network must be designed to meet the goals, needs, and preferences of the specific organization.

Performance Requirements

The network must have the capacity to provide consistent and satisfactory service to all users of the network.

Performance requirements for users of terminal networks are most often stated in terms of response time—the time taken to send a message to the destination host system and receive a reply. Usually, response time is more precisely defined as the length of time it takes to either receive a complete reply or the first character of a reply message.

For distributed computer systems communicating on a peer-to-peer basis, the performance requirements can specify the acceptable range of times for the average end-to-end delay for a packet. However, to be meaningful to the end user, this delay must be further defined in terms of the expected response time for the applications being accessed.

Performance requirements and the associated response times for users should be based on the level of performance that is required to sustain business without adversely affecting the overall functioning of the organization. Generally, there is a defined time value to the information travelling through the network. A delay beyond that time may cause that information to have limited value. For example, a user accessing stock market quotations needs very timely information in order to buy and sell stocks at the appropriate times to maximize profits and minimize losses.

A short response time can be critical for many interactive applications involving the public. Some applications are so critical that it is disastrous for the business if the response times are not consistently of short duration. If too much delay is encountered, the company may lose customers to competitors.

Defining Performance Requirements. Performance requirements are often defined as an average response time, a maximum response time for a certain percentage of transactions occurring in the network, or a combination of the two criteria. The percentage of transactions is frequently set at 90% or 95%. A typical performance requirement might be that 95% of the transactions should have a response time not exceeding 3 seconds. The criteria selected should be reasonable and practical; attempting to push the percentage of transactions to 100% may rapidly increase costs to a point where very little is gained for large increments in expense.

The value of the desired response time must be a realistic value that takes into consideration the message sizes, queueing delays, and the amount of processing and database accesses required at the host to service a transaction. For example, if the host system takes 2 seconds to process a transaction and the message lengths are long, it may be difficult or impossible to keep the average response time to less than 3 seconds unless the host processing time can be reduced.

Traffic Flow Rates. Related to the response time is the determination of what rate of traffic flow the network is being expected to handle. The design criteria is often defined as to whether the network should be able to accommodate the average or peak traffic flows.

Peak traffic flows may be sudden surges at certain times of the day or at certain periods of the week, month, or year. The designer must have data that is as complete as possible on the traffic that is anticipated for the network, including future growth expected over the lifetime of the network. A design for peak flow is more expensive, and some of the network capacity is not used during non-peak times. However, using average rates may result in a system that soon reaches its capacity limits and may not provide adequate growth capacity.

To solve this problem, designers often prefer to size the network for a value that is somewhere between the average and peak flow values and can provide satisfactory service for the majority of the users most of the time. The general goal is to provide adequate service for most of the time but to allow a somewhat degraded service during short periods of unusually heavy peak traffic flow. The appropriate level of service is heavily dependent on the organization and the types of applications running on that network.

Network Reliability

The network must be reliable and available to provide all necessary service to its users. Users judge the network on its ability to provide consistent levels of performance, availability, and integrity. Networks unable to provide a minimum standard of availability and performance waste money and resources for the organization and, due to situations that impair the organization's ability to conduct business successfully, can lead to reduced revenues.

The probability that the network might fail under stress as well as other types of failure must be assessed to determine the cost of the failures. The more costly a failure is, the more important it is to have a design for the network that minimizes the effects of failures. The network needs to have sufficient flexibility built into it to provide for new users, new applications, new technologies, and future growth and expansion. This also means that the resulting network must be capable of being satisfactorily managed to react to failures and other abnormal situations.

Gathering Design Information

The designer must first have a complete picture of the users of the network and exact, detailed data on the amount of traffic activity that the network is expected to handle in order to determine the capacities needed within the network. Some of the general requirements of the network may become redefined as more information about the operations of the users is obtained.

Gathering accurate data is probably the most difficult step because the data is often not readily available in the required format. However, it is one of the most important steps. The accuracy of the information directly affects the design precision, network performance, and network cost.

What Information Is Needed?

The general information a designer requires includes the following:

- What locations are included in the network.
- What applications are used.
- What resources these applications require.
- How much traffic these applications generate into the network.

From each location, there is information that must be transmitted to other locations. The sources and destinations of the information must be identified, and all the expected traffic through the network, including interactive, batch, and file transfer, must be determined. Other elements to consider include the number of applications that network users are running, how frequently these applications are used, and any unique requirements for each user application.

For interactive applications, complete profiles of the transactions are necessary: message sizes, terminal processing times, think times for the host system users, protocol overheads, and other processing delays. In the case of batch and file transmissions, the file sizes and the maximum transmission time allowed must be determined. Some data compression may occur for batch and file transmissions, significantly reducing the amount of data to be transmitted.

If existing systems are part of the network being designed or modified, it is necessary to have information about these components; for example, the protocols in use, line speeds, and hardware delays. If available, performance information about the existing system can be used as a baseline and can help the designer to determine if there are any current problems. The new or modified design should not adversely affect the existing performance levels.

While gathering data on network activity, the ideal goal is to have information on the traffic from all locations and all users. However, in some situations, this is not possible or practical. As a result, it may be necessary to build a sample of the predicted network activity by selecting samples of network users at appropriate times.

Such a sample must be sufficiently representative to predict the absent users and thus reflect the requirements of the whole population of users. The time periods selected must be representative and coincide with the performance requirements established for the network. Consideration of seasonal variations and other traffic periods that the network is being designed to accommodate must be included.

Frequently, some preliminary study is needed to determine the patterns of the data traffic. The amount of traffic in the peak periods is then determined by measuring and data gathering. It is not sufficient to simply take a day's or month's worth of data then divide it by the total number of hours to arrive at an average figure. The difference between a non-busy period and a busy period may be very great. If some periods are very light and some very heavy, an average calculated in this way often underestimates or overestimates the traffic that the network should be designed to handle.

Ways to Obtain Data

It may be necessary to use several different methods to get sufficient data to proceed with the design process. Often, the method chosen to gather information depends on the type of network a designer is studying.

An Existing System. For existing systems, a good source for at least part of the data should be the results of performance analyses and measurement of network activities. This data can be gathered from software and hardware counters and accumulated statistics. Tandem programs such as the MEASURE™ System Performance Measurement Tool, Network Statistics System (NSS), Communications Management Interface (CMI), Subsystem Control Facility (SCF), and others can provide useful information. Other sources may include traces, manual tracking, response-time monitors, and system and program specifications.

The designer should ensure that all offered traffic is determined, not merely the "good" or successful activities. For example, if dial-in inquiries are lost because of busy circuits, the unsuccessful inquiries probably are not recorded. It is important to estimate any lost traffic to provide adequate facilities to handle these calls successfully.

For dial-up telephone service and public packet data networks, the carriers and call accounting devices can provide information about the number, duration, destination, and cost of the calls. Traffic activity can also be monitored through application software, including transaction files and audit trails.

New Network or Applications. If no network exists or one or more new applications are being added, it may be necessary to do some preliminary design of these systems to get adequate data. This may require talking to the end users of the current manual systems or examining current manual methods of recording activities, such as sales slips or invoices, to determine what functionality and capacity is needed in the new systems.

The information obtained should always be checked for reasonableness and completeness. In many situations, some assumptions need to be made, and the data must be adjusted to account for incomplete information or anticipated growth. Also, user estimates are frequently low; it is often not possible to accurately predict the increased traffic into the network that may occur as a result of the availability of new functions. The assumptions made and the omitted data that is extrapolated or predicted should be noted because they have implications for the validity of the end results of the network design.

Designing the Network

The actual network design involves sizing and connecting the various components through communications media to form a network that meets the established objectives and criteria. Because each has an effect on the other, the host, terminal, and concentrator sizing implications must be considered together with the line sizing and routing. All these elements must work together to provide satisfactory service to the users.

Generally, the goal is to arrive at a network configuration that offers an economical design yet maintains a desired level of reliability and provides an acceptable response time or intersystem delay. The network design should be based on the flow of information and the organizational structure and should facilitate the flow of information and data to the end users. In addition, it should be engineered to accommodate future growth; it is generally cheaper and quicker to add to an existing network than to completely redesign it.

Balancing trade-offs between costs, reliability, and performance is not an easy task.

Balancing the many trade-offs between costs, reliability, and performance is not an easy task, especially when dealing with large networks. Many alternatives may be used to arrive at a satisfactory network design.

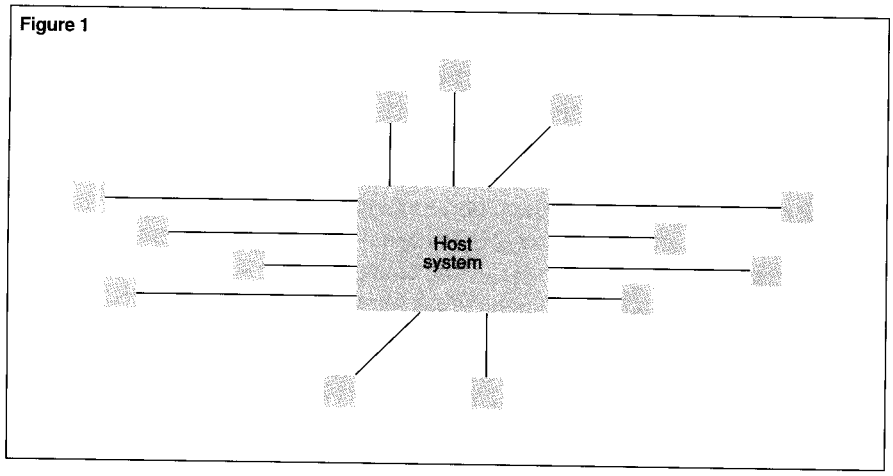


Figure 1.
A point-to-point network.

Network Topology

The topology defines the physical structure of the network; it should follow and facilitate the flow of information within the network. In many cases a network is made up of several subnetworks, each of which may have a different topological structure.

Topologies can be divided into two general types of networks: hierarchical and distributed. For this discussion, hierarchical means that information flows between lower and higher levels. This can simply be terminals connected directly to the host system or can include intermediate levels of devices such as cluster controllers, multiplexers, and concentrators.

Hierarchical systems typically have one main host system, although they can have more than one host system. Terminals are connected to the host by point-to-point or multipoint lines. Sometimes concentrators, multiplexers, or other devices are used to concentrate the connection of the terminals. The host system is the main destination of the traffic, and typically there is very little communication between the individual terminal sites. If there is a need for communication, it is through the host system. The host system also usually contains most of the databases.

There are many design variations possible using a hierarchical topology, and many different types of equipment can be used. A point-to-point network, a multipoint network, and a hierarchical network with multiplexers and concentrators are three commonly used hierarchical topologies.

Distributed topology refers to flow of information between peer-level systems. This topology can also contain hierarchical elements, especially if a backbone of switching nodes is part of the network. It is very likely that a given network is a hybrid where part of the network is hierarchical and part is distributed.

Point-to-Point Network. This type of network, the simplest form of the hierarchical topology, provides direct connection of each terminal or cluster controller to the host. (See Figure 1.) Each device is connected by a separate line forming a star arrangement, where the lines radiate out from the host system. Point-to-point lines are also used in other interconnections such as host systems to other host systems and concentrators and multiplexers to hosts.

Generally, there is no polling delay unless the point-to-point line involves a line-sharing device, which requires each attached device to be polled individually. If each terminal is connected by a separate line, the result should be optimum response times. The point-to-point network has high reliability because a line failure or misbehavior of a terminal only affects that one line. However, this topology generally has the highest communications lines cost, and it usually requires one communications port at the host for each site in the network. To reduce costs, it is often preferable to have several locations share a line in a multipoint arrangement.

Multipoint Network. This type of hierarchical topology can frequently offer substantial savings in terms of communications costs. In most instances, multipoint configurations are only used to connect terminal devices or cluster controllers, as illustrated in Figure 2, rather than provide host-to-host or concentrator-to-host connections. This network arrangement requires the use of a polling protocol, which adds a polling delay to the response time. In order to keep this delay to an acceptable figure, the number of points per line may have to be limited.

A multipoint network is not as reliable as a point-to-point network. Failure of a line generally brings down all the devices on the line, and there are more potential failure points on the line. Line failures can be partial in some cases, leaving some of the devices still in service. Many multipoint designs provide for backup, either over a redundant line or as a dial-up connection, to ensure satisfactory service for the users of the network in the event of line failures.

When designing a multipoint network, the designer must determine which points are placed on each line as well as how many points and devices to put on each line. To do this, costs and distances, as well as the traffic from the devices, must be considered in order to arrive at lines that are optimum in cost but meet the performance constraints.

It can be very risky to use simple rules of thumb, such as maximum number of points or terminals on a line, unless there is a uniformity of use at each terminal or location. A better approach involves considering the amount of traffic from each potential point as well as the amount of polling delay incurred. In this way, it is possible to prevent the overloading of lines and excessive polling delays, which result in unsatisfactory response times. In addition, too many devices on a line can cause severe disruption to many users if the line should fail. Finally, the designer must consider any total circuit distance limitations imposed by the communications carriers that supply the lines.

Hierarchical Network with Multiplexers and Concentrators. A third common hierarchical topology provides a network that uses concentrators, multiplexers, or both. (See Figure 3.) Including these components often reduces line costs, especially when compared to point-to-point configurations. Concentrators can provide valuable remote processing, reducing the user response times and the load on the network lines. In the Tandem environment, a smaller EXT™ or CLX™ system can serve as a concentrator and provide local processing for the terminals connected to it. Two disadvantages are (1) that the concentration device might fail, bringing down all the devices attached to it, and (2) that it adds additional delay to the response time and means additional cost for the network. Careful selection of the concentrator device ensures that it is both reliable and efficient; this minimizes potential problems.

Figure 2

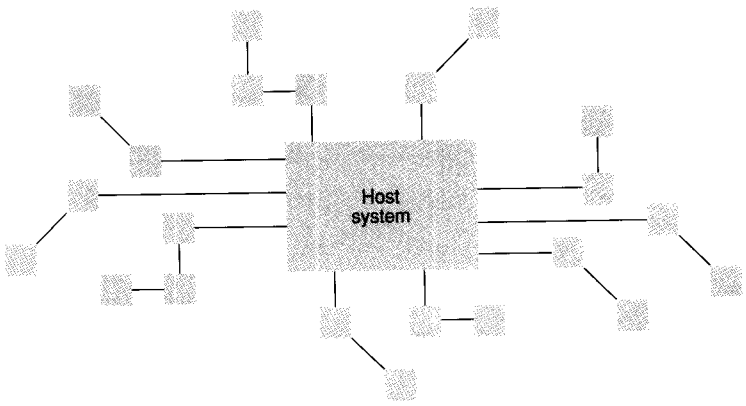
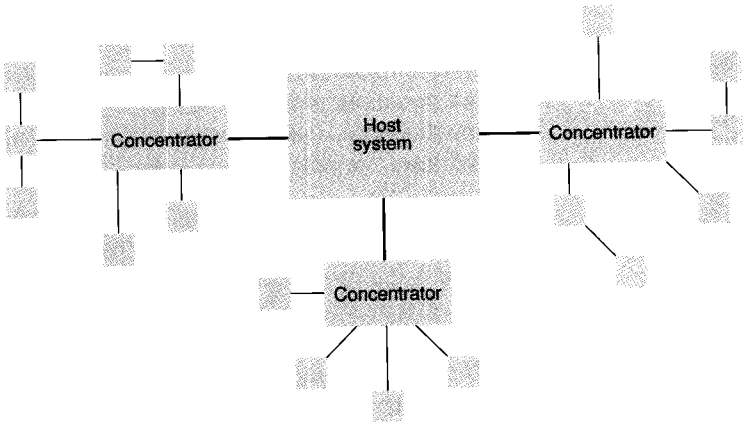


Figure 3



The designer decides where the concentration devices are located and which devices are connected to them. These decisions are generally based on cost and performance trade-offs, but they also include considerations unique to the company such as adequate facilities for the equipment, trained personnel, and proximity to maintenance support. If the user wants processing in addition to line concentration, the designer must ensure that the concentration device can handle both the planned processing and communications load; this prevents the creation of any bottlenecks. Limiting the number of terminals connected to the concentration device may be required.

Figure 2.
A multipoint network.

Figure 3.
A hierarchical network with concentrators.

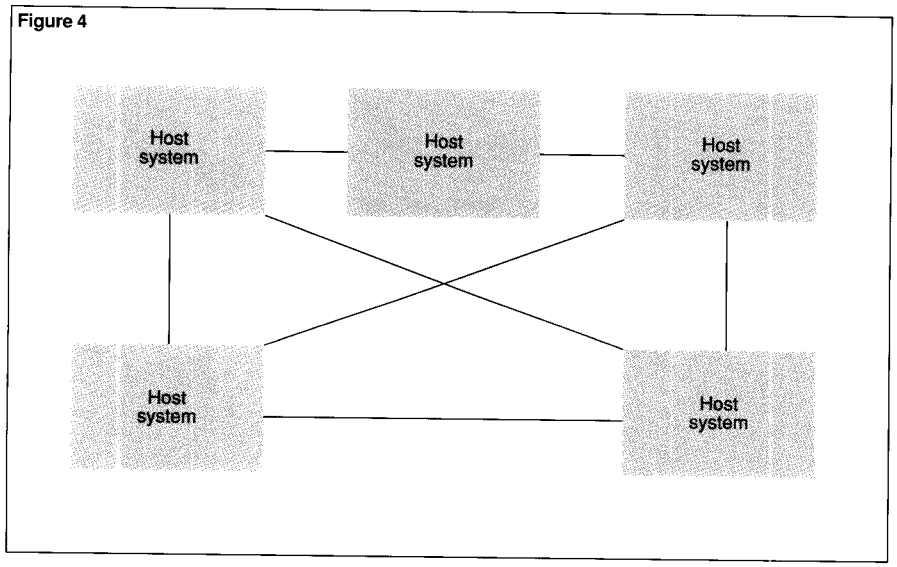


Figure 4.
A distributed network.

Several levels of concentration are theoretically possible. However, the network designer usually limits the number of levels of concentration to avoid two main problems: the total delay can become too large, and the number of devices affected in a failure can reach an unacceptable level.

Distributed Topology. In a network designed to use a distributed topology, as illustrated in Figure 4, the processor systems deal with each other on a peer-to-peer level. The systems send data to and receive data from the other systems. This topology provides the capability for distributed processing and distributed databases and allows decentralized control by each system of the associated devices connected to it. The intelligence and processing capabilities can be distributed throughout the geographical scope of the network. Resource sharing is possible, and some functions may be localized to specific systems.

From the network user's point of view, the ideal situation is to have a fully connected network with each system directly connected to every other system. However, this is generally cost-prohibitive and unnecessary for most networks. Instead, many designs require that each system is directly connected to only two other systems. The access to other systems is then through the directly connected systems. The number of minimum connections is generally based on these criteria: the need for alternative paths if primary paths fail, cost, and network delay constraints. The amount of delay incurred as a result of going through intermediate systems as well as the overall cost must be assessed when making decisions on system connections.

