K/DSRD-419

MARTIN MARIETTA

APPLIED TECHNOLOGY

PACIFIC MISSILE TEST CENTER
INFORMATION RESOURCES MANAGEMENT
ORGANIZATION (CODE 0300)

THE *ORACLE CLIENT-SERVER* AND DISTRIBUTED PROCESSING ARCHITECTURE

A. L. Beckwith
J. T. Phillips

OPERATED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

DISCLAIMER

K/DSRD--419

DE90 013271

PACIFIC MISSILE TEST CENTER
INFORMATION RESOURCES MANAGEMENT ORGANIZATION
(CODE 0300)
THE *ORACLE CLIENT-SERVER* AND
DISTRIBUTED PROCESSING ARCHITECTURE

A. L. Beckwith
J. T. Phillips

June 10, 1990

Sponsored by the
Pacific Missile Test Center
Point Mugu, California 93042
Under Interagency Agreement 1714-1714-A1

Prepared by
Data Systems Engineering Organization
Data Systems Research and Development Program

Located at
OAK RIDGE GASEOUS DIFFUSION PLANT
Oak Ridge, Tennessee 37831-7129
Operated by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
Under Contract No. DE-AC05-84OR21400

DO NOT MICROFILM
THIS PAGE

TABLE OF CONTENTS

ACKNOWLEDGMENTS

The following products and names are registered trademarks of their respective companies.

3Com Etherlink is a trademark of 3Com Corporation.

COMPAQ 386 Deskpro is a trademark of COMPAQ.

dBASE is a trademark of Ashton-Tate Corporation.

Emerald Bay is a trademark of Ratliff Software Production Inc.

Ethernet is a trademark of the XEROX Corporation.

INFORMIX-SQL is a trademark of INFORMIX Software, Inc.

Ingres Version 6.3 Database Server is a trademark of Relational Technologies, Inc.

MicroVAX/VMS, *DECnet*, *DECnet-DOS*, *MicroVAX II/VMS*, *VT100*, *VAX*, *VMS*, and *DEC* are trademarks of the Digital Equipment Corporation.

MS-DOS is a trademark of Microsoft Corporation.

NCR 286 is a trademark of the NCR Corporation.

NetWare SQL is a trademark of Novell Inc.

ORACLE, *SQL*NET*, *ORACLE PC TOOL PACK*, *ORACLE PC Tools*, *SQL*Connect*, *ORACLE Server*, *SQL*PLUS*, *SQL*FORMS*, *SQL*MENU*, *SQL*ReportWriter*, *Professional ORACLE*, *RPT*, *SQL*LOADER*, and *Import/Export Utilities* are trademarks of the Oracle Corporation.

Paradox is a trademark of Borland International, Inc.

Rbase is a trademark of Microrim, Inc.

SQL Base is a trademark of Gupta Technologies.

Systems Application Architecture, *SAA*, IBM, *OS/2*, and *IBM PC/AT* are trademarks of International Business Machines.

XDB Server is a trademark of XDB Corporation.

ABSTRACT

Computing architectures using distributed processing and distributed databases are increasingly becoming considered acceptable solutions for advanced data processing systems. This is occurring even though there is still considerable professional debate as to what "truly" distributed computing actually is and despite the relative lack of advanced relational database management software (RDBMS) capable of meeting database and system integrity requirements for developing reliable integrated systems. This study investigates the functionality of *ORACLE* database management software that is performing distributed processing between a *MicroVAX/VMS* minicomputer and three *MS-DOS*-based microcomputers. The *ORACLE* database resides on the *MicroVAX* and is accessed from the microcomputers with *ORACLE SQL*NET*, *DECnet*, and *ORACLE PC TOOL PACKS*. Data gathered during the study reveals that there is a demonstrable decrease in CPU demand on the *MicroVAX*, due to "distributed processing", when the *ORACLE PC Tools* are used to access the database as opposed to database access from "dumb" terminals. Also discovered were several hardware/software constraints that must be considered in implementing various software modules. The results of the study indicate that this distributed data processing architecture is becoming sufficiently mature, reliable, and should be considered for developing applications that reduce processing on central hosts.

INTRODUCTION TO DISTRIBUTED COMPUTING ARCHITECTURES

"Distributed" computer systems have been accepted as being desireable and feasible for many years. Much discussion of this trend began in the late 1970's and early 1980's.[1,2,3,4,5] During those years, the decreasing cost of processors reversed an early trend in data processing toward large centralized computer systems. The original assumption (Grosch's Law) had been that "the cost per machine instruction executed was inversely proportional to the square of the size of the machine", thereby leading to an economy of scale for large centralized computers in data processing centers. However, it slowly became accepted that for many applications, minicomputer and microcomputer architectures offered more cost effective solutions. In addition to lower initial systems costs, smaller systems offered a potentially higher availability (less down time), simpler software support requirements, and lower installation, maintenance, and training requirements.[6]