Three types of networks—Tandem EXPAND™, multidomain systems network architecture (SNA), and open systems interconnection (OSI) networks—provide examples of systems using a distributed topology design. There are additional delays in response times if terminals communicate with systems to which they are not physically connected. The amount of delay depends on the number of systems and lines that the data must traverse.

The design problem for this type of topology involves determining the most effective routing for traffic from one node to another within cost and performance constraints. This includes choosing nodal sites, assigning routing, and configuring connecting lines and line capacities. The greater the number of network nodes, the more complicated is the determination of internode connections. The effect of failed primary lines needs to be examined so that adequate capacity is available on alternative routes when they are needed.

The number of intermediate systems between source and destination can have significant cost and performance considerations for a distributed topology. On one hand, there may be substantial cost savings from routing data through several systems; however, delays encountered during periods of heavy traffic may become quite substantial.

The locations of databases also have implications for distributed network designs. Accesses to remote databases are network transactions. More centralized databases create a higher volume of internode traffic in the network. This may be especially significant if large amounts of data need to be transferred frequently from one system to another.

Network Components

Various communications facilities can be used to connect the devices and systems and can have an effect on the network topology. For example, the use of satellite technology may make a point-to-point network topology the best choice, especially when there are locations in remote areas. On the other hand, the use of terrestrial lines to connect a network covering a large geographical area may make a multipoint configuration the most cost-effective solution.

The selection of the communications media can also be affected by the type of information being sent across the network. For example, if voice, message, data, and facsimile are to be integrated into one network, very high-capacity lines may be required. If voice circuits are not used after business hours, they can be made available for data traffic (such as file transfer) during these hours; this reduces the total number of circuits needed without degrading the service quality. Often, individual requirements for voice, message, or data lines at a given site may not justify a leased circuit, but a combination of two or more types of requirements might.

Local devices, terminals, and other equipment at the computer system site can be connected directly to the system or indirectly connected by way of a local area network. Remote terminals and concentration devices can be connected by various methods, including dial-up and leased lines, satellites, public packet data networks, fiber-optic lines, and microwave technology. Depending on their features, these methods fall under the general categories of switched or private lines.

Switched Lines. If the charges for private lines are very high and there is a low traffic volume, a measured service that bases its charges primarily on usage may be the best choice. Costs are only incurred when the lines are actually in use. Switched lines such as the standard dial-up telephone network and public packet data networks can be an effective way to connect points with a modest volume of transactions. They provide for low fixed cost, short-term commitment, and accurate network accounting. The added advantage is that virtually any point on a world network can generally be reached by a dial-up or a public packet data network.

Disadvantages in the use of dial-up telephone lines include the variable line quality, low speeds, and long call setup times. The quality and speed of dial-up lines may make it impossible to effectively transfer large amounts of data; varying line quality generally restricts dial-up lines to speeds no greater than 9600 bits per second. The lines often vary widely in error rates, which may result in many retries when attempting to transmit large amounts of data. Response time requirements can also rule out dial-up lines because of the amount of time involved. Depending on the technology involved in the dialing and switching of the call, the time to make a connection can take as long as 25 seconds.

Private Lines. Private, dedicated lines are valuable for networks that have a high volume of transactions from a relatively small number of highly localized points and have more stringent response-time requirements. A variety of technologies can be used to provide long-distance lines, such as satellite, microwave, and other high-capacity services. Where large amounts of data can be concentrated, especially if voice and data are integrated, the high-speed carrier services can play an important part in the network.

Figure 5.
Estimating input and output message service time.

Figure 5

(a) Example: Calculating input message service time.

Assumptions:
Half-duplex input sequence:
Host polls the device.
Device responds with its input message.

Values for this example:
PrtO = 16 characters
PrpD = two @ 0.02 second each
MD = one @ 0.025 second
ML = 60 characters
LS = 4800 bits per second

$$\left(\frac{(60 + 16) \times 8}{4800}\right) + (2 \times 0.02) + 0.025 = 0.192 \text{ second}$$

(b) Example: Calculating output message service time.

Assumptions:
Half-duplex output sequence:
Select message from the host to the device.
Acknowledgment from the device.
Output message from the host.
Acknowledgment from the device.

Values for this example:
PrtO = 32 characters
PrpD = four @ 0.02 second each
MD = one @ 0.025 second
ML = 200 characters
LS = 4800 bits per second

$$\left(\frac{(200 + 32) \times 8}{4800}\right) + (4 \times 0.02) + 0.025 = 0.492 \text{ second}$$

The clear advantage to this type of service is the assured, immediate, and generally more reliable connection. Among the disadvantages are the high cost and the accompanying long-term commitment. Altering and re-engineering private lines is expensive and time-consuming, thus hindering a network's ability to respond quickly to usage changes.

Satellite technology presents several unique considerations. Potentially long propagation delays may exist when sending data to the satellite then to the destination; this may preclude satellite use in some applications. A data link protocol that provides for a large number of unacknowledged frames to be outstanding must be employed to make effective use of satellite links. Weather can be an important factor for both satellite and microwave installations. However, some satellite implementations can be quite cost-effective and satisfactory, especially when remote rural locations are involved.

When dealing with private lines supplied by the various communications carriers, the designer often has a choice between analog and digital lines. However, some line speeds may only be available as digital offerings. Digital service is generally considered to be more reliable than analog, but it is also more expensive and not universally available.

Selecting Line Capacities

When selecting the speed, or capacity, of the lines to connect the various components, the designer must keep the performance requirements for the line clearly in mind. For most situations, this equates to meeting the response-time requirements for applications on the network. Where there are several lines involved in the connection, individual line delays must be calculated.

Components of Response Time

Response time consists of the following components:

- Queueing delay waiting to use the input line.
- Polling delay, if polling is involved.
- Input message service time.
- Host processing time.
- Queueing delay waiting to use the output line.
- Output message service time.
- Other hardware delays from equipment such as multiplexers and concentrators.

The service times, polling delay, and queueing delays can be calculated using basic queueing theory formulas. The formulas described below are methods for providing quick approximations of the response times and other delays. More rigorous and precise methods are also available.[1]

[1]See Martin, 1972, and Kleinrock, 1976, for detailed descriptions and discussions of queueing theory techniques.

Message Service Time. Message service time is the length of time it takes to transmit a message block across a single communications line. This value can be calculated by using the formula:

$$MST = \frac{ML + PrtO}{LS} + MD + PrpD$$

where

MST = message service time
ML = message length
PrtO = protocol overhead
MD = modem delay
PrpD = propagation delay
LS = line speed

Figure 5 presents examples that illustrate calculations of input and output message service times.

The formula for calculating message service time assumes that the line is released as soon as the message is received by the next hardware component. If this is not the case, the additional line holding times must be included as part of the service time. Also, if a message is divided into more than one block when being transmitted across the communications line, this calculation should be done for each block.

Modem delay depends on the type of modem. It commonly ranges from 25 to 50 milliseconds for speeds up to and including 9600 bits per second. Propagation delay for nonsatellite lines is commonly estimated at 1 millisecond per 100 miles. For satellite circuits, the propagation delay must include the time to go up to the satellite then down to the destination, so the delay is longer. The estimate is about 250 to 300 milliseconds.

Polling Delay. The average polling delay is the average length of time a device can expect to wait to be polled. This depends on the number of pollable points on the line. As illustrated in Figure 6, this value can be approximated by using this formula:

$$PolD = \frac{n-1}{2} \times NPT$$

where

PolD = polling delay
n = number of polling points
NPT = negative poll time

Figure 6

Values for this example:
n = 9
PolO = 16 characters
PrpD = two @ 0.02 second each
MD = one @ 0.025 second
LS = 4800 bits per second

$$\frac{(9-1)}{2} \times \left(\frac{16 \times 8}{4800} + (2 \times 0.02) + 0.025\right) = 0.367 \text{ second}$$

Negative poll time is the amount of time, including propagation and modem delays, to poll one point without obtaining data. Figure 6 illustrates how the calculation for this value is embedded in the calculation for the average polling delay. The formula for negative poll time has these elements:

$$NPT = \frac{PolO}{LS} + PrpD + MD$$

where

PolO = number of characters included in the polling message and the negative reply

Figure 6.
Calculating polling delay.

Figure 7.
Calculating queueing delay and line utilization.

Figure 7

Values for these examples:
Input MST = 0.192 second
Output MST = 0.492 second
n = 3500 messages per hour

$$\text{Input LU} = \frac{3500}{3600} \times 0.192 = 0.187\ (18.7\%)$$

$$\text{Input QD} = \frac{0.192 \times 0.187}{(1-0.187)} = 0.044 \text{ second}$$

$$\text{Output LU} = \frac{3500}{3600} \times 0.492 = 0.478\ (47.8\%)$$

$$\text{Output QD} = \frac{0.492 \times 0.478}{(1-0.478)} = 0.451 \text{ second}$$

Figure 8.
Formula and example for calculating response time.

Figure 8

Formula.

$$RT = \underset{QD}{Input} + PolD + \underset{MST}{Input} + HPT + \underset{QD}{Output} + \underset{MST}{Output}$$

where

RT = response time
HPT = host processing time

Values for this example:
HPT = 1.5 seconds
Other values used in this example are gathered from the calculations displayed in Figures 5, 6, and 7.

Example.

0.044 + 0.367 + 0.192 + 1.5 + 0.451 + 0.492 = 3.046

Queueing Delay. Queueing delay is defined as the time incurred waiting for the communication line to be available for use. Queueing delay must be calculated for both input and output services. As presented in Figure 7, queueing delay can be estimated by using this formula:

$$QD = \frac{MST \times LU}{1 - LU}$$

where

QD = queueing delay
MST = average message service time (in seconds)
LU = line utilization

The line utilization value is calculated as follows:

$$LU = n \times MST$$

where

n = number of messages per second

The formula for estimating the queueing delay assumes that the interarrival times for the messages follow a Poisson distribution, and the service times for the messages follow an exponential distribution.[2] While these assumptions are not strictly true for some situations, the use of this formula should result in satisfactory approximations to the queueing delays for most cases. When these assumptions are not met or there is a need for more precise results, other methods should be used.[3]

Host Processing Time. In addition to the values for the line delays, the value for the host processing time is a necessary component of the response time. Host processing time represents the time the host takes to process a message, from the time it is received at the host until the response is sent out on the line.[4]

Calculating Response Time

Once all the required calculations have been made, estimating the response time is normally a simple matter of adding together all the values for the components discussed above. (See Figure 8.) However, there are special cases that demand the inclusion of additional values.

When calculating response times for data messages over switched, dial-up lines, the call establishment time must be included as part of the total response time. This can be as much as 25 seconds, depending upon the technology involved. Similarly, if a public packet data network is being used, the connect time has to be included.

The designer must take into consideration the end-to-end flows and delays of the logical connection. This is especially true in a network where hierarchical and distributed elements exist in the network and where a terminal user, physically connected to one system, may be accessing data at another system. The path from the terminal to the destination system can cross several lines and systems. When calculating response time for a terminal line that connects to a remote system, all the various delays and processing times that are encountered from source to destination and return must be included.

[2]Martin, 1972, and Kleinrock, 1976, provide explanations of Poisson and exponential distribution.

[3]See Martin, 1972, and Kleinrock, 1976.

[4]See Horwitz, 1988, for a discussion on host response-time calculation.

Effect of Line Utilization on Queueing Delays

As noted previously in Figure 7, message service time and line utilization determine the amount of queueing delay waiting to use a line. In order to provide satisfactory response times for the network users, service times and line utilizations must be kept at acceptable levels. Service time is affected by the chosen line speed, and the line utilization by the amount of traffic on the line. A higher line speed or reduced line utilization results in shorter queueing delays. For example, a line that runs at 2400 bits per second should generally not have as high a utilization as a line that is running at 56,000 bits per second.

The effect of line utilization (LU) on line queueing delays can be quickly approximated by the following formula:

$$\frac{1}{(1 - LU)}$$

This formula calculates a value that, when multiplied by the average service time, gives a quick approximation of the average time to service a transaction. This estimate includes the average queueing delay before the transaction is serviced.

For example, a line utilization of 75% results in a multiplier of 4. This means that the total time (message service time plus queueing delay) is equal to four times the service time. Similarly, for line utilizations of 40% and 85%, the multipliers are 1.67 and 6.67, respectively. Figure 9 presents a graph illustrating the calculated values of this multiplier for line utilizations up to 95%.

For very high-capacity point-to-point lines, high line utilizations may be possible without adversely affecting response times or intersystem delays. On the other hand, restricting low-speed lines to utilizations that do not exceed 35% to 40% may be necessary to achieve satisfactory response times.

The network designer must also consider the effect of peak traffic flows. The queueing delay increases very rapidly after a line utilization of about 60% is reached, and this can have a disastrous effect on the network if high line utilizations are encountered.

Figure 9

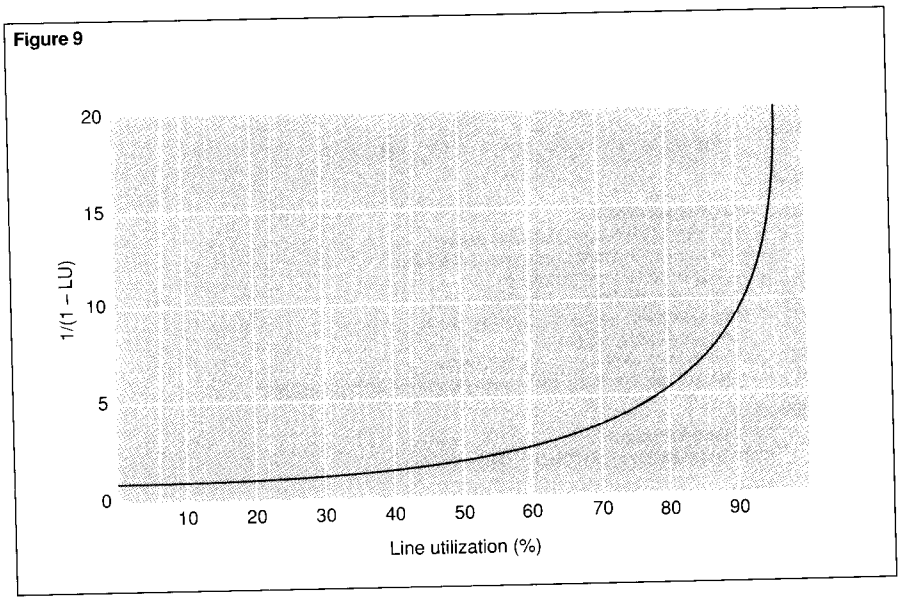


Figure 9.
Effect of line utilization on message service time and queueing delays.

For dial-up or switched lines, the design calculations essentially involve determining how many lines are needed to meet certain service considerations as well as determining response times. These considerations are usually stated in terms of what percentage of the time all circuits are busy such that a call connection cannot be completed. Typical percentages are 1% and 2%, depending on the type of circuit and the requirements of the users of the line.

The calculation involves determining the amount of time an average call occupies a circuit, then multiplying the time value by the number of calls in a "busy" hour. Depending on whether the call is retried, lost, or queued, the appropriate formula or method can then be used.[5] In most cases, this is done through the use of traffic engineering tables, developed to determine the number of circuits that are necessary to meet a specified grade of service.

[5]Held, 1983, and Freeman, 1980, provide more details on the calculation involved in designs with switched lines.

Figure 10

Formula.

$$ELS = \frac{IB \times IC \times (1 - PE)}{((TNB \times (IC + OC)) / LS + TBS) \times (1 - PE + (BR \times PE))}$$

where

ELS = effective line speed
IB = number of information bits in a character
IC = number of information characters in a block
PE = probability of an error in a block
TNB = total number of bits in a character
OC = number of overhead characters in a block
LS = line speed (bits per second)
TBS = time between sending successive blocks
BR = number of blocks that must be retransmitted if an error occurs

TBS values:
for full-duplex lines = 0
for half-duplex lines = 2 x (PrpD + MD + ACTT)

where

ACTT = acknowledgment characters transmission time

The values for PE and BR are calculated separately:

$$PE = 1 - (1 - ER)^{((OC + IC) \times TNB)}$$

where

ER = error rate; typically, 0.00001 for analog and 0.000001 for digital lines

$$BR = 2 + \frac{(2 \times PrpD) \times LS + CRM}{(OC + IC)}$$

where

LS = line speed in characters per second
CRM = number of characters in reject message

Example.

Values for this example:
IB = 8 bits
IC = 238 characters
OC = 26 characters
TNB = 8 bits
LS = 9600 bps (1200 characters per second)
TBS = 0 (full-duplex line)
ER = 0.00001
CRM = 6 characters

1. Compute the value of BR:

$$2 + \frac{(2 \times .02) \times 1200 + 6}{238 + 26} = 2.2$$

2. Round up the value 2.2 to 3, the next whole number.

3. Compute the value of PE:

$$1 - (1 - 0.00001)^{((26 + 238) \times 8)} = 0.0209$$

4. Compute the value for ELS:

$$\frac{8 \times 238 \times (1 - 0.0209)}{((8 \times (238 + 26)) / 9600) \times (1 - 0.0209 + (3 \times 0.0209))} = 8134 \text{ bps}$$

Taking this value for ELS and assuming there are 7 million characters of data to be sent, the predicted time to transfer the data is as follows:

$$\frac{7{,}000{,}000 \times 8}{8134 \text{ bps}} = 6885 \text{ seconds (1.9 hours)}$$

Figure 10.
Calculating an approximation of the effective line speed.

Transmission of Large Volumes of Data

For batch or file transmissions, the concerns are generally how long it takes to transmit the data and whether or not this data can be transmitted in the specified "window" of time. For example, there may be the requirement for remote locations to transmit the day's activities after hours within a time period of six to eight hours.

Determining the Appropriate Line Speed. For networks that expect to be transmitting large amounts of data, the designer must determine a line speed that can adequately transfer the data within the specified time period. An approximation of the effective line speed can be calculated by using the formula developed by Zielke;[6] see Figure 10.

The calculation presented in Figure 10 assumes that the line is being used exclusively by the batch or file transmission such that there is no queueing for the line. If the line is to be shared concurrently between this type of transmission and interactive transactions, it may be necessary to treat the file transmission as interactive for the purposes of line utilization analysis. That is:

- Take the number of hours in the projected file transmission window.
- Determine the number of blocks to be transmitted.
- Divide this value by the number of hours.

The resulting value represents the hourly rate of message blocks that can used along with the interactive hourly rate. This calculation should give approximate line utilization for both, although in practice the file transmission may actually monopolize the line.

Communication Line Error Rate. The designer must pay close attention to the error rate expected from the communications line. If the error rate is high and the blocks of data being sent are quite large, substantial amounts of retransmission time may occur on the lines. For example, if the error rate is 1 bit per 100,000 bits and the block size is greater than 100,000 bits, an error can be expected to occur in every block on the average. In the extreme case, the file might never be successfully transmitted because every retry would also have the same expectation of an error.

[6]See References, which lists the course notes (Zielke, 1986) that discuss this formula.

As errors on communications lines tend to come in bursts, the situation is not generally that bad. Nonetheless, the designer must place importance on considering the block size and the error rate when large amounts of data are expected to be transmitted. Reducing the block size or using a communication medium or protocol that results in a better error rate may be necessary.

Packet Switching Delay

Individual line delays in a packet-switched network, such as an EXPAND network, can be calculated treating each line as a point-to-point line (described previously). Alternatively, the estimate of the average approximate delay for a complete message can be calculated using the formula described below. Each message is divided into one or more packets.

$$\text{MsgD} = \frac{(\text{PS}) \times (n + 1) + (\text{MS} - \text{PS})}{\text{LS}}$$

where

MsgD = message delay
PS = packet size
n = the number of switches
MS = message size
LS = line speed

An example of this formula is presented in Figure 11. This method assumes that all lines have the same speeds. Some additional time may be required if there are some slower lines. Processing delays incurred at the switches are not included. The example in Figure 11 also assumes no significant queueing delays on the lines. More precise methods for calculating this type of delay are also available.[7]

Communications Carriers

Several potential communications carriers are available for use, depending upon the technology and the geographic area to be covered by the network. However, in some countries, the selection of dial-up and private lines may be limited to the government-run telephone companies. Other choices may exist if the group decides to use a public packet data network.

Figure 11

Values for this example:

PS = 238 characters (1904 bits)
n = 5
MS = 1552 characters (12,416 bits)
LS = 9600 bps

$$\frac{1904 \times (5 + 1) + (12{,}416 - 1904)}{9600} = 2.285 \text{ seconds}$$

Figure 11.
Estimating the average approximate delay for a complete message in a packet-switching network.

The divestiture of American Telephone & Telegraph and the consequent restructuring of communications line tariffs has made the pricing of large networks in the United States more complicated than it was prior to the divestiture. Now the network designer must be aware of many more tariffs if costs are to be kept at a minimum.

The tariff structure may also affect the network design. Connecting the components in ways that take advantage of the tariff pricing may be more cost-effective. For example, in the United States, the distinction between inter-LATA (Local Access and Transport Area) and intra-LATA pricing may make it cost-effective to try to connect as much as possible within the geographical boundaries of a LATA and reduce the number of inter-LATA connections.[8] The pricing structures in other countries might also have similar implications for network designs; an example might be the concentration of the traffic lines within a country to reduce the network's number of international lines.

[7] See Kleinrock, 1976, and Tanenbaum, 1981.

[8] LATAs are geographical areas defined as a result of the divestiture and restructuring of AT&T and the Bell operating companies. The Bell operating companies and local independent telephone companies offer regulated telecommunications services within LATAs (intra-LATA), while AT&T and other long-distance carriers offer services between LATAs (inter-LATA).

Methods and Tools

The task of network design, if it is to be done properly, can involve considerable time and effort. The larger the network and number of locations, the more numerous the possible combinations become as the designer evaluates the many design alternatives.

To help the designer evaluate the different design options, three general methods are available:

- Manual calculations.
- Computerized analysis and modeling.
- Computer simulation.

Each of these methods has its benefits and attendant costs. The designer decides how accurate the prediction or analysis should be, then balances the design costs against the benefits of improved accuracy and precision. Selecting the tool that is most appropriate for the design effort is very important. For example, computer network design packages are available for both hierarchical and distributed network designs. While a designer can use a hierarchical design package to do much of the performance analysis of a distributed network, it is not usually possible for such a package to design the connections within the network.

Selecting the right tool for the design analysis is essential.

Manual Calculations

Two general types of manual calculations can be used in the network design process. One type, performance analysis and prediction, uses statistics and queueing theory to determine line utilizations, queueing delays, and response times. The other, network connectivity, focuses on determining optimum connectivity within the network.

Performance Analysis and Prediction. The performance analysis and prediction can be done using various queueing theory calculations and operations research methodology. Where switched lines are involved, the designer may need to calculate the number of circuits or ports to be used. This type of calculation tends to focus on the total use and availability of the circuit, and it is used to determine the number of circuits or ports needed to provide a specified grade of service.

For file and batch transmissions, the focus may be on the effective data rate of the line and the amount of time it takes to transmit the data over the line. Often, there is a restricted window of time within which the data must be transmitted.

Many of these calculations can be done by people who do not have extensive math backgrounds. A designer can use manual methods to calculate quick approximations of response times, queueing delays, service times, polling delays, packet delays, and effective line speed. The calculations described previously provide examples of the variety of manual calculations that can be done. These results will not be as precise and accurate as those achieved through the use of simulation or a more sophisticated analytical model, but they can give reasonable and useful approximations.

Network Connectivity. The tasks involved in determining network connectivity are a little more difficult and time-consuming to do manually. For example, if there are n locations in the network, there are $n\ (n-1)/2$ potential lines because each line can be present or absent, and the number of potential topologies is $2^{n\,(n-1)/2}$.

To make this task more manageable, a number of heuristic algorithms have been developed that can be used to produce nearly optimal networks. The use of the algorithms speed up the design process by making some well-chosen approximations.[9]

Doing a complete design by hand is theoretically possible, but because of the many possible combinations, particularly with large networks, this is generally not a practical solution. If a design does present numerous alternatives or it requires the evaluation of various protocols, tariffs, and configurations, it is often better and more practical to use computer-based models.

Computerized Analysis and Modeling

Programs that offer computerized analysis and modeling are generally computerized versions of the queueing theory calculations and include algorithms that have been developed to expedite and simplify the design process. Several network design packages have been developed to help the designer quickly evaluate many alternatives, such as the variety of hardware, software, tariffs, protocols, and facilities related to many design possibilities, and to manage the complexity of the design process.

Most of the available tools provide assistance in network topology design layout and connectivity as well as response-time analysis. In addition, many packages provide some or all of the following:

- Support for various tariffs.
- User-modifiable parameters.
- Various design run criteria.
- Network database, containing network characteristics and designs.
- Various reports that describe network topology.
- Detailed and summary design costs.
- Support for multiplexer, concentrator connectivity and siting, or both.
- Support for user-defined tariffs and protocols.

There are limitations, however, in the use of these packages. They generally support tariffs for only a limited number of countries, many do not provide adequate support for the integration of voice and data, and several iterations are sometimes necessary for more complex network designs.[10]

Most of these packages essentially only model the application messages and the link level protocols. Therefore, when modeling layered protocols such as EXPAND, OSI , and SNA , the designer must be aware of the additional line utilization possibly incurred by upper layer message overheads. In addition, if control messages are being transmitted, significant, additional line utilization can result.

[9]For a discussion of heuristic algorithms, see Tanenbaum, 1981; Kleinrock, 1976; and Ellis, 1986. For a discussion of a number of techniques developed to analyze and predict performance, see Fernandez, 1988; Stern, 1985; Johnson, 1985; Wood, 1985; and Sharma, 1988.

[10]For a discussion of available network design packages, see Van Norman, 1988, and Green, 1986.

Many factors should be considered before selecting a computerized network design package. A brief list includes:

- Ease of use.
- Amount of training required to learn the package.
- Interactive or batch mode operation.
- The type of system required to run the package.
- Design models supported.
- Readability of output reports.
- Vendor maintenance and consulting support.

The designer must also keep in mind that different tools are required for different topologies. The hierarchical design problem is not quite the same as a distributed peer-to-peer packet switch network design. It may be possible to model parts of a peer-to-peer network with a program designed for hierarchical networks, or the reverse. However, the best approach is to use the appropriate tool for the job.

Computer Simulation

The designer uses this method to attempt a computer simulation of the events that take place in the network. The use of simulation provides for more accuracy and flexibility, but it requires detailed knowledge of the working of the system being modeled. The designer must supply very thorough details of the protocols, software, and processing events of the network for simulation to be an effective approach.

Simulation can produce very accurate results in timing individual events, such as message transit time. Execution, though, can take many hours of processing time, especially if several alternatives are examined. Simulations can overcome problems and inaccuracies caused by simplified assumptions that can occur in the use of analytic models. However, the cost in resources—computer time and time taken to gather and specify the required data—can be quite significant.

Nonetheless, some problems cannot be effectively solved without the use of simulation. These include problems where the assumptions necessary to use queueing theory do not hold or where more precise results are needed.

Some commercially produced packages and simulation languages are available that can be used to simulate networks. These programs are generally limited to the analysis of existing designs rather than the creation of new designs. Some are packages that provide relatively easy methods to analyze a proposed network design; others are simulation languages that can be used to define simulation models. Because of the high cost in terms of computer and human resources to run extensive simulations, the designer must examine the trade-offs between the accuracy needed and the cost of the simulation run.

Documenting the Final Network Design

Once the final design has been selected, the results of the network planning and design should be well documented. Such a report should be a complete statement describing the network and its components. This should present a detailed description and diagram of the network as well as a narrative of the network functions, which include:

- The number and types of lines.
- The placement of concentrators.
- The number of terminals required.
- The number of points on multipoint lines.
- The number of dial-up lines or computer ports.

Cost, performance, and reliability capabilities should all be detailed, and special interface requirements or areas that provide potential risk to the network should be explained thoroughly. Plans for the monitoring and control of the network should also be included. Finally, any assumptions that were made during the design process should be recorded.

Conclusion

Data networks in most companies today are integral parts of the functioning of the organization. A bad network design can have serious, detrimental effects on the operation and profitability of the organization. Conversely, a well-designed network can facilitate the functioning of an organization and provide many benefits such as increased productivity and timely access to information. The significant advantages of a well-designed data network underscore the need for careful and effective network designs that meet the current and future needs of network users.

Many considerations are involved in the network design process. Because the potential configurations and alternatives for designing networks can be so numerous, successful network designers must carefully evaluate the various alternatives based on cost, performance, and reliability. Some calculations have been presented that can be used to provide quick approximations by hand, if desired; however, these calculations only provide reasonable approximations. If more precision is required, the use of more sophisticated formulas or perhaps a computerized network design package is recommended.

References

Ellis, R. 1986. *Designing Data Networks*. Prentice Hall.

Fernandez, J., and Liddy D. March 1988. Don't Just Guess: How to Figure Delays in Private Packet Networks. *Data Communications*.

Fernandez, J., and Liddy D. May 1988. Is Your Packet Network a 'Model' of Response-Time Efficiency? *Data Communications*.

Freeman, R. 1980. *Telecommunications System Engineering*. John Wiley and Sons.

Green, J. April 1986. Microcomputer Programs Can Be Adapted for Data Network Design. *Data Communications*.

Held, G. October 1983. No More Guesswork for Sizing Network Components. *Data Communications*.

Horwitz, H. 1988. Estimating Host Response Time in a Tandem System. *Tandem Systems Review*. Vol. 4, No. 3. Tandem Computers Incorporated. Part no. 15748.

Johnson, J. February 1985. Universal Flow and Capacity Index Gives Picture of Network Efficiency. *Data Communications*.

Kleinrock, L. 1976. *Queueing Systems Volume II*. John Wiley & Sons.

Martin, J. 1972. *Systems Analysis for Data Transmission*. Prentice Hall.

Sharma, R. September 1988. T1 Network Design and Planning Made Easier. *Data Communications*.

Stern, D. October 1985. A Quick Model for Getting Network Response Time Close to Perfect. *Data Communications*.

Tanenbaum, A. 1981. *Computer Networks*. Prentice Hall.

Van Norman, H. April 1988. A User's Guide to Network Design Tools. *Data Communications*.

Wood, J., and Wood D. July 1985. Computing Costs—The Important Bit Per Second. *Data Communications*.

Zielke, G. 1986. *Data Communications Network Design Course Notes*. Institute for Advanced Technology.

Jerry Evjen joined Tandem in April 1984. He is presently part of the Product Support and Introduction Group within the Customer Support Organization. Previous assignments within Tandem have included course development in Software Education and field analyst support. Experience prior to Tandem includes working for computer and communications vendors as well as end users.

NonStop SQL: The Single Database Solution

Tandem conducted a series of performance evaluation benchmark tests for the California Department of Motor Vehicles (DMV) in November 1988. NonStop® VLX™ systems using NonStop SQL[1] demonstrated high-performance online transaction processing (OLTP) and batch processing on a database of nearly 60 Gbytes.

The DMV benchmark utilized the database management system (DBMS) capabilities of the Tandem® NonStop SQL product, as well as the high availability, fault tolerance, and linear expandability of Tandem VLX hardware. The Tandem hardware and the DBMS architecture showed the performance, availability, and ease-of-use characteristics required by the complex processing environment of the California DMV.

This article describes the performance aspects of the DMV benchmark—the configuration approach, the application, system tuning, and the required tests.

Reviewing Database Management Needs of the DMV

The California Department of Motor Vehicles is the largest U.S. motor vehicles department.[2] The DMV maintains information on more than 30 million vehicles and over 20 million driver's licenses, making it the state's largest database on California residents. This data must be available to law enforcement officials at all times throughout the state. For example, a change made in Los Angeles, California, must be available to law enforcement officials hundreds of miles away in San Francisco.

The DMV currently operates a highly customized data processing environment. Although this system is effective for current needs, increasing demands anticipated for the system required the DMV to reexamine the management of its very large database. Therefore, the DMV, with the help of a nationwide management consulting firm, developed a data processing approach for the next ten years. When implemented, this approach will take the DMV from an IBM-based MVS Assembler/VSAM environment to a COBOL/relational database management system environment. The new system is designed to eventually handle the Vehicle Registration and Driver Licensing functions.

[1]This benchmark used NonStop SQL, release 1.

[2]See *Shared Logic*, Autumn 1988.

The Operational Assessment

The DMV evaluated potential vendors against a common set of technical, business, and cost considerations in what they termed an operational assessment.[3] The technical considerations included:

- Continuous availability.
- Performance.
- Operational ease-of-use.
- Application development.
- Infrastructure software.
- Future DMV requirements.
- Training.
- Connectivity.

The DMV assessed the first three areas—continuous availability, performance, and operational ease-of-use—during a hands-on benchmark of the proposed system. Tandem addressed the remaining criteria through a written response.

To demonstrate availability and performance, the benchmark used a realistic representation of the key DMV processing workloads. The benchmark tested each vendor's ability to complete these tasks:

- Implement a very large database (60 Gbytes).
- Process complex online transactions and batch jobs while maintaining 30 transactions per second (tps) at a 1.5-second response time at the 95th percentile.
- Operate 24 hours a day, 7 days a week.
- Demonstrate central processing unit (CPU), disk, and controller failure without significant interruption of system availability.
- Perform anticipated system maintenance (such as reorganizing database indexes and tables and adding partitions) with minimum impact to system availability.

The DMV evaluated the operational ease-of-use criteria during the planning and execution of the benchmark. Because the DMV specified a hands-on demonstration of the system, DMV personnel participated actively in every aspect of the benchmark by assuming the various roles of application programmer, database administrator, systems programmer, and operator throughout the entire benchmark. DMV personnel worked closely with individuals from several groups within Tandem. These individuals represented the Western Region, the Customer Support Organization, and Software Development.

Normally, benchmark setup, execution, and measurement are the responsibility of Tandem personnel. In this case, the hands-on DMV specification made it necessary to provide a more sophisticated and automated environment. Thus, during the benchmark testing, TACL™ (Tandem Advanced Command Language) macros automated the execution, measurement, and monitoring functions.

[3] See *Database Development Project—DBMS Operational Assessment*, referenced at conclusion of article, for more information. Copies may be obtained from the California DMV.

Table 1.
Profile of initial vehicle registration database.

SQL table	Row count	Row length	Gbytes	Columns
ACTIVITY*	0	3,860	0.0	3
ANI*	0	94	0.0	21
CITATION	9,185,323	67	0.62	10
CROSSREF	30,512,381	39	1.19	8
LESSEE	886,146	218	0.19	20
OWNER	19,688,314	163	3.21	8
PENDING	2,270,660	616	1.40	75
PWO	236,124	56	0.01	11
RECCOND	28,700,889	105	3.01	21
RESTRAIN	560,034	85	0.05	9
ROLOHIST	22,956,073	221	5.07	20
STOPPROC	13,248,455	115	1.52	23
SUSRDFHO	2,270,660	197	0.45	46
TRANHIST	80,675,951	60	4.84	12
TRANOT	7,581,564	113	0.86	10
VEHICLE	50,000,000	602	30.10	71
Indexes			6	24
Total			**58.52**	

*This shows the initial state of the database. Both the ANI and the ACTIVITY tables were empty at the start of the benchmark, but as logging tables they were added to during the various tests.

The Database

To conduct the benchmark, the DMV supplied a realistic representation of their actual production database. This benchmark database differed from the actual database in that it contained only program-generated data; however, the sizes of the two databases were the same.

The benchmark database consisted of 16 SQL tables for a total of nearly 60 Gbytes. (See Table 1.) An additional 10 Gbytes of files required for system operation (such as GUARDIAN® 90 operating system program files, Transaction Monitoring Facility (TMF™)[4] audit trails, and spooler files) brought the total database size to 70 Gbytes.

The largest table in the database, VEHICLE, contained 50 million rows (30.1 Gbytes). In contrast, a previously run debit-credit banking benchmark (also known as ET1) had four tables for a total of only 3 Gbytes.[5] The largest table in that benchmark had 25 million rows (2.6 Gbytes).

Database Partitioning Approach

The size of the database tables dictated, to some extent, the partitioning strategy. The approach for partitioning these tables (across a reasonable minimum of disk volumes) allowed for each disk volume to have an approximate capacity of 70%. Most tables had 14 partitions, though some had only 12. The largest table, VEHICLE, needed 56 partitions because of its size. Where clustered inserts were anticipated, the tables were configured with empty partitions.

All database tables were partitioned equally across the system. Though the smaller tables could fit on a single mirrored disk volume, partitioning all tables across several volumes prevented any occurrences of disk bottlenecks.

[4]The TMF product provides a number of facilities to maintain data consistency amid concurrent transactions. TMF guarantees database integrity whether the transaction is distributed across network nodes or is local to a single node.

[5]See *Datamation*, April 1985.

Database Load

The DMV provided 326 tapes containing program-generated data to load the database tables. The parallel architecture of Tandem systems allowed all available tape drives to work concurrently on the load. Each drive loaded one partition of a table. With 12 tape drives operating, one 12-partition table could be loaded at a time. This substantially reduced the database load time.

Tandem also used concurrent processing to create the database indexes by dividing the workload among all available CPUs. The multiple processes each worked concurrently on a portion of the database; the index creation process was completed ten times faster than is possible by sequential processing methods.