The growth of multiple data processing sites within companies and the cooperative use of enterprise information systems between companies has required organizations to develop strategies for ensuring a positive impact of distributed data processing. The impact of distributed processing on system users and the nature of required organizational changes calls for planning implementation as an evolutionary process rather that allowing end-users to drive the development of corporate information systems in an uncoordinated fashion from the bottom up. Issues such as data ownership, privacy, security, and auditability must be addressed as well as the challenges that occur in decentralized applications development where standards are needed for corporate information systems consistency. The requirements of distributed systems for data from central systems must also be planned and monitored, including both distributed data "owned" locally and requested data that is simply processed locally [7]

All of these concerns are still true today, and top management faces increasing challenges in deciding how much and where distributed processing is appropriate in their organizations.[8] Because of the effort required to migrate large centralized systems to distributed processing, many organizations are starting with the migration of mini-computer based systems with compromise interim solutions that attempt to combine keeping the "critically important business applications on the host and farming out end-user applications to the desktop."[9]

Despite an acknowledgement that distributed systems offer many challenges and opportunities, it is only in the last few years that full-featured software has appeared as development platforms for integrated distributed systems. Some early systems often used "intelligent terminals" that allowed some "function distribution" such as data entry without participating in complete transaction processing. Other systems used networks or hierarchies of scattered "integrated" processors that completed parts or all of some transactions. No specific developmental approaches, database standards, or communications protocols were commonly accepted or in place for developing distributed data processing systems and software had not been developed taking into consideration the unique requirements of distributed systems[10].

DISTRIBUTED DATABASE

Distributed computing systems can take advantage of distributed data, distributed processing or both facilities. Although the subject of this investigation is distributed processing, some discussion of the issues in distributed data will help clarify the boundaries of distributed processing and the requirements for interaction between systems.

> A distributed database system consists of a collection of sites (also called nodes), connected together via some kind of communication network, in which ... each site is a database system site in it's own right, but ... the sites have agreed to work together..., so that a user at any site can access data in the network exactly as if the data were all stored at the user's own site...[11]

Although there are undoubtedly no ideal distributed database systems in existence today, the overall objective is to achieve a "seamless" operation while adhering to principles of operation such as the following goals for data and systems operations:[12]

1. Local autonomy
2. No reliance on a central site
3. Continuous operation
4. Location independence
5. Fragmentation independence
6. Replication independence
7. Distributed query processing
8. Distributed transaction management
9. Hardware independence
10. Operating system independence
11. Network independence
12. DBMS independence"

All of these issues have an impact on the functionality, performance, and reliability of both distributed databases and distributed processing. Major objectives for distributed database systems are location transparency, data fragmentation, fragmentation transparency, and data replication.[13] Although an explanation of all of these issues is beyond the scope of this investigation, some of the issues do directly impact an evaluation of distributed processing. They are the issues of data fragmentation, data replication, data location, and the overall issue of system transparency, because of their involvement in the manipulation and alteration of data during the processing of database updates.

DISTRIBUTED PROCESSING

Any system that divides the data processing workload among several processors or computer systems is performing distributed processing (often called cooperative processing). Parallel processing can be considered a form of tightly coupled distributed processing with the processors all in close geographic proximity (on the same machine). File-sharing systems on Local Area Networks (LANs) allow individual users to access programs and data stored on a remote LAN file server. Client-server architectures split the front-end user interface processes from the back-end request and updates of the database. Some distributed transaction systems replicate applications or break large applications down into smaller independent components to offload processing to smaller processors.[14]

Whereas, the term "distributed database" obviously means the distribution of actual data, the exact activities performed in a "distributed processing" manner are not always as clear. Data entry, screen handling, communications interfaces, database updates, and applications development may all be occurring on different processors. Some computer networks divide the overall system processing between CPUs on different machines, but the specific implementation varies widely. The importance of networking and communications protocols and architecture for implementing distributed processing has been known for some time.[15]

A major impact on distributed processing is the cost of data communications. Different computer systems often use different communications protocols. The communications media and distance between systems can affect performance and system responsiveness drastically. For this reason, many existing distributed and networked computer systems are unique implementations for specific business challenges, that have been inappropriate as general solutions for other settings or environments.

A more generic approach to one type of distributed processing is beginning to arise. It is a type of cooperative processing in which some of the processing originally performed on a central mainframe computer is actually performed on a workstation, microcomputer or intelligent terminal. This "client-server" cooperative working relationship between the user's interface machine and the host computer is finding increasing acceptance in many environments. Until recently, a lack of standard interfaces or software made the development of such solutions often cost prohibitive. As the computing industry changes, the client-server architecture is expected to become the norm.

This study is an investigation of the implementation of distributed processing **without** distribution of the actual database data. The *ORACLE* data actually resides on a *MicroVAX* that is accessed by a microcomputer performing some of the data processing. As will be discussed in technical detail later, the microcomputer software must have knowledge of the data location, data fragmentation, data replication, and the communications environment necessary to access and process the data in a distributed manner on the microcomputer.