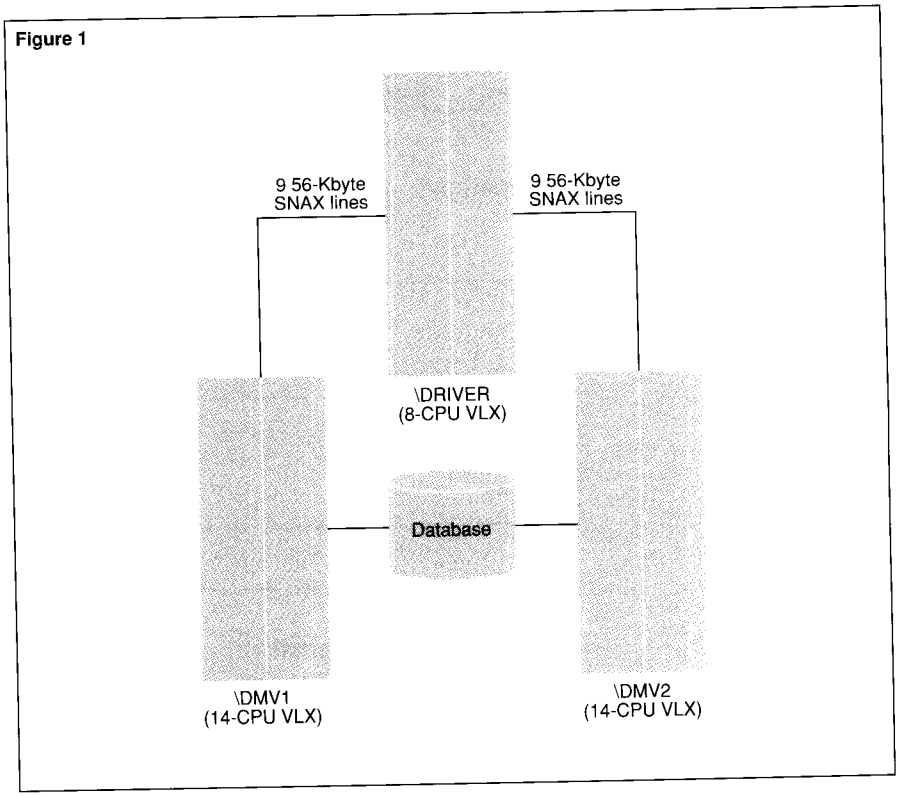
Hardware Configuration

The benchmark system configuration consisted of two components: one driver system (\DRIVER) and a two-node target system (\DMV1 and \DMV2). (See Figure 1.) Because this benchmark was designed to replicate the real-world situation of the DMV, the data communications facilities used for the benchmark represent the actual DMV production network environment.

The actual DMV network involves a message switching system that accepts transactions from remote offices and routes them to the appropriate application. The communications protocol used by this message switching system is Systems Network Architecture (SNA). Therefore, for the benchmark, Tandem provided the Systems Network Architecture Communications Services (SNAX) product as the communications path between the driver and the target nodes.

As specified by the DMV, 18 56-Kbit SNAX lines delivered the business transactions to the OLTP benchmark application. The system configuration provided two physical units (PUs) with 10 logical units (LUs) for each SNAX line. This resulted in 20 LUs per SNAX line and 360 active sessions distributed across the system.

Figure 1



Driver System Configuration

To represent 360 sessions, the configuration of \DRIVER incorporated a separate eight-CPU VLX system. A single tape drive and eight XL8™ disk drives supported the driver system. The driver system delivered transactions to the two-node target system. Transaction driver processes (one per PU) read transaction script files provided by the DMV. The SNAX product, using modem eliminators to connect the communications lines, delivered transactions at a constant rate. The driver system also stored the detailed transaction logs that contained the response-time information.

Figure 1.
Benchmark configuration schematic. The database consists of 27 XL80 disk cabinets (8 XL8 disk drives per cabinet) for a total capacity of 100 Gbytes (mirrored).

Table 2.
Driver and target node configuration summary.

Processors	Storage	Memory	Tape drives
Driver			
8 VLX	8 XL8	8 Mbytes	1
Target nodes (\DMV1 and \DMV2)			
14 VLX	13.5 XL80	CPU0–7: 96 Mbytes CPU8–11: 48 Mbytes CPU12–13: 8 Mbytes	6

Table 3.
Software components of each target node.

System processes	Application
CPU0–7	
GUARDIAN 90XF NonStop SQL SNAX/HLS	Batch
CPU8–13	
	OLTP

Target System Configuration

The target system configuration consisted of two 14-CPU VLX nodes, each with 13.5 fully configured XL80™ disk cabinets for a total of more than 100 Gbytes of mirrored data capacity (200 Gbytes of unmirrored data capacity). Because the actual benchmark requirements were not available when the target system was configured, system designers decided to connect all the various peripheral hardware components to the first eight CPUs in each node.

Restricting peripherals to certain CPUs, rather than balancing the hardware across every available CPU in a system, worked in this case because the disk access rate per disk volume was low. Because Tandem systems allow incremental system growth, CPUs could be added if needed without reconfiguring the entire system.

Memory configuration for both nodes of the target system is shown in Table 2. Each node had approximately 1 Gbyte of random-access memory (RAM) distributed across 14 CPUs. Disk cache was fully configured for each mirrored disk, allowing most table index pages to be kept in memory. This minimized the number of physical disk accesses per transaction and improved performance.

Software Configuration of the Benchmark System

All of the online application software was configured to run in CPUs 8–13 of each target node. (See Table 3.) Because batch processes are I/O intensive and tend to monopolize system resources, the configuration directed the batch programs to be run in the first eight CPUs, in addition to the system processes. This created a throttling mechanism, which lowered the priority of the batch processes and supported a more balanced environment for both batch and OLTP processes. Positive test results indicate this approach is an effective one for NonStop SQL batch processing.

This software configuration was possible because the memory and CPU requirements of the OLTP benchmark application balanced roughly with those of the large number of disk processes in the first eight CPUs. This configuration made the OLTP application's CPU requirement per transaction easily derivable from gross system measurements, simplifying performance sizing.

Benchmark Application

The benchmark application, designed to test both OLTP and batch processing performance, consisted of five online transactions and three batch jobs. The online transactions included:

- C50.
- C55.
- R60.
- R61.
- SNT.

The three batch jobs consisted of:

- Law Search.
- POT Renewal.
- ITT.

The DMV chose to benchmark these particular OLTP transactions and batch jobs because they represent key parts of the Vehicle Registration system. While OLTP and batch processing do not regularly occur simultaneously in the present environment of the DMV, this may become a requirement in the future. This benchmark tested the feasibility of this potential requirement by specifying concurrent batch and OLTP processing.

The benchmark application was written by a group of individuals from DMV; Tandem; a consulting firm; and Expressway, a third-party software vendor. The group used a development tool called Expression.

OLTP Components

The OLTP components running the benchmark application included driver processes, SNAX High Level Support (SNAX/HLS), and requesters and servers of the PATHWAY transaction processing system.

Processes residing on the driver system delivered the OLTP transactions to the two target nodes. (These processes, written by Tandem, were not considered part of the DMV application.) These driver processes simulated the DMV message switching system that routes transactions from remote devices to an application system.

Each driver process reads a transaction from a script file, time stamps it, and writes it to a SNAX/HLS process running on the target system. Using a constant delivery rate algorithm, it then does the same for the next transaction. Each returning response is time stamped, matched with its original transaction, and written to a table in memory. This information is later written to one of several log files on the driver system disks. These log files recorded response times and, during the validation procedures, provided the data for certifying the correct operation of the benchmark application.

On the two target nodes, the OLTP transactions were processed by PATHWAY using Intelligent Device Support (IDS) requesters and COBOL85 servers. The IDS requesters accepted each transaction and routed it based on a transaction code in each message. The IDS requesters also provided the SNAX protocol management through SNAX/HLS. Requesters initiated and terminated a TMF transaction and logged errors to a logging server for three of the OLTP transactions (C50, C55, and SNT). The two read-only OLTP transactions (R60 and R61) did not invoke TMF.

OLTP Transactions

The C50, C55, R60, R61, and SNT OLTP transactions used in this benchmark represent 71% of the volume of online transactions in the current DMV system. As such, these five transactions perform significant business functions and access many of the database tables. Each OLTP transaction is processed by one server class. Table 4 summarizes the SQL statements used by each server class. Except for a join between the VEHICLE and PWO tables in R60, R61, C50, and C55, all the SQL statements in all the servers act upon a single table.

C50 and C55. C50 is a vehicle inquiry/update transaction based on license number. It is used to retrieve information about a vehicle and to set a flag in the VEHICLE table. The flag indicates that further updates will be done to this vehicle at a later time. C55 is used to add information about a vehicle under certain conditions.

R60 and R61. R60 and R61 are virtually identical and are the most common transactions. They retrieve information about a vehicle using either the vehicle license number or the vehicle identification number. The vehicle license number is part of the primary key and is used by all the server classes to access the VEHICLE table. The vehicle identification number is a unique alternate index for the VEHICLE table and is only used by R60 and R61.

Table 4.
OLTP transactions.

	Transactions				
SQL tables	**C50**	**C55**	**R60**	**R61**	**SNT**
ACTIVITY	I	I	I	I	I
CITATION	F		F	F	
CROSSREF					F,I
LESSEE	F	D,I	F	F	
OWNER	S	I	S	S	S
PENDING	S	D,I	S	S	
PWO	S	S	S	S	
RECCOND	F	D,I	F	F	
ROLOHIST			F	F	
STOPPROC	F	F,I,D	F	F	F
SUSRDFHO	S	S,I,D	S	S	
TRANHI	F	I	F	F	
TRANOT			S	S	I
VEHICLE	S,U	S,U,I	S,F	S,F	S,U
# tps out of 30:	**9**	**2**	**3**	**14**	**2**

SQL statement legend:
I = INSERT
F = FETCH, 0 to many rows
D = DELETE
S = SELECT, 0 or 1 row
U = UPDATE

The R60 and R61 transactions are inquiry only, except in the rare case when an exception condition is recorded in the ACTIVITY table. Because writing to the ACTIVITY table (an audited table) is infrequent, the requester does not begin or end TMF transactions for the R60 and R61 server classes. These servers have begin and end logic in place only for the ACTIVITY table. This reduces the number of TMF transactions and improves performance.

SNT. The last transaction type, called SNT, adds or updates a vehicle's information when the vehicle is transferred.

Batch Jobs

In addition to the OLTP transactions, the benchmark application includes three batch jobs: Law Search, POT Renewal, and ITT. Table 5 summarizes the SQL statements used by each of the batch jobs.

Table 5.
Batch process profiles.

SQL tables	Law Search	POT Renewal	ITT
ACTIVITY	I		
ANI			I
CITATION		F	F,I,D
CROSSREF			F,I
LESSEE	F	F	F,I,D
OWNER		S	I
PENDING			S,D
RECCOND	F	F	F,I,D
RESTRAIN			S
ROLOHIST			F,I
STOPPROC	F	F	
SUSRDFHO	S		S
TRANHI			F,I
TRANOT	F	F	D
VEHICLE	F	F	S,U,I

SQL statement legend:
I = INSERT
F = FETCH, 0 to many rows
D = DELETE
S = SELECT, 0 or 1 row
U = UPDATE

Selection Criteria:
Law Search
4 columns from the VEHICLE table (alternate index IVVEHYMM)
1 column from the VEHICLE table (alternate index IVVEHCZC)
POT Renewal
1 column from the VEHICLE table (alternate index IVVEHEDT)

Law Search. This first batch job, Law Search, is actually more like an online query; however, in the current DMV system it takes too long to be executed online. Five parameters describing a vehicle —make, model, year, type, and county code—are used to retrieve information about approximately 2000 vehicles. To collect the pertinent information about each of the 2000 vehicles, six tables (one of which is the VEHICLE table) must be read. Some of the information from these six tables is written to a file for later processing.

Of the five vehicle parameters listed above, four make up one alternate index (IVVEHYMM) to the VEHICLE table. The fifth parameter, county code, relates to a fifth VEHICLE table column that is part of another alternate index (IVVEHCZC) for the VEHICLE table.

POT Renewal. The second batch job, POT (Potential) Renewal, gathers information that is later used to create license plate renewal notices. POT Renewal processes all VEHICLE table rows that have an expiration date falling within a given seven-day span.

A query-only job, POT Renewal uses a selection criteria that retrieves 1/52 of the VEHICLE table (approximately 962,000 rows). The selection criteria is based on an expiration date stored in each row of the VEHICLE table. This date falls within one year of the current date and is another alternate index for the VEHICLE table (IVVEHEDT). In addition to the VEHICLE table, up to six of the other tables may be queried to complete one license plate renewal.

ITT. The last batch job, ITT (Input Transaction Tape), is composed of C10 transactions. (C10 is the name of another DMV transaction currently in use.) C10 transactions are similar in nature to C55 OLTP transactions, but instead of being processed online, the C10 transactions are captured and stored. C10 transactions are periodically copied to tape for later batch-mode processing, which is known as ITT.

The C10 transaction portion of the ITT batch job, the most complex of all the benchmark transactions, may handle three different vehicle registration functions. The type of function performed (add, transfer, or update) depends on the transaction data. In order to minimize TMF transaction commit processing, 150 transactions are handled as one TMF transaction.

Table 6.
Benchmark pass/fail tests and results.

Result	DMV pass criteria
Test 1: OLTP	
30.45 tps	30 tps
95% at 1.3 secs	95% at 1.5 secs
98% at 1.5 secs	
Test 2: OLTP Ramp-up	
See Figure 2	30 tps
	95% at 1.5 secs
Test 3: OLTP Transitional	
29.79 tps	30 tps
95% at 1.65 secs	95% at 1.5 secs
Test 4: OLTP and Law Search	
30.49 tps	30 tps
98% at 1.5 secs	95% at 1.5 secs
29 mins, 31 secs elapsed time	1 hr elapsed time
Test 5: OLTP and POT Renewal	
30.76 tps	30 tps
95% at 1.5 secs	95% at 1.5 secs
8 hrs, 32 mins elapsed time	9 hrs elapsed time
Test 6: OLTP and ITT	
30.52 tps	30 tps
97% at 1.5 secs	95% at 1.5 secs
4 hrs, 49 mins elapsed time	5 hrs elapsed time

Batch Job Processing Approach

Processing these three batch jobs relied on the Tandem architecture to allow splitting the single batch jobs into several concurrently running processes. Each batch process used start-up parameters to determine what part of the database it should act on. Parallel processing proved an important factor for both the OLTP and batch tests; a more detailed discussion is presented in the following section.

Benchmark Tests and Results

The benchmark consisted of these six performance tests:

- Test 1: OLTP.
- Test 2: OLTP Ramp-up.
- Test 3: OLTP Transitional.
- Test 4: OLTP and Law Search.
- Test 5: OLTP and POT Renewal.
- Test 6: OLTP and ITT.

The first three tests focused on meeting the OLTP criteria only. (The OLTP criteria involved running the five OLTP transactions for 30 tps with 95% of the transactions completing within 1.5 seconds.) The last three tests combined the OLTP criteria with the various batch job requirements. Because each benchmark test contained the OLTP criteria, this criteria was viewed as a baseline for all six tests. In order to pass any of the tests, the OLTP portion of each test must pass.

To pass the combined OLTP and batch tests, Tests 4, 5, and 6 required that a balance between processing the five OLTP transactions and the particular batch job had to be achieved. OLTP performance could not be affected by the batch jobs, yet the batch jobs must be completed within specific times.

Table 6 compares the benchmark results with the DMV requirements. The DMV viewed these six tests as pass/fail; all criteria must be met in order to pass a test.

Test 1: OLTP

This test required running the five OLTP transactions (C50, C55, R60, R61, and SNT) at 30 tps for 30 minutes. Ninety-five percent of these transactions must complete from and to a driver process, including SNA communications time, within 1.5 seconds. The tps rate by transaction type is summarized below.

- C50: 9 tps.
- C55: 2 tps.
- R60: 3 tps.
- R61: 14 tps.
- SNT: 2 tps.

The results of this test exceeded the DMV requirements by passing at 30.45 tps, with 95% of the transactions completing in 1.3 seconds; 98% of the transactions completed in 1.5 seconds. To achieve these results, disk balancing was necessary. The goal was to move enough tables from the busy disks to other disks to balance the load. Smaller tables were chosen over larger tables in order to decrease the move time. At the same time, the block size of these tables was changed from 4096 to 512 bytes. This reduced the data transfer time from disk to memory.

Test 2: OLTP Ramp-up

Test 2 required generation of a table of transaction response-time values starting at 30 tps, with 95% of these transactions completing in 1.5 seconds, and finishing at system saturation.

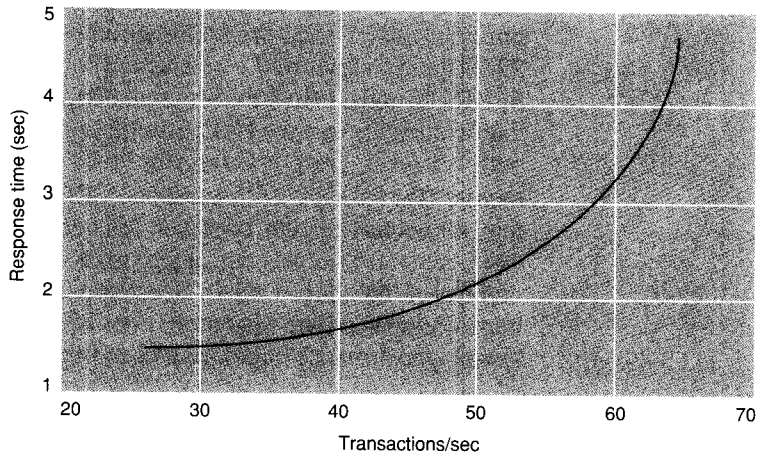
This test used the same OLTP transactions as Test 1. However, the objective was to determine the number of transactions per second that could be achieved before system resources are used up. The test began at approximately 30 tps, with 95% completing in 1.5 seconds. The tps rate was then increased until saturation. Figure 2 shows the results of the OLTP Ramp-up test.

Some additions to the system were made for the OLTP Ramp-up test. Two VLX CPUs with 8 Mbytes of memory were added to each target node. Another 12 SNAX lines were added with 24 driver processes. Terminal control processes (TCPs) and servers were added to handle the increased communication lines and driver processes. These additions (extra CPUs, memory, SNAX lines, TCPs, and servers) were used only for the OLTP Ramp-up test and were not used for any other test.

Test 3: OLTP Transitional

Test 3 required OLTP processing (Test 1 specifications) with all inquiry transactions routed to the target nodes' architecture, while all update transactions were routed to the transitional architecture.

Figure 2



Transactions (per second)	Response time (95th percentile)
26.66	1.5
36.90	1.7
46.65	2.0
61.77	3.5
63.75	4.5

Figure 2.
Ramp-up test results.

This test involved two different architectures: the proposed Tandem relational DBMS system and a transitional system. The objective of this test was to demonstrate a probable interim computing environment where only part of the applications have been converted to the new system. In this case, all inquiry transactions were processed on the Tandem system, while all update transactions were decomposed into calls required by the existing database. Since the database of record was NonStop SQL and resided on the Tandem system, this test demonstrated the ability to translate primitive database calls received from the existing architecture into NonStop SQL calls.

While test results did not meet the DMV pass criteria, it was determined that this was due to the transitional architecture, not the Tandem architecture. The overall measured transaction rate was 29.79 tps with 95% of the transactions completing in 1.65 seconds, below the required 30 tps. The Tandem results met the DMV criteria at 17.14 tps with 1.5-second response time for 96.97% of all inquiry (the R60 and R61) transactions. The existing architecture results, however, did not meet DMV criteria. This was due in part to the limited number of threads (18 SNAX lines) available to transmit transactions. Had more lines been allowed, performance could have been improved.

Test 4: OLTP and Law Search

In addition to meeting the requirements of the OLTP Test 1 (30 tps, 95% at 1.5 seconds), Test 4 required one execution of the Law Search batch job within one hour.

This test was demonstrated twice, showing that the DMV has a choice when running the Law Search job. One demonstration used only one Law Search process. The Law Search completed in 29 minutes and 31 seconds, well within the DMV 1-hour requirement. The OLTP response time was reported at 30.49 tps, with 98% of the transactions completing in 1.5 seconds. In another demonstration, with eight parallel Law Search processes, the Law Search completed rapidly in 7 minutes and 47 seconds. However, the OLTP response time fell to 30.49 tps, with 91% of the transactions completing in 1.5 seconds.

In an emergency situation, priority can be given to the Law Search, while under normal circumstances this batch job can be run more slowly and not impact OLTP response time.

Test 5: OLTP and POT Renewal

Test 5 combined the OLTP Test 1 requirements with completion of the POT Renewal batch job within 9 hours.

The POT Renewal portion of this test ran in 8 hours, 32 minutes, and 57 seconds. Because of the length of the test, two representative 30-minute periods were selected to report the OLTP transaction performance. During the first period, the OLTP transactions ran at 30.26 tps, with 95% completing in 1.5 seconds. During the second period, the tps rate was 30.76, with 95% of the transactions completing in 1.5 seconds.

Successful completion of this test relied on allowing concurrent processing of 14 POT Renewal processes. Each process worked on 1/14 of the VEHICLE table. By specifying to each process which day (out of the seven possible) to handle and restricting it to one-half of the vehicle license numbers, all 14 processes could run at the same time, in different CPUs, without duplication of effort.

Test 6: OLTP and ITT

Test 6 required the OLTP test combined with the ITT batch job processing 60,000 C10 transactions to be completed within 5 hours.

This test proved to be the most challenging to complete. The complexity of the C10 transactions required additional system tuning and disk balancing. To ensure a balanced system, partitions of several tables were moved to underutilized disks. During the move, the block size of one of the tables was changed from 512 bytes back to 4096 bytes to decrease the number of index levels.

This test successfully completed in 4 hours and 50 minutes, with 30.51 tps, 97% of those transactions completing in 1.5 seconds. Parallel processing capabilities were fundamental to meeting the stringent performance requirements of this test. Instead of dividing the script files evenly among the four ITT batch processes, script files were configured such that only one of the three C10 functions appeared in a file. This way, each ITT batch process handled only one type of function; this made performance requirements more predictable.

In addition, the four ITT batch processes paced the C10 transaction processing. Each ITT process was able to determine how many transactions it had to process and how long it had to complete them. It calculated the amount of time to delay between each transaction. This calculation was done after every transaction so the rate of processing could be adjusted based on other factors in the system.

Measurement

Several methods were used during each of the benchmark tests to monitor, capture, and verify performance information. The transaction driver software included a monitoring facility that collected transaction response times by driver process. A terminal displayed this information, which showed the response times for various percentiles, in histogram and tabular format.

Another process collected system-wide tps rates (as opposed to tps rates from individual driver processes). Each driver process sent its tps rate to a collector process on the driver system. Output from the collector process assured the DMV that the correct tps rate was maintained by the driver processes throughout a test.

OLTP Transactions

For the OLTP transactions, the driver processes captured transaction response times, which were defined as the difference between the message SEND and RECEIVE times in the driver processes. This information was written to log files stored on the driver system. At the completion of a test, the log files were combined into one file. The ENFORM™ Query Language/Report Formatter was then used to produce the four reports, summarized below, that were required by the DMV to document test results.

1. The first report documented average transaction response time by transaction type, both at the driver processes and at the server processes.
2. The second report presented a chart of the number of transactions, by transaction, that finished within certain time frames. The time frames began at 1.4 seconds and were incremented by 0.2 second to a maximum of 2.6 seconds. More importantly, this report also showed the tps rate and transaction response time by percentile for all transaction types combined. The DMV used these numbers to document the official results of each test.
3. The third report showed the number of SQL transactions that could not be completed.
4. The fourth report documented the minimum, maximum, and average length of transaction by transaction type.

Batch Job Processes

The three batch job processes—Law Search, POT Renewal, and ITT—periodically wrote the number of processed transactions to log files. These log files were then read to determine the status of the job and its completion time.

Performance Statistics Reporting

During each test, the MEASURE™ System Performance Measurement Tool[6] ran for approximately 30 minutes. The information provided by MEASURE served two functions: to aid in the tuning effort and to verify that all the OLTP transactions were equally distributed between the two target nodes. MEASURE was also used to cross-check tps rate; one message to a server equates to one transaction, so the total of receive rate for all the server processes yields the total OLTP tps rate. In addition, VIEWSYS™ System Resource Monitoring Utility[7] ran during each test to provide online histograms; these provided an immediate, visual verification that the system was balanced.

Validation

In order to ensure the integrity of the benchmark, the DMV contracted with The Codd and Date Consulting Group[8] to assist in the validation procedures. Codd and Date verified that:

- The inter-arrival rates were constant.
- The transactions and batch programs were properly implemented as specified by the DMV.
- The database records were properly loaded.
- A single copy of the database and the transactions were evenly distributed across both nodes.
- The response time was measured correctly.
- The batch programs' execution times were measured correctly.
- Transactions were protected by a dual recovery log.

The DMV also contracted with a second consulting firm to aid in the definition and validation of the benchmark application and tests. The validation procedures addressed several aspects of the benchmark—the operations, the application, the driver program, the transactions, and the measurement.

Prior to running any tests, members of the DMV group reviewed and documented configuration files and command scripts used during the tests. The consulting firm examined server and batch source code to ensure it matched DMV specifications. Each server type was tested individually to verify that it returned the correct responses. The consulting firm also reviewed the source code for the driver program to ensure that response-time measurements were being recorded correctly.

[6]The MEASURE product allows collection and examination of performance data for a range of hardware and software components. In a network environment, MEASURE enables remote measurement and the examination of remote data collection files from a local system.

[7]VIEWSYS is an interactive utility that allows monitoring of system resources while the system is running, making it especially useful as a system-balancing tool. VIEWSYS displays processor usage in terms of percentage used.

[8]Founded by E.F. Codd, the originator of the relational model, and C.J. Date, a leading author and lecturer on relational technology, The Codd and Date Consulting Group is considered the ultimate source for relational product education, evaluation, and consulting.

The origin and completion of transactions were verified by changing a text field in randomly selected transactions. DMV and Codd and Date used a system utility program to change the field content to something recognizable, such as a phrase or an individual's name. The "marked" transactions were tracked throughout the test—during the test by querying the database and after the test by examining the driver system log files. Additionally, a transaction representing each transaction type was selected at random from the driver log files. The DMV used the transaction input to query the database and to compare the contents of the database with the response stored in the log file.

Conclusion

The realistic nature of the database, the data communications facilities, and the application software increased the complexity of all the activities normally associated with the execution of a benchmark. Nonetheless, Codd and Date verified that Tandem exceeded all the requirements specified by the DMV in five of the six tests.

There are several reasons for the success of this benchmark. NonStop SQL and the 28-processor VLX system delivered the performance and functionality required by the technical considerations of the DMV's operational assessment. On the DMV's very large database, the Tandem architecture proved effective in the areas of operability, availability, and performance. The 60-Gbyte database was loaded rapidly on concurrently running tape drives. The parallel architecture allowed a division of the database between processors in the system, increasing the availability of the data and enabling swift index rebuilding or data recovery, if necessary. The benchmark results for Tests 4, 5, and 6 demonstrated that system performance exceeded the requirements specified for the OLTP and the combined OLTP and batch job processing. Ultimately, the high performance of NonStop SQL was a major factor in determining that Tandem provides the best technical database management system solution for the DMV.

Acknowledgments

The authors would like to thank Suzanne Ambiel for her tireless efforts in the preparation of this document. They would also like to list the people who, besides themselves, participated on the Tandem benchmark team: Randy Baker, Christine Bates, Rick Birmingham, Richard Coleman, Al Duran, Pete Gladstein, Ray Glasstone, Jim Gray, Larry Hammond (consultant), John Kunhart, Jeff LaPlante, Marsha Lee, Scott Lerner, Charles Levine, Jeff Marturano, Jack Mauger, Frank O'Donnell, Mitch Pierce (Expressway), Ron Poulin, Terry Schlenz, Pete Schott, Reid Sherwin, Rich Smith, Gary Stevens, Tom Stibrik, Mike VanHeusen, Mark Washeleski, Tom Wilkens, and Dave Wong.

References

Shared Logic. Autumn 1988. Department of Finance, Office of Information Technology. State of California.

Anon, et al. 1985. A Measure of Transaction Processing Power. *Datamation*. Vol. 31, No. 7. A Tandem benchmark based on the ET1 benchmark is documented in Gray, J., Tandem's NonStop SQL Benchmark. *Tandem Systems Review*. Vol. 4, No. 1. Part no. 11078.

Database Redevelopment Project—DBMS Operational Assessment, Volumes 1, 2, and 3. January 1989. State of California Department of Motor Vehicles.

Jim Cassidy is manager of the Application Prototyping Group in Large Systems Marketing Support. He joined Tandem in 1983 as a senior systems analyst in the Tucson office. Jim acted as the LSMS project coordinator for the DMV benchmark.

Terrye Kocher is currently in Large Systems Marketing Support within the Tandem Customer Support Organization. Since joining LSMS in early 1988, she has been working in the areas of database and application development, which led to her participation in the DMV Operational Assessment. Prior to joining CSO, Terrye spent six years with Tandem as a field analyst.

Optimizing Batch Performance

Batch processing performance on Tandem® systems has been enhanced with recent releases of the GUARDIAN® 90 operating system and associated software products. Each successive version of GUARDIAN 90 has improved sequential processing techniques for disk access, memory usage, or file management. In addition, version C30 allows simultaneous file reorganization and online transaction application processing. Other enhancements allow batch applications to take advantage of the Tandem system's parallel architecture.

This article is designed for the technical reader interested in improving batch performance and reducing overall system resource consumption. It describes enhancements that are automatically implemented by the software and those that a user can control programmatically. This article also discusses how user-written batch applications can be developed to achieve performance gains. Releases C10, C20, and C30 of GUARDIAN 90 and corresponding releases of associated software products are covered.

The information in this article provides guidelines for:

- More efficient use of idle system resources during off-peak hours.
- Reducing the amount of time to complete a batch application.
- Reducing the overall consumption of system resources by improving the efficiency of batch jobs.

Improvements Available with Version C10 of GUARDIAN 90

Releases of GUARDIAN 90 prior to version C10 offered several enhancements to ENSCRIBE and NonStop® SQL that have improved the performance of batch processing. They include generic record locking, buffered writes, the PARALLEL option of FASTSORT (the Tandem batch sorting program), sequential block buffering, compression of audit checkpoint records, unstructured buffer lengths, read-through locks, and unstructured cache bypass.

The C10 version of GUARDIAN 90 further improved batch processing performance by increasing the efficiency of inter-controlpoint flushing and cache management and by reducing block splitting and index accesses. Benefits that are only available to NonStop SQL with GUARDIAN 90, version C10, include automatic virtual sequential block buffering, automatic large input/output (I/O), and asynchronous prefetch and postwrite.

COBOL85 Fast I/O

The COBOL85 runtime library automatically blocks and buffers accesses to the disk. This improves, by up to ten times, the throughput of programs using ENSCRIBE entry-sequenced files. Referred to as *fast I/O*, this process occurs automatically when the programmer specifies certain COBOL85 clauses such as RESERVE, no LINAGE, and ACCESS MODE SEQUENTIAL.

The maximum size of a data transfer provided by fast I/O is 4 Kbytes. This facility also automatically transfers data using double-buffered no-wait I/O. This makes it faster, for example, than 4-Kbyte blocking mechanisms written by an application programmer in the standard synchronous (waited) manner.

COBOL85 fast I/O is one of several I/O blocking schemes available to batch processing. Figure 1 lists those described in this article and ranks them according to their typical effectiveness.

Improved Inter-Controlpoint Flushing

For effective transaction management, it is necessary to ensure that data updated in the disk cache is written to disk at regular intervals (usually every five minutes). A DP2 procedure called *controlpoint flushing* does this by writing to all the disk cache buffers that have been modified but have not yet written back to disk. The simultaneous execution of batch processes, however, can increase the amount of controlpoint flushing.

To reduce the amount of data written every five minutes, DP2 attempts *inter-controlpoint flushing*, or writing data during periods of low disk I/O between the 5-minute controlpoints. DP2, version C10, improves the process by sorting the buffers by their disk addresses and then issuing the write command using bulk I/O (up to 28 Kbytes of data per physical write) when possible. Sorting of disk addresses provides more efficient write operations, improves overall throughput, and makes system response time more consistent for OLTP as well as for combined OLTP and batch processing.

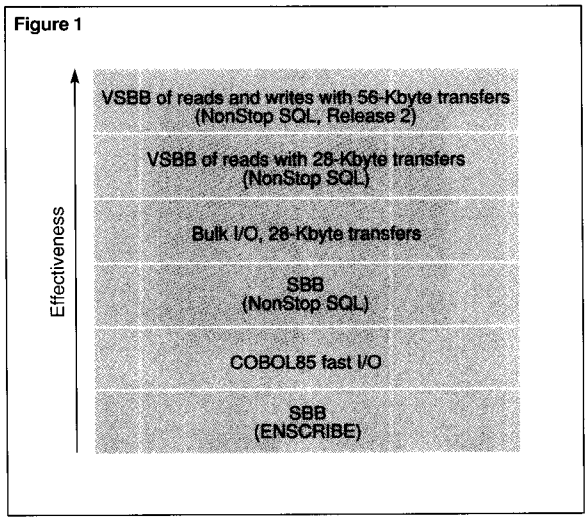


Figure 1.
Ranking of typical performance improvements for six buffering schemes. Use this table as a starting point to choose an appropriate method.

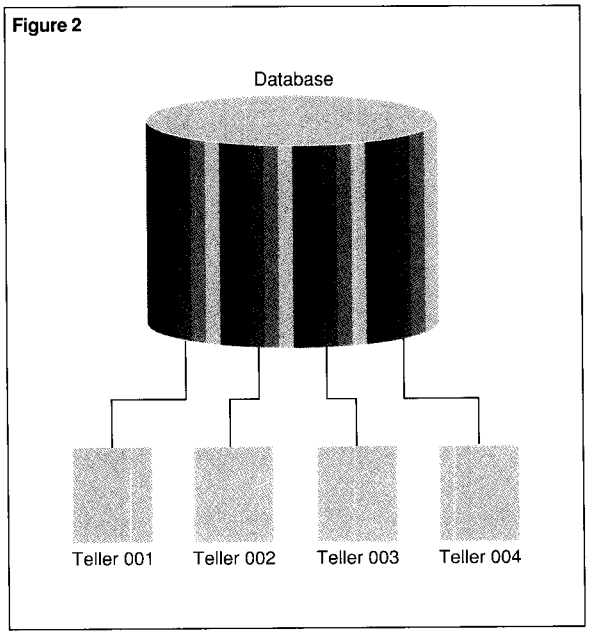
Retain Cache at Close of File

Prior to the C10 version when all application programs close a file, the cache information is no longer available. With the C10 version, the cache information remains available for a period of time. This gives successive applications that use the same file an opportunity to use the cache information without having to rebuild it from scratch. The use of cache is, therefore, more efficient and system resources are conserved. For example, batch jobs that close and then reopen the same files seconds later may still have cache information available to them and, as a result, can run faster. This improvement is completely automatic with GUARDIAN 90, version C10.

Bulk I/O

The GUARDIAN 90 *bulk I/O* facility transfers 28 Kbytes of data to and from disk with a single I/O. The use of bulk I/O can reduce total execution time by one-fourth over that used by applications with typical record-at-a-time routines.

Figure 2.
Logically sequential insertion of records into a banking database.



All application languages must call a GUARDIAN 90 file system procedure such as SETMODE 141, parameter 1, or send interprocess communications to FUP (File Utility Program) to provide bulk I/O. The use of bulk I/O should be restricted to unstructured files. For structured files, the application that generates the records can pass those records directly to FUP so it can take advantage of bulk I/O. FUP loads the file with only about one I/O every 28 Kbytes. Also, if FUP is in another CPU, these processes can occur in parallel. (Note that to reduce message traffic, the records should, if possible, be passed to FUP in blocks of records.)

NonStop SQL automatically decides when and how to transfer data using bulk I/O. Therefore, the decision to use bulk I/O is only important when using ENSCRIBE files. NonStop SQL automatically uses bulk I/O for writes to structured files.

Reduced Block Splits for Sequential Inserts

A change was made to DP2 file management that detects logically sequential insertions to files and changes the block-splitting algorithm so that fewer block splits occur.[1] Therefore, files can consume less physical disk space. This improvement lowers CPU and disk activity and reduces disk space requirements.

As described in the following example, there are some situations, however, in which the automatic detection of logically sequential inserts does not occur quickly enough to be of assistance. Tandem provides a method to manage this situation.

The example in Figure 2 shows the logically sequential insertion of records into a banking database. The database file depicted inserts records with a primary key of TELLER NUMBER and TIMESTAMP (such as 001880101120000, where the teller is 001, the date is 880101, and the time is 12:00:00). The bank has just four tellers online, but their activities always result in logically sequential additions to the file.

Figure 3 shows the block splitting for this example. The first block of teller 001 data splits because the tenth record does not fit. Only half the records are carried into the new block. However, a few minutes later when the teller has completed enough transactions to also fill that block, it splits and carries half the data to the next block. This process continues for all insertions for all tellers and results in a file that is slightly more than half full.

The DP2 file management system then analyzes the insertions. If five insertions are all logically sequential within a block, the next insertion results in a block split where no records are brought forward. Therefore, the next insertion begins filling an empty block, and the previous block has no empty spaces. Figure 4 illustrates the new process.

There are two shortcomings with the improved approach. First, in a typical system, the teller does not make five entries into the file before another teller makes an entry into the file. Once an entry occurs that is not logically sequential to the previous one, the DP2 file system returns to the *half-block-splitting* algorithm. This may mean that the *leave-full-and-start-new-block* algorithm is never (or rarely) activated.

Second, once the system moves on to a new block, it starts over at the half-block-splitting algorithm again. This further defeats the leave-full-and-start-new-block algorithm. Therefore, in situations where the data tends to consist of four to ten sequential inserts at a time, measurable reductions in block splits can generally occur by enabling SETMODE 91, parameter 3, to always use the leave-full-and-start-new-block method.

The C30 version of GUARDIAN 90 eliminates the block boundary limitation of the leave-full-and-start-new-block algorithm. Therefore, the block limitation no longer has any effect on which algorithm is used. However, if an application tends to mix writing to multiple tellers at a time, the SETMODE 91, parameter 3, may be used to call GUARDIAN 90 so that it permanently sets the block splitting for that OPEN to leave-full-and-start-new-block.

Note that whenever records are added to EOF (end of file), the leave-full-and-start-new-block method is always used regardless of the GUARDIAN 90 version or the SETMODE parameter. An overview of other block-split considerations can be found in Glasstone, 1986.

Reduced Index Access

SETMODE 91, parameter 3, also stops unnecessary *tree walking* (reading all the index levels to find the location of a record). This call ensures that when a record is added and is logically sequential to the previous record, unnecessary physical disk reads or cache accesses do not occur.

If SETMODE 91, parameter 3, is enabled and a record is accessed that is not logically sequential to the previous one, no error occurs. The system recognizes that the record it is about to insert must be preceded by a tree walk, and the tree walk then occurs automatically.

Figure 3

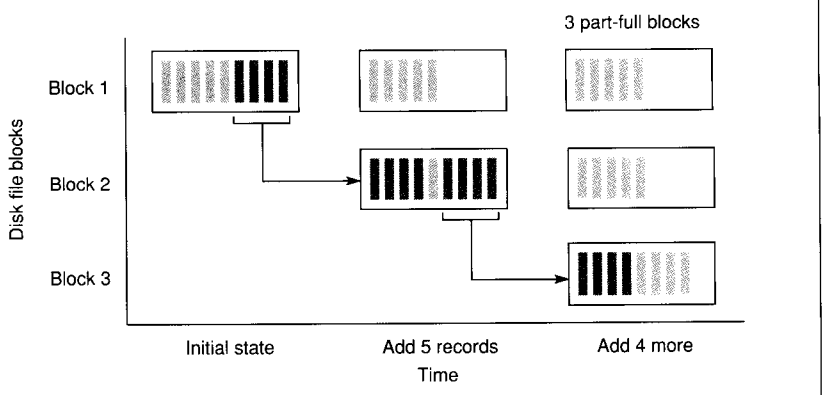


Figure 4

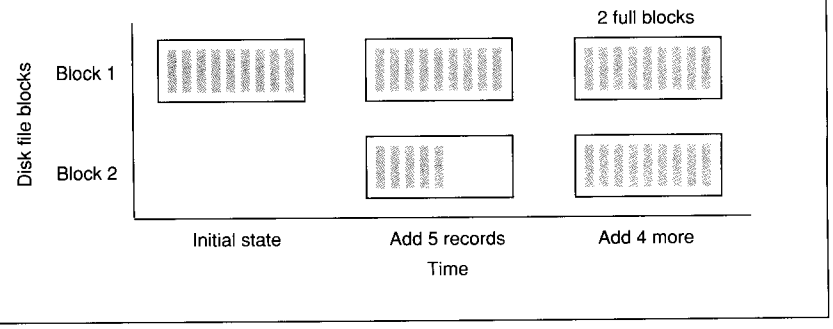


Figure 3.
Block splitting occurs when records are sequentially added to a key-sequenced file. This results in a slightly more than half-full file.

Figure 4.
Version C10 of GUARDIAN 90 improves block splitting for sequential inserts, which results in less disk consumption.

As compared to always tree walking, a slight amount of additional overhead is incurred when SETMODE 91, parameter 3, is in effect and a tree walk must occur. Therefore, do not enable the SETMODE if a tree walk is generally needed or a performance degradation could occur.

The DP2 file management system also stops tree walking after the fifth insertion of a logically sequential record. This feature was part of GUARDIAN 90 prior to release C10; however, it is possible that the insertion of logically sequential records may tend to be only three or four records instead of more than five at a time. Therefore, if the application program performs a SETMODE 91, parameter 3, index accessing can be more efficient.

Generic Record Locking

ENSCRIBE and NonStop SQL provide generic record locking for transactions that must lock many records per transaction. Applying generic locks can help system performance by reducing main memory usage, CPU consumption, and execution time.

A generic lock can be applied to the leading portion of the primary key of any key-sequenced file. As a result, the main memory space consumed by locks is reduced, and more records can be locked at one time. Generic locks can be removed with less overhead per record than typical locks. In many situations, this can save a significant amount of system resources and time.

Generic locks enhance system performance.

There are, however, some limitations to the ENSCRIBE implementation of generic locking. For example, ENSCRIBE does not stop applications that have access to unaudited files from adding records that are in the generic lock range. Therefore, although records that existed at the time the locks were applied cannot be accessed, another application can add records within the generic lock range. This typically causes problems for the application that issued the generic lock. The control of this problem is left to the application designer. (Note that NonStop SQL does not have this limitation.)

The only way to remove ENSCRIBE generic locks is to use the GUARDIAN 90 file system procedure call UNLOCKFILE. Therefore, to remove a generic lock, the programmer must remove all the locks, close the file, or if the file is audited, use END TRANSACTION. (This limitation does not exist with NonStop SQL.)

Even with the limitations described above, generic locking is a powerful tool for improving the throughput of many applications. A generic lock range is established for NonStop SQL using the NonStop SQL DDL (Data Definition Language) statement and for ENSCRIBE either with FUP or programmatically within the application.

Buffered Writes

Buffering, or deferring writing records to disk until several records are accumulated, reduces the number of disk writes and usually improves performance. The default for buffered writes is OFF for unaudited files and ON for audited files. A buffered write updates disk cache in both the primary and backup CPUs if SYNCDEPTH is greater than zero and the file is unaudited.

ENSCRIBE files can be opened as buffered or unbuffered (write through cache) files on an individual basis by using the SETMODE parameter. If the method used to recover a failed batch job is to restart, it may be advisable to use buffered writes for unaudited files.

FASTSORT Parallelism

The FASTSORT PARALLEL option allows the user to call up to eight processors to work simultaneously on portions of the file. This can reduce the elapsed time to sort the file by several times over using one processor. One of the parallel FASTSORT processors acts as a traffic manager for the others and, thus, should not participate in subsorting when a large number of processors are used.

In some situations, FASTSORT performance can also be increased by specifying the MINTIME option. MINTIME instructs the processor to consume more main memory than the AUTOMATIC option (the default) specifies. Alternatively, the user can explicitly specify main memory consumption (Gray, 1986).

Because the algorithm used by FASTSORT cannot be expected to always make the best decision for a particular environment, it may be worthwhile for the user to explicitly set memory consumption for long or often-used sorting runs. (Note that COBOL85 programs operating under GUARDIAN 90, version C10, can directly call the FASTSORT PARALLEL option.)

Sequential Block Buffering

Sequential block buffering (SBB) is the procedure where requesting a read of a single record brings an entire block of data to the application process file segment. The record requested is then sent into application working storage, thereby reducing the number of disk reads and CPU overhead. Subsequent reads then can be made from the block of records in the process file segment. These subsequent reads occur faster than similar reads that might otherwise occur from DP2 cache or from disk. (Note that SBB for ENSCRIBE has no record-locking capability.)

A common misconception is that the file must be OPEN EXCLUSIVE to use SBB on ENSCRIBE files. This is not the case. When the file is opened for shared access, other processes can update records in the block that was read in by the batch process. Therefore, if three records are in the block, any of those records could be changed on disk by another process before or after the initial block is read in by the batch process. If the application does not write to the data file being read with SBB, and if it is acceptable for the data to possibly be slightly out of date, these limitations may be manageable.

A file being read with SBB can be positioned and then read sequentially. Random reads and writes operate correctly, but the benefits of SBB do not apply to random accesses.

Despite the limitations of SBB, the savings in read time are considerable; 100-byte records on a NonStop TXP™ system can be read more than twice as fast as those not using SBB (Welsh, 1984). Other analyses measured SBB to give a performance increase of a factor of three over the usual record-at-a-time interface (Tandem Database Group, 1987).

In the case of reading in alternate-key order, SBB operates on the alternate-key file but not on the data file, which limits the benefits. (Refer to Mattran, 1986, for a detailed description of the use of SBB for ENSCRIBE files.)

SBB is supported for COBOL85 entry-sequenced files using the RESERVE clause. (Refer to Tandem language reference manuals for other language calls.)

Compress Audit Checkpoint Records

If auditing is used or if SYNCDEPTH is set to 1, invoking SETMODE 94 causes compression of checkpoint records and audit logs. Compression of the records, although consuming a slight amount of CPU time, can reduce I/Os and I/O traffic on the audit trail and can reduce message system traffic, thus improving performance. (This process currently works only with ENSCRIBE.)

Unstructured Buffer Length

The cache buffer length is routinely set to 4 Kbytes when an unstructured file is read. However, if random reads of the unstructured file are occurring and the application tends to access much less than the 4 Kbytes in the buffer, shortening the buffer length may help performance of both the application and the overall system.

Use SETMODE 93 to modify the buffer length. SETMODE 93 should only be used with unstructured files.

Parallel Hardware Writes

If an application is performing writes to one mirrored volume that is connected to two controllers, it is possible to transfer data to both halves of the mirrored disk in parallel. It is important, however, that the system configuration is correct. There are fault-tolerant risks associated with this process that must be managed.

Use SETMODE 57 to enable parallel hardware writes. For NonStop SQL, specify parallel hardware writes with CREATETABLE or ALTER.

Read-Through Locks

Batch processing applications can be unnecessarily constrained by other applications that place record locks against the file. If using ENSCRIBE, these jobs can ignore the locks and proceed with processing by issuing SETMODE 4. If using NonStop SQL, the batch applications can use the BROWSE access.

Unstructured Cache Bypass

Bypassing the cache can provide faster reads and writes if there is little value in having the data in the cache. Unstructured cache bypass appears to be beneficial when reading and writing starts and ends on block boundaries.

Unstructured files can bypass the DP2 disk cache (under various limitations) by issuing SETMODE 91, parameter 1. This call, however, is not needed for bulk I/O because it automatically bypasses the cache. Although SQL sequential prefetch automatically bypasses cache, there is no command to instruct NonStop SQL to bypass cache for other operations.

NonStop SQL Virtual Sequential Block Buffering for Reads

Using the disk process selection and projection, the ENSCRIBE concept of sequential block buffering was extended for NonStop SQL to *virtual sequential block buffering* (VSBB) blocks. In VSBB, data is returned through the disk-process read interface after projected fields have been extracted from key-range-satisfying records. The performance gains of VSBB are from the reduced message traffic that results from filtering data at its source so that only qualified fields are selected and only the projected data is returned to the requester (Borr, 1988).

DP2 decides which records meet the specifications of the read request and passes back to the application only those columns in the records meeting the requirements. VSBB performs best on files located on different CPUs from the main application process, partitioned files, and geographically distributed files. VSBB has been shown in some situations to improve performance by a factor of three over SBB. SBB, in turn, has been shown to improve performance by a factor of three over record-at-a-time interfaces (Tandem Database Group, 1987).

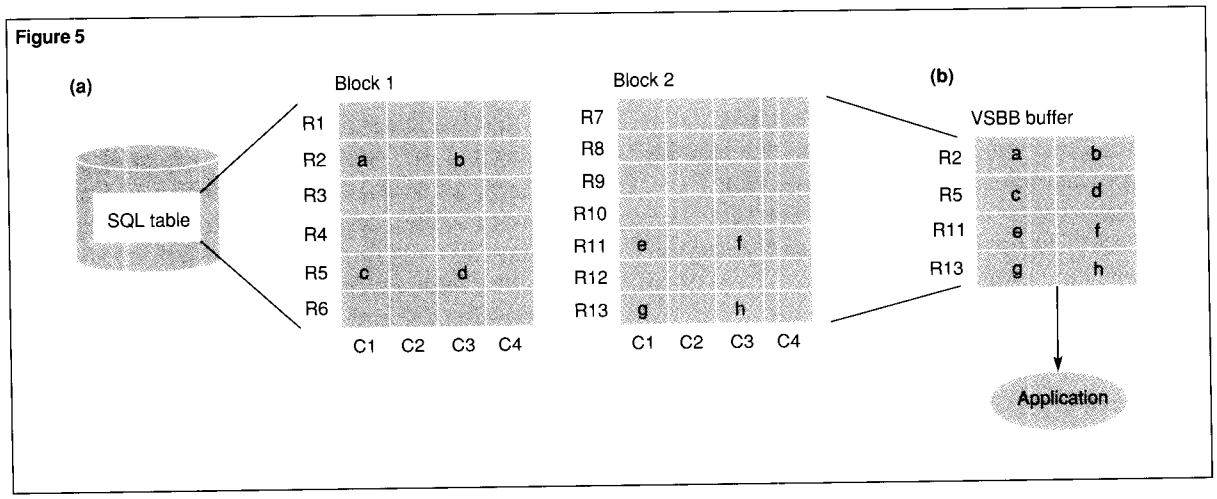


Figure 5.
DP2 filters data through the VSBB mechanism. (a) Rows and columns of data that meet selection criteria are read into the VSBB buffer. (b) Condensed VSBB buffer consisting only of relevant information is sent to the application.

VSBB works only for SELECTS with release C10 of GUARDIAN 90. Additional improvements to VSBB are available in version C30 of GUARDIAN 90.

The examples in Figure 5 illustrate how DP2 filters data through the VSBB mechanism. After DP2 performs both projection (selecting only requested fields) and filtering (returning only qualified records), the logical records move into the VSBB buffer and then to the application. Because projection and filtering are low-level DP2 processes, there is a significant reduction in data traffic and an increase in performance.

In Figure 5a, DP2 can read blocks 1 and 2 of a file and select records R2, R5, R11, and R13. If only columns C1 and C3 are selected, the VSBB buffer looks similar to Figure 5b.

Large I/O for Sequential Reads/Writes

NonStop SQL detects when it is reading or writing in a sequential manner and during this time uses single transfers of seven blocks. Therefore, if the block size is 4 Kbytes, a total of up to 28 Kbytes is transferred into main memory with one I/O.

Asynchronous Prefetch for Sequential Reads

NonStop SQL reads ahead of the application by using *sequential prefetch*. Sequential prefetch issues bulk I/O during idle time (for example, asynchronously to the application process). Depending on system load conditions, sequential prefetch can provide the application with 100% disk cache retrieval rate.

Prefetch is performed for the statements SELECT, UPDATE, and DELETE, which each specify a BEGIN-KEY and END-KEY. This function is completely automatic.

Asynchronous Post-Write for Sequential Writes

NonStop SQL can issue bulk writes asynchronously (in idle time) so that sequential cache blocks are written to disk after the related audit has been written. This saves DP2 from having to synchronously write old cache blocks one at a time to make way for new blocks coming in.

Post-write is performed for the NonStop SQL UPDATE and DELETE statements. This function is completely automatic.

Improvements Available with Version C20 of GUARDIAN 90

The C20 version of GUARDIAN 90 increases the available cache and main memory. These memory enhancements improve network message traffic efficiency, optical disk performance, and overall system performance.

EXPAND Efficiency Improvement

Completely automatic with version C20, the EXPAND™ networking software reduces message traffic and speeds up delivery of messages in heavily loaded environments by including message acknowledgments in message packets that are already traveling in the same direction. This mechanism is especially important for batch jobs operating in a distributed environment, such as X.25 communications.

Optical Disk Cache

To improve read access speeds, GUARDIAN 90, version C20, introduces a read-only cache for the optical disk process (ODP). The optical disk process is a fault-tolerant process pair that handles requests to an optical storage unit from PUP (Peripherals Utility Program), FUP, TMDS (Tandem Maintenance and Diagnostic System), and application programs.

Optical Disk Sequential Read and Write

New routines in the ODP improve most sequential disk accesses by fully supporting structured read access and by using the serial read and write features of the hardware. The SETMODE command allows the application programs to take advantage of serial write, and the ODP automatically uses serial read. FUP DUP also takes full advantage of these serial read and write facilities. First available with an upgrade to the C10 version of GUARDIAN 90, this improvement also results in faster backup and restore processing.

Larger Maximum Cache Available

In version C20, the maximum size of software cache has been increased from 6 Mbytes to 56 Mbytes per disk volume. This increased cache improves throughput for various system configurations, especially random I/O operations.

The additional main memory required is available using Tandem's VLM™ (Very Large Memory), which allows main memory in NonStop VLX™ CPUs to be expanded to 96 Mbytes. Tandem systems that use VLM and are already configured with large amounts of main memory may be able to specify caches larger than 6 Mbytes per disk without any additional main memory.

In the event of a CPU failure, the cache must be reconstructed in the CPU that takes over the duties of the failed CPU. As a result, the larger the total disk cache becomes, the longer the recovery period if the CPU fails.

Improvements Available with Version C30 of GUARDIAN 90

GUARDIAN 90, version C30, offers online file reorganization, adding to the ability of Tandem systems to simultaneously run batch and OLTP. In addition, audited files can be reorganized while OLTP applications are running.

Working in conjunction with version C30, Release 2 of NonStop SQL automatically executes SQL query statements in parallel to take advantage of multiple processors. This release also offers faster writes to disk, updates tables with multiple indexes more quickly, retrieves index information with fewer disk accesses, and has greater throughput with less system resource consumption because of optimizer[2] enhancements.

[2]The NonStop SQL optimizer automatically selects an efficient access plan for each database query.

Parallel Execution of SQL Statements

NonStop SQL performs parallel execution[3] within SQL query statements for such operations as table joins, SELECT AVG calculations, and batch temporary file manipulations.

Use the CONTROL EXECUTORS PARALLEL EXECUTION ON statement to enable parallel execution. The default is CONTROL EXECUTORS PARALLEL EXECUTION OFF, which disables parallel execution.

When parallel execution is invoked, all processors in the node are used. This can be overridden if the SQL optimizer uses file partition[4] boundaries (the number of partitions sets an upper limit on the number of processors that are used). Also, CMON can be used to restrict access to some CPUs.

For example, a table entitled EMPLOYEE has six partitions, and the query

```
SELECT AVG(SALARY) FROM
  EMPLOYEE
```

asks for the average salary for all employees.

NonStop SQL can start up six processes, one for each partition. Each of these "subprograms" reads records serially from its partition and computes the average salary for its partition. All six subprograms execute in parallel in different CPUs. When all the appropriate table rows (records) have been read by a subprogram, the summary information is sent to a "master" NonStop SQL process. After the master process has the averages from all six subprograms, it computes the average salary and delivers the results.

In this example, a parallel implementation is approximately six times faster. Note that the partitions could also be located on six different nodes, not just six different processors within a node.

[3]Parallel execution, in this case, refers to NonStop SQL using multiple processors to read, filter, and write files.

[4]A partition is a portion of a table or index that resides on a particular disk volume. SQL tables and indexes can be partitioned.

Another example (see Figure 6) has two local tables. The first table, named CUSTOMER, has NAME as the primary key. The second table, named ACCOUNT, has NAME ACCOUNT_NO as the concatenated primary key.

Consider the following query:

```
SELECT * FROM CUSTOMER, ACCOUNT
WHERE
CUSTOMER.NAME = ACCOUNT.NAME
AND
ACCOUNT.BALANCE < 0
```

If the CUSTOMER and ACCOUNT tables start with three partitions each, the optimizer can select three processors for use throughout the process. If, however, the tables are not partitioned and the optimizer chooses to extract temporary tables prior to joining them, parallelism may be automatically invoked using two CPUs to create the partitioned temporary files.

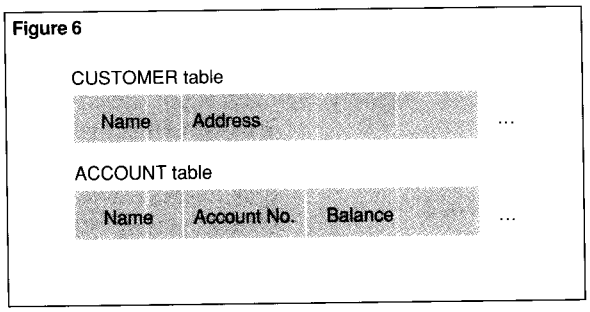


Figure 6.
Example of two tables on which NonStop SQL Release 2 may automatically apply parallelism in the extract and join process.

These temporary files each may have three partitions, and when the join occurs, three processors are used. Furthermore, writing of any permanent output file also occurs in parallel. This example applies to a single-case query as well as a batch application.

Virtual Sequential Block Buffering for Reads and Writes

Version C30 makes VSBB operational for write as well as read operations. In addition, the additional VSBB provides substantial peformance gains over VSBB for reads (supplied with version C20) because the number of disk I/Os and the CPU utilization due to message traffic is further reduced.

VSBB is controlled by the CONTROL TABLE * SEQUENTIAL INSERT ENABLE/ON/OFF commands and is obeyed by NonStop SQL for INSERT and UPDATE... WHERE CURRENT. Because the default is ENABLE, VSBB is automatic for C30-compiled programs.

Improved FASTSORT Data Passing

Enhancements to GUARDIAN 90, version C30, improve the speed at which FASTSORT reads records from NonStop SQL sort input files. First, instead of 4-Kbyte transfers of data from disk to FASTSORT, 28-Kbyte bulk I/O transfers are used for general selection expressions. Secondly, instead of the NonStop SQL executor reading the information from disk and then passing the data to FASTSORT, FASTSORT is able to read the data directly from disk into its own buffers, thus reducing data traffic. Prior to version C30, files that are not already in SQL format are read directly by FASTSORT and with 28-Kbyte transfers.

Version C30 provides asynchronous and parallel update of alternate indexes.

Parallel Index Maintenance

GUARDIAN 90, version C30, significantly improves the performance of inserts and updates in NonStop SQL tables having multiple alternate indexes. Prior to version C30, a NonStop SQL key-sequenced file added and updated rows to the indexes in a serial manner. For example, a table that had a primary key and three alternate keys would issue the update for the base table and wait for a response, then issue the update for the first alternate key and wait for a response, then continue its updating for the second and third keys.

With version C30, after SQL accomplishes the base table update, all alternate index update requests are issued asynchronously; the updates are issued in rapid succession without waiting for the completion of an update. The application pauses and waits for each update to occur. In addition, if the indexes are located on disks controlled by other CPUs, the CPU that controls the disk manages the processing of the updates. Therefore, the update of the alternate indexes is not only asynchronous but is also parallel.

If a key-sequenced file has more than one alternate key, version C30 improves the elapsed time. This is especially apparent in situations where the indexes are created on different disks, and even more so if those disks are on different CPUs.

Index-Only Retrievals

Occasionally, a retrieval of SQL information involves columns of data that are described in an index, and the base table is not needed to satisfy the query. With the C30 release, if the base table row is not needed, SQL does not retrieve it, and therefore, performance improves.

Online File Reorganization

Any audited file may be reorganized online with GUARDIAN 90, version C30. Therefore, files may be reorganized simultaneously with OLTP, which can allow more batch processing to be performed within a given period of time.

Files using the GUARDIAN 90 operating system need infrequent reorganizing. For example, the space within a structured file that results from record deletion is automatically made available for use. Reorganization, however, is occasionally appropriate because empty or partially empty blocks can result from a large number of inserts, deletes, or updates. Also, data clustering can be improved and the number of index levels may be reduced by reorganizing. Because online file reorganization may make it desirable to reorganize files more often, the resulting cleaner file structure saves both disk space and time to access files.

Online file reorganization provides a parameter, or throttling mechanism, that allows the reorganization to occur at 1% to 100% maximum possible speed. Therefore, to minimally impact OLTP applications while reorganizing the file, a low speed can be specified for the online reorganization.

The throttling mechanism is especially useful because the reorganization software works closely with DP2. The online file reorganization process sends only small I/O commands to DP2. The reorganization process establishes the time of the request to DP2 and the time that DP2 satisfied the request. The difference in the times, known as the *delta,* is then calculated.

If the user specified 100% throughput for the reorganization, no throttling occurs. If the user specifies 50%, the application waits for one delta and then sends the next request. Similarly, if 10% is specified, the reorganization process waits for ten deltas before sending the next request. (Note that this throttling mechanism easily adapts to changing system loads, which further adds to its effectiveness.)

Online file reorganization uses a negligible amount of additional disk space within the file while reorganizing. This file space and any file space freed by the reorganization returns to the system upon completion. However, because audit records are created for each record reorganized, significant amounts of audit trail are consumed. (Only audited files may be reorganized online.)

Optimizer Enhancements

With NonStop SQL, Release 2, the optimizer allows both sides of a statement with the OR operand to use indexes of items. This improves execution time and reduces disk activity.

In another enhancement, more control of optimizer decision making is possible with a command expressly designed to aid batch throughput. Issued within the application as a compile directive, the default setting is CONTROL QUERY SET INTERACTIVE ACCESS OFF (or simply, INTERACTIVE OFF), which instructs the optimizer to utilize sequential reads in situations when it is uncertain whether sequential or indexed reads are optimal. The CONTROL QUERY SET INTERACTIVE ACCESS ON (or simply, INTERACTIVE ON) command indicates to NonStop SQL that when host variables are in use, the use of an indexed read may be more appropriate than a sequential read.

The usual results of establishing INTERACTIVE ON are that the first records to be read return more quickly and that the last record to be read by the command arrives later than INTERACTIVE OFF. Also, the arrival rate of the records that are retrieved by INTERACTIVE ON tends to be more regular. (Note, however, that it is likely that the INTERACTIVE ON command consumes more total system resources by the time the batch job finishes reading the tables.)

There may be a few situations in which the optimizer could read a table sequentially and there is an unexpectedly small number of records to be returned. In these cases INTERACTIVE ON would be superior to using INTERACTIVE OFF.

Clustered Keys

Clustered keys are an SQL database option that allows data to be geographically distributed based on a primary key that is not logically unique. Partitioned databases are easier to access with parallel applications programs and the automated parallelism of NonStop SQL improves elapsed execution time.

Clustered keys make it possible to establish a primary key that represents only the volume or system on which to geographically distribute table rows. This is true even if the primary key is not unique.

Application Techniques

There are several application development techniques that can be used in the GUARDIAN 90 environment to speed sequential processing. Among them are process execution priority, batch scheduling, disk contention, partitioning, SPOOLER considerations, COBOL prereading, sequential I/O, and parallelism.

Process Execution Priority

Generally, batch applications should run at a lower execution priority than the OLTP processing to minimize the impact on OLTP applications. Batch jobs can also have different execution priorities. For example, a batch job in CPU 1 that is running at a low execution priority can be competing with a batch job also in CPU 1 that is running at an even lower execution priority. The Tandem system services the higher priority job more quickly than the lower priority job. The difference in runtimes can be advantageous when the completion of one job is more critical than another.

NetBatch-Plus software structures the batch environment.

Automated Scheduling of Batch Jobs

The NetBatch™-Plus software product, introduced with GUARDIAN 90, version C20, structures the batch environment, manages the execution of batch jobs based on scheduling options and priorities, distributes the execution of jobs across different processors, and controls all processes and dependencies between jobs. Once the options are selected on the NetBatch-Plus screens, batch processing can take place without operator intervention.

Job Selection Priority. NetBatch has a *selection priority* feature that allows specified jobs in the queue (waiting to execute) to have priorities within the queue. No matter how many other jobs are ready and waiting to start execution, if a new job has a higher SELECTION PRIORITY, it is submitted by NetBatch for execution. However, once the processing starts, the selection priority has no effect on the process execution priority. NetBatch can also create an extra queue to ensure that a job runs at a specified time (*NetBatch User's Guide,* 1989).

Control Number of Simultaneous Batch Jobs. NetBatch manages the number of simultaneous batch jobs in execution so that unwanted degradation of interactive jobs or critical batch jobs is avoided. Such queue management helps prevent batch jobs, including reports and compiles, from impacting OLTP (Wakashige, 1988).

Fast Recovery from Errors. Because of the Tandem system's fault-tolerant architecture, fewer failures due to malfunctioning hardware can be expected. However, as in any batch environment, unexpected errors in programming or unusual environmental factors (such as lack of disk space) can cause the failure of a job. As a result, a method of fast recovery is critical to the performance of batch processing.

References to files within a job should be performed using the GUARDIAN 90 DEFINE statements. This makes it convenient to access different files and devices should the scheduled job fail and need to be restarted. Referencing files in this manner aids performance because restarting a failed job is faster and less prone to error than explicitly stating file names in the program.

CPU Selection. One method of providing parallel batch processing on Tandem systems is to run batch jobs simultaneously in multiple CPUs. Ideally, batch jobs are run on CPUs with otherwise low usage and low OLTP activities. A batch scheduler such as NetBatch-Plus can help prevent a CPU from running too many batch processes at once.

TMF Use in Batch Jobs

Batch jobs usually do not use Tandem Transaction Monitoring Facility (TMF™) for auditing. In the case of a failure, jobs that do not write to master files can usually be restarted from the beginning. Batch jobs that are good candidates for TMF, however, are those that write to master files, cannot afford to start over from the beginning, or are part of applications that require simultaneous OLTP and batch processing.

Batch use of TMF is usually slightly different than for OLTP. Because TMF writes at least one audit record and one commit (ensures action of request) record for each transaction, TMF can become a significant bottleneck for some batch programs. Furthermore, the advantages of buffered writes does not usually occur because of the fewer number of writes that make up a batch transaction.

If possible, several logical transactions should be grouped into one TMF transaction. This allows efficient buffering of audit records and fewer commit records to go to disk. Many experienced batch designers write their programs so that 10 to 100 logical transactions can occur between the TMF statements BEGIN TRANSACTION and END TRANSACTION.

It is important to note that the disk process limits the number of records that can be locked at one time, and this number can be exceeded (Glasstone, 1986). Also, the more records locked at once, the more likely that an OLTP process encounters one of the batch jobs' locked records.

LOCKFILE is a GUARDIAN 90 file system procedure that locks all records in a file. In controlled situations, it can be used to eliminate record-level locking to improve performance and reduce the risk of exceeding the maximum number of permitted record locks.

Although the TMF AUDIT attribute can be turned off to increase performance in some instances, it is not recommended; there is a risk that TMF will not be turned back on. Considering the risks associated with performing unaudited updates on files, in most cases the relatively minor level of performance enhancement created by turning off TMF is not warranted.

Refreshing End of File

When the file label control called REFRESH is ON, the file label is written to disk every time the end-of-file (EOF) changes and may cause a considerable slowing of the entire system. The REFRESH attribute should be OFF for most files (possible exceptions exist when the nature of the application requires that the disk directory always reflect the current value of EOF). A general rule for refreshing the EOF is: REFRESH should be set to OFF when recovery from the failure of a batch job requires deleting the file that was created and rerunning the batch job (Welsh, 1984).

Reduce Disk Contention

If two or more files on the same volume are being accessed simultaneously, moving one or more files to different disk volumes can provide an immediate reduction in disk-search time. Exceptions to this rule include parallel processing or other methods that tend to make a CPU the most constrained resource. In these cases, files may be read by a disk process in one CPU and then processed in another CPU.

It is also possible to reduce disk swapfile[5] activity against specific disk volumes by spreading object files to several disks. When object file libraries exist only on a single volume, the system risks degrading production performance.

Similarly, reduction in disk controller contention should be considered. Moving the file to a controller that is not heavily used can add to the performance improvement. It is common to see at least a 15% improvement in performance by paying special attention to contention issues.

Large Block Sizes

Larger block sizes for structured files usually aid performance for sequential access. Large block sizes reduce I/O operations.

[5]A swapfile is a temporary holding place on disk that GUARDIAN 90 uses to store applications that would otherwise reside in main memory. GUARDIAN 90 uses swapfiles to release main memory to other applications.

Partitioning for Performance

Heavily used files or large files can usually be partitioned to provide higher performance. Even if the file is not about to fill its current volume or if the file is a temporary one, partitioning can improve bottlenecks by spreading access across multiple disks. Note that if RESTORE or TMF ROLLFORWARD is required, only the partition that was damaged need be recovered (Welsh, 1984).

Specify Temporary File Names in COBOL Programs

By specifying a temporary unstructured file name rather than using #TEMP in the SELECT clause, the COBOL85 runtime library can be kept from creating the file on the default volume. Typically, the default volume has the greatest amount of disk contention on a Tandem system.

SPOOLER

Because the SPOOLER collector process can become a bottleneck for batch applications, be sure that enough collectors are running. Also, for batch jobs with large volume output, use a collector that has a large UNIT definition to improve collector throughput (which will have some expense in disk consumption). Use level 3 spooling for 2 to 15 times better throughput.

Unstructured Files

Use unstructured files when possible. For example, the replacement of ENSCRIBE entry-sequenced files with unstructured files (and the use of large I/Os) can provide many times faster throughput for reads and writes (Welsh, 1984).

In COBOL, a BLOCK CONTAINS clause in the file definition statement improves read and write access by writing an entire block at once.

Parallel Processing of Jobs

Performance enhancement of batch processing is available by taking advantage of parallel processing. Two ways to use multiple processors with simple application changes are to eliminate intermediate files and to run each process in multiple CPUs.

Figure 7

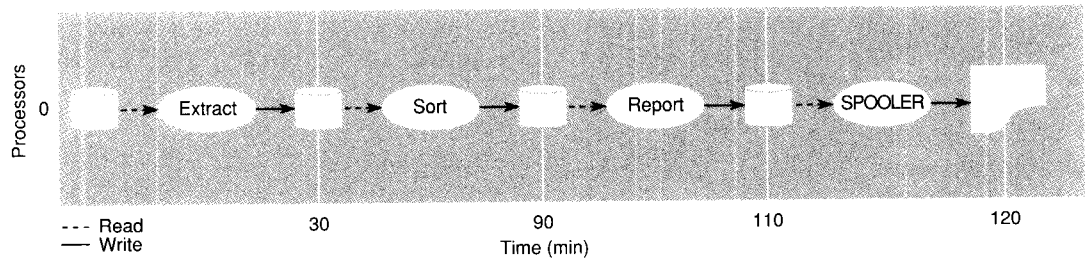


Figure 7.
Example of a traditional batch processing job flow.

Figure 8

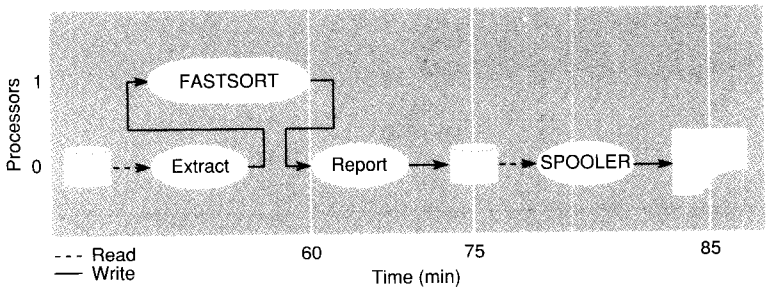


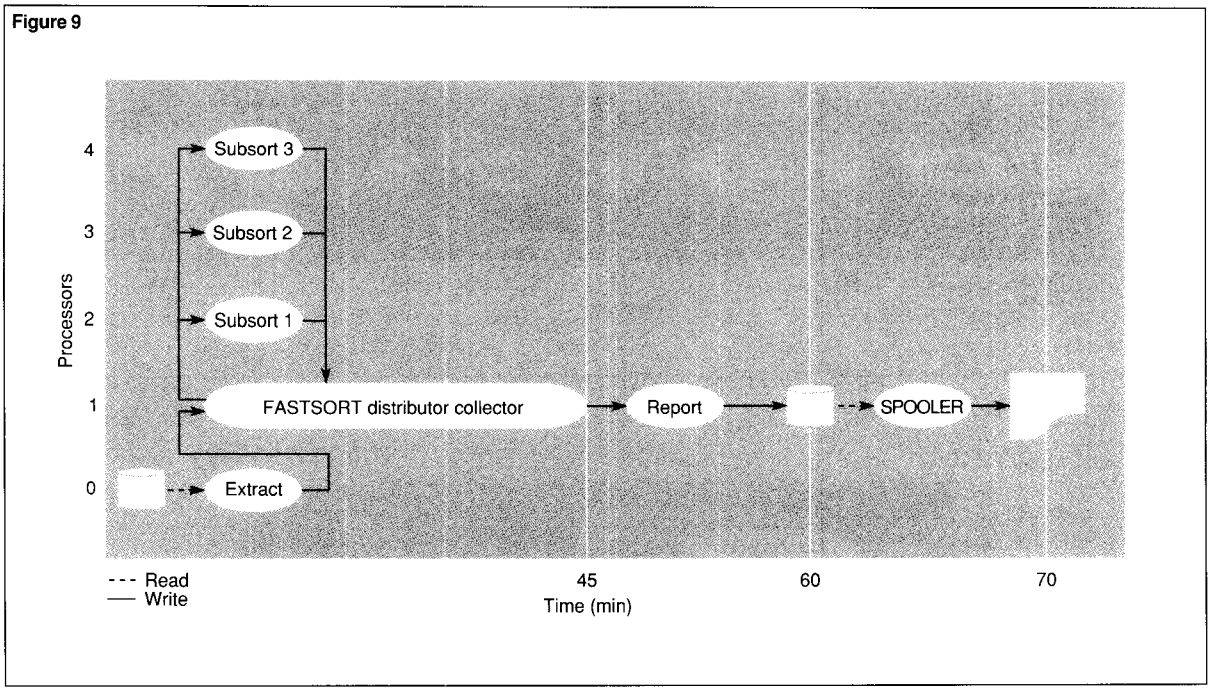
Figure 8.
Possible first step in applying parallelism to a batch job.

Eliminate Intermediate Files. Typically, batch jobs extract data from a master file, sort the extracted data into a sort output file, and then run the sort output file through a program to produce output. (See Figure 7.) The interprocess parallel processing concepts, however, eliminate the need for the EXTRACT and SORT intermediate files. (See Figure 8.)

The disk time necessary to read and write files is eliminated from this job run, and disk space is also conserved. The system manager, analyst, or operator does not have to be concerned with available intermediate work file space. (However, FASTSORT may still require scratch file space.)

It is essential that the process that sends the data to FASTSORT do so in blocks that have a maximum size of 8 Kbytes. Sending the data a record at a time defeats some of the performance enhancement. In addition, the process that sends data to FASTSORT can send it more quickly with the use of double buffering.

Figure 9.
Possible second step in applying parallelism to a batch job.



Applying Parallelism within the Processes. Each process in the previous example can run in multiple CPUs. GUARDIAN 90 communicates with processes in other CPUs within the system (node) as easily as those within the same CPU. Therefore, in the previous example, the job could be run by using the PARALLEL option of FASTSORT as in Figure 9.

The bottleneck of the slowest or longest running process can also be improved by additional parallelism at the application level. For example, if the extract process was a bottleneck, and the input master file was partitioned, separate job programs could work on each partition and write the data to a common output process called a *feeder*. As shown in Figure 10, the feeder passes the records to FASTSORT and limits to one the number of processes feeding FASTSORT, which does not allow more than one process to pass it records.

COBOL Prereading

Prereading in COBOL85 refers to I/O being performed in a no-wait (asynchronous) manner on behalf of the application program. It occurs when the file is open for INPUT, and ACCESS MODE SEQUENTIAL is stated or implied. In addition, exclusion mode must be EXCLUSIVE for tape, terminal, or card reader and EXCLUSIVE or PROTECTED for disk. Prereading can significantly increase performance.

Figure 10

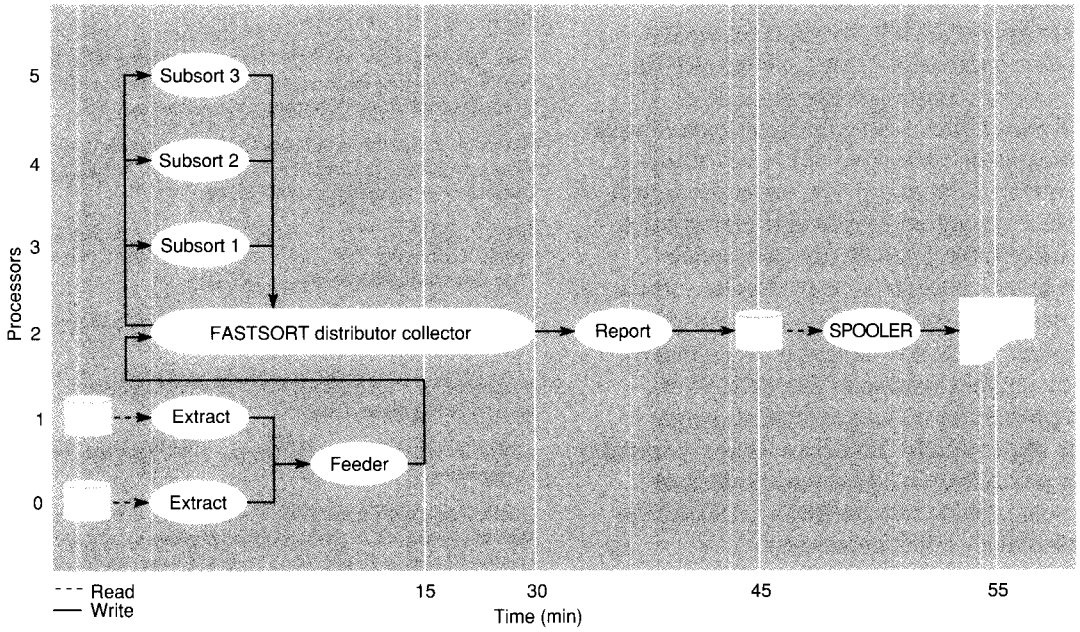


Figure 10.
Advanced parallelism applied to batch job.

Consider Sequential Instead of Random I/O

Random I/O can consume several times more computer I/O, CPU, and clock time than an equivalent number of sequential I/Os. For example, a batch job that randomly reads one-fifth of the records in a file might be rewritten to sequentially read all the records and pass over the ones that are not needed. When this is possible, the change to sequential access can result in much faster access time. Whether performance is better reading a portion of the file randomly or all of the file sequentially varies from application to application.

If the records are small, sequential reads may be up to 20 times faster than random ones. In addition, buffering methods such as bulk I/O may further improve sequential access time.

Because the NonStop SQL optimizer decides whether to use sequential or keyed access, the recommendations above apply only to ENSCRIBE. However, it is possible to bias the decision of the optimizer by issuing the CONTROL QUERY SET INTERACTIVE ACCESS ON/OFF command.

Conclusion

The techniques described in this paper are intended to be used as a foundation for understanding the elements of batch performance on Tandem systems. Starting with the methods described above, users can gain significant returns on investments in optimizing batch performance. At the same time, NonStop SQL has made significant advances in automatically optimizing batch performance. Table 1 can be used as a quick reference to the concepts described in this article. Readers can also view Table 1 as an overview, as it presents some of the advantages to be gained by moving to a higher version of GUARDIAN 90.

Batch processing no longer requires single processor sequentiality. Tandem systems' parallel hardware architecture and the multiprocessor environment provided by GUARDIAN 90 deliver high-performance batch processing.

References

Oleinick, P. and Shah, P. 1986. A Performance Retrospective. *Tandem Systems Review*. Vol. 2, No. 3. Tandem Computers Incorporated. Part no. 83938.

Borr, A. 1988. Technical Paper: High-Performance SQL through Low-Level System Integration. *Tandem Systems Review*. Vol. 4, No. 2. Tandem Computers Incorporated. Part no. 13693.

Glasstone, R. 1986. Performance Considerations for Application Processes. *Tandem Systems Review*. Vol. 2, No. 3. Tandem Computers Incorporated. Part no. 83938.

Gray, J., and et. al. 1986. FASTSORT: An External Sort Using Parallel Processing. *Tandem Systems Review*. Vol. 2, No. 3. Tandem Computers Incorporated. Part no. 83939.

NetBatch User's Guide. 1989. Tandem Computers Incorporated. Part no. 17518.

NetBatch-Plus User's Guide. 1989. Tandem Computers Incorporated. Part no. 19527.

NetBatch Independent Study Program. 1988. Tandem Computers Incorporated. Part no. 18097.

Tandem Database Group. 1987. *NonStop SQL, A Distributed, High-Performance, High-Availability Implementation of SQL*. Tandem Technical Report 87.4. Tandem Computers Incorporated.

Wakashige, D. 1989. NetBatch: Managing Batch Processing on Tandem Systems. *Tandem Systems Review*. Tandem Computers Incorporated. Part no. 18662.

Mattran, R. 1986. Buffering for Better Application Performance. *Tandem Systems Review*. Vol. 2, No. 1. Tandem Computers Incorporated. Part no. 83936.

Welsh, R. 1984. Optimizing Sequential Processing on the Tandem System. *Tandem Journal*. Vol. 2, No. 3. Tandem Computers Incorporated. Part no. 83938.

Acknowledgments

I would like to thank the many reviewers and contributors from Tandem software development, field support, customer support, and product management who assisted in this article.

Timothy Keefauver is software product manager for batch and sequential processing and Disk Process 2. He has worked in systems management, design and management consulting, econometric research, and marketing research using various systems platforms. Tim holds a B.A. in economics from The Monmouth College and an M.B.A. in business policy from The University of Chicago.

Table 1.
Summary of batch performance mechanisms. (Tear along the perforation to remove table for quick reference.)

Mechanism	ENSCRIBE	NonStop SQL	Comments
Version C10 of GUARDIAN 90			
Sequential block buffering	Pre-C10		SBB performs no record locking and does not require OPEN EXCLUSIVE. Larger block sizes (for example, 4096 bytes) means fewer messages to and from DP2 and larger I/Os.
Unstructured buffer length	Pre-C10		The default is 4 Kbytes. Use SETMODE 93 if an application is randomly reading an unstructured file and it needs less data.
Unstructured cache bypass	Pre-C10		Unstructured cache bypass can be used for reads and writes by issuing SETMODE 91, parameter 1. It allows unstructured files to bypass DP2 cache (under various limitations). This is not needed when using bulk I/O.
Bulk I/O	Pre-C10	C10	Bulk I/O refers to large, usually 28-Kbyte transfers of disk data. It is available to GUARDIAN 90 calls or to ENSCRIBE when using FUP. The necessary buffering occurs within FUP or the application. The use of bulk I/O is automatic within NonStop SQL. (Note that version C30 of NonStop SQL automatically supports 56-Kbyte bulk I/O for the 3107 and newer controllers.)
Sequential updates	Pre-C10	C10	When the modified block split described above is used, DP2 also forces "no tree walking," which can improve performance and system resource consumption.
Generic record locking	Pre-C10	C10	The SETMODE 123 command for ENSCRIBE files and the use of DDL for NonStop SQL invoke generic record locking. It applies only to the leading portion of keys. Subsequently, issue UNLOCKFILE for ENSCRIBE files.
Buffered writes	Pre-C10	C10	Use GUARDIAN 90 SETMODE or use FUP to change the file label. TMF always uses buffered writes. NonStop SQL DDL enables this for SQL.
FASTSORT parallelism	Pre-C10	C10	FASTSORT offers automated parallel execution. Parallel sort can be called by ENFORM and COBOL85 (as of version C10).
Compress audit checkpoint record	Pre-C10	C10	If SYNCDEPTH is set to 1, it causes compression of checkpoint records and audit log. Use SETMODE 94 for ENSCRIBE.
Parallel writes	Pre-C10	C10	Parallel writes are only for mirrored disks with two controllers. Use SETMODE 57 for ENSCRIBE files or CREATE or ALTER commands for NonStop SQL files. There are fault-tolerant risks associated with this procedure. However, the correct system configuration enhances throughput.
Read-through locks	Pre-C10	C10	Use SETMODE 4 for ENSCRIBE files or BROWSE for NonStop SQL files to ignore locks.
COBOL85 fast I/O	C10		This enhancement buffers COBOL85 accesses of ENSCRIBE entry-sequenced files by merely specifying clauses such as RESERVE and ACCESS MODE SEQUENTIAL, and by not issuing such clauses as LINAGE. Up to a tenfold performance increase is possible.
Modified block split	C10	C10	Fewer block splits occur when sequential inserts occur before the end of file (EOF). Therefore, fewer file reorganizations are needed. Although DP2 tries to automatically assert the appropriate insert method, SETMODE 91, parameter 3, can help when there are about five logical insertions at any one location in the file.
Inter-controlpoint flushing	C10	C10	This function is automatic. DP2 sorts cache buffers by disk address and then writes them using bulk I/O.
Retain cache at close of file	C10	C10	This function is automatic. When all users of a file CLOSE, data is written to disk, but at the same time, it remains in the cache until it ages out.
SQL SBB		C10	The CONTROL TABLE * SEQUENTIAL INSERTS ON/OFF command influences NonStop SQL to buffer reads. It works similarly to ENSCRIBE SBB but usually has better performance results. In some cases, VSBB is invoked instead. VSBB filters the transfer of selection and projection information. Locking is always complete for VSBB.
Sequential prefetch		C10	Sequential prefetch automatically provides asynchronous prereading using bulk I/O. About 28 Kbytes can be transferred with one I/O. C30 and 3107 disk controllers can transfer about 56 Kbytes of data.
Sequential postwrite		C10	Sequential postwrite automatically performs asynchronous writes to disk and uses bulk I/O when possible. See Sequential prefetch.

Mechanism	ENSCRIBE	NonStop SQL	Comments
Version C20 of GUARDIAN 90			
EXPAND	C20	C20	Acknowledgments (L4) piggyback on data that travels in the opposite direction. Especially good for X.25 communications and heavily loaded systems, it is completely automatic with GUARDIAN 90, version C20.
ODP read-only cache for optical disk process	C20	C20	Faster backup and restore processing is also supported. Structured read access is available for ENSCRIBE.
ODP serial read/write for optical disk process	C20	C20	New routines improve serial (nonrandom) accesses. Structured read access is available to ENSCRIBE.
Large software cache	C20	C20	C20 raises the 6-Mbytes-per-volume maximum to 56 Mbytes per volume.
Version C30 of GUARDIAN 90			
Online file reorganizations	C30	C30	Audited files may be reorganized without taking down OLTP applications. Reorganizing files while OLTP is running effectively offloads work from potentially scarce batch window time, thus allowing more batch processing to be performed in the window.
Parallel SQL queries		C30	Use multiple processors to execute query statements with the CONTROL EXECUTORS MULTIPLE command. The default is SINGLE.
SQL VSBB, C30		C30	Used for WHERE CURRENT commands, VSBB is automatic for C30 and operates for both reads and writes. Use the CONTROL TABLE * SEQUENTIAL INSERT ON/ENABLE/OFF to influence its use.
FASTSORT data passing		C30	FASTSORT can directly read 28 Kbytes of data in one transfer from the data input file. Note that FASTSORT began reading ENSCRIBE input files directly and with 28-Kbyte transfers starting with a release prior to C10.
Parallel index maintenance		C30	NonStop SQL tables that have multiple indexes will be updated asynchronously, which improves response time.
Index-only retrievals		C30	Reads of data that only involve columns from an index will no longer access the base table, which improves performance.
Query optimizer improvements		C30	OR, IN, ANY, SOME, and ALL operations are improved by this enhancement. See Batch declaration.
Access declaration (for host variables)		C30	The CONTROL QUERY SET INTERACTIVE ACCESS ON/OFF command encourages the optimizer to scan the base table rather than use an index. Consider using this command when a host variable is used in a predicate and the comparison operator is not an equal sign (=).
Clustering keys		C30	Through use of the CREATE TABLE command, more power is provided to partitioning and, thus, more performance. Also, greater partitioning control is available, making it more convenient to attempt various partitioning schemes.

Reminder: Consider using in an application design and implementation the following: low priority, automated scheduling, CPU selection, REFRESH OFF, disk contention leveling, large block sizes, partitioning, unstructured files, parallel application processing, COBOL85 prereading, and sequential versus random I/O.

TANDEM SYSTEMS REVIEW CUSTOMER SURVEY

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

1. How useful is each article in this issue?

An Overview of SNAX/CDF
01 ☐ Indispensible 02 ☐ Very 03 ☐ Somewhat 04 ☐ Not at all

Network Design Considerations
05 ☐ Indispensible 06 ☐ Very 07 ☐ Somewhat 08 ☐ Not at all

NonStop SQL: The Single Database Solution
09 ☐ Indispensible 10 ☐ Very 11 ☐ Somewhat 12 ☐ Not at all

Optimizing Batch Performance
13 ☐ Indispensible 14 ☐ Very 15 ☐ Somewhat 16 ☐ Not at all

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

17 ☐ Overview discussions of new products and enhancements.
18 ☐ Performance and tuning information.
19 ☐ High-level overviews on Tandem's approach to solutions.
20 ☐ Application design and customer profiles.
21 ☐ Technical discussions of product internals.
22 ☐ Other_____

3. Your title or position:

23 ☐ President, VP, Director 24 ☐ Systems analyst 25 ☐ System operator
26 ☐ MIS manager 27 ☐ Software developer 28 ☐ End-user
29 ☐ Other_____

4. Your association with Tandem:

30 ☐ Tandem customer 31 ☐ Tandem employee 32 ☐ Third-party vendor 33 ☐ Consultant
34 ☐ Other_____

5. Comments

_____

_____

_____

_____

_____

NAME

_____

COMPANY NAME

_____

ADDRESS

_____

_____

◂ 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 COMPUTERS INCORPORATED
TANDEM SYSTEMS REVIEW
LOC 216-05
19333 VALLCO PARKWAY
CUPERTINO CA 95014-9990

◂ FOLD

TANDEM PUBLICATIONS ORDER FORM

Subscriptions to the *Tandem Systems Review* are free. Use this form to subscribe, change a subscription, and order back copies.

For requests within the U.S. send this form to:

Tandem Computers Incorporated
Tandem Systems Review
18922 Forge Drive, LOC 216-05
Cupertino, CA 95014-0701

For requests *outside* the U.S., send this form to your local Tandem sales office.

Check the appropriate box(es):

- New subscription (# of copies desired _____)
- Subscription change (# of copies desired _____)
- Request for back copies. (Subject to availability.)

To order back copies, write the number of copies next to the title(s) below. Please allow six to eight weeks for delivery.

NUMBER OF COPIES	***Tandem Systems Review***
_____	Part No. 83937, Vol. 2, No. 2, June 1986
_____	Part No. 83938, Vol. 2, No. 3, December 1986
_____	Part No. 83939, Vol. 3, No. 1, March 1987
_____	Part No. 83940, Vol. 3, No. 2, August 1987
_____	Part No. 11078, Vol. 4, No. 1, February 1988
_____	Part No. 13693, Vol. 4, No. 2, July 1988
_____	Part No. 15748, Vol. 4, No. 3, October 1988
_____	Part No. 18662, Vol. 5, No. 1, April 1989

Print your current address here:

COMPANY NAME

ADDRESS

ATTENTION

JOB TITLE

PHONE NUMBER (U.S.)

If your address has changed, print the old one here:

COMPANY NAME

ADDRESS

ATTENTION

JOB TITLE

PHONE NUMBER (U.S.)

TANDEM EMPLOYEES: ORDER YOUR COPIES THROUGH COURIER OR YOUR MARKETING LITERATURE COORDINATOR.

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

BILL EASTWOOD
DEPT 1831
LOC NUM 4-15
CUPERTINO CA CORPORATE HQ

Part No. 28152

400107 9/89 Printed in USA