IMPACT OF THE RELATIONAL MODEL, SQL, AND COMMUNICATIONS

A major software industry change that has supported the development of distributed systems is the growing acceptance of the relational data model and Structured Query Language (SQL) as an access interface. Contemporary thought is that the relational model will continue to be refined in software implementations with faster performance and additional functionality. Many non-relational database management systems are being re-engineered to accept SQL interfaces, and SQL is being refined by the American National Standards Institute to provide a standard data access language.[16]

As an example, the ANSI X3H2 committee is presently completing the SQL2 standard draft, which is expected to include referential integrity.

> Referential integrity involves making certain that any database object that references another database object must be valid .... [*ORACLE*] plans to make referential integrity a part of its upcoming Version 7.[17]

The importance of development of standards became very evident with the formation of the SQL Access Group. This vendor association includes Informix Software, Inc., *ORACLE* Corporation, Relational Technology Inc., Ashton-Tate, and other software and hardware vendors that are working to develop a standard version of SQL, a standard network protocol, and a standard call-level application protocol interface (API) to SQL that would allow developers to write applications that could access data from various databases. Sybase Inc. offered it's own product's technology for utilization as a standard, while offering to license other vendors with it's technology now, potentially establishing a defacto standard. Some users are working on their own interoperability, while IBM expects to use its *Systems Application Architecture* (*SAA*) for eventual micro to mainframe compatibility. IBM has at present only "succeeded in implementing limited mainframe-to-mainframe database interoperability and workstation database communication."[18]

As data and processes become distributed, mechanisms that enforce consistency increase system reliability and performance. The overall goal is one of enhanced operability between heterogeneous vendor environments. This strongly impacts the future of distributed processing, as it is increasingly to each users advantage to be able to share and access data from remote as well as local applications. In addition to access languages and database integrity concerns, there is a lack of a communications standards for data base management systems. However, while *ORACLE* is using *SQL*Connect*, other vendors have their own implementation. As can be seen from this study, a standard SQL (or other access language) and a standard communications interface will be required for complete transparency between heterogeneous systems.[19]

An acceptance of standards in SQL, interfaces, and communications protocols is generating increasing interest in cooperative processing applications. However, much of the technology required to transition users and developers to creating distributed processing applications are still unavailable or implemented in a piecemeal fashion. Sophisticated LAN managers, operating systems (such as *OS/2*), and graphical user interfaces (GUI) are just

now being offered, but there are still few products to aid programmers. Considerable training of staff will be necessary for those programmers to achieve productivity with the new tools. Existing software such as wordprocessors and program editors do not always offer full functionality in *OS/2*, for instance, and *dBASE* programmers will need to understand SQL and data dictionaries that they may not have used previously. Many of the presently offered products are simply front ends to mainframe products.[20,21] It is expected that distributed processing applications will become very widely accepted and installed however, over the next two or three years.

CLIENT-SERVER ARCHITECTURES

The microcomputer client-server or database server seems to presently be the most active area of development and technological advancement for distributed processing systems. Most microcomputer database software today -- e.g., *dBASE*, *Rbase*, *Paradox*, etc. -- are simply file server databases. True database servers differ from file servers in that "a file-server system sends out to PCs not only copies of programs but also whole data files, [whereas] a database server decreases network load by centralizing record manipulation at the server. The database server sends only the data needed by the client application."[22] The database server works with several client microcomputers in a multitasking mode (under *OS/2*, a DOS LAN, or other operating systems) to provide clients requested data and also update the database. Components of the client-server architecture are usually:[23]

1) "The database server or engine, residing on a dedicated computer or the network file-server;

2) The client program, an application or development tool, running on a PC;

3) A layer of communication software on each end of the process."

This architecture is usually supported by SQL as an access language, because it accompanies the relational database that these systems usually support. Relational systems support the separation of the data from the applications thus making it easier to separate the front-end client application processes from the data oriented database processes. Not all vendors are supporting this approach as can be seen with Ratliff Software Production Inc.'s *Emerald Bay* database management software, which supports a record and pointer oriented methodology for data management. Most vendors, however, are supporting SQL based implementations.

Present major vendors developing and offering SQL client-server architectures are:[23,24,25,26,27,28]

Microsoft/Sybase/Ashton-Tate - *SQL Server*

Oracle - *ORACLE Server*

Gupta Technologies - *SQL Base*

IBM - *OS/2 Extended Edition Database Manager*

Ingres Corporation - *Ingres Version 6.3 Database Server*

Novell Inc. - *NetWare SQL*

XDB - *XDB Server*

Informix - *Informix-SQL*

The database server architecture is thus a compromise between older centralized database systems on mainframes and stand alone microcomputer systems. It combines the centralized/shared data architecture of a mainframe system (on the database server often using a microcomputer) with a microcomputer (client) handling its own screen and keyboard I/O in a front-end application. Database processing, including updates, original query processing, and transaction logging, occurs on the database server. Screen handling, some data validation, keyboard I/O, database requests, occur on the microcomputer client. Requests for data are developed on the client and sent out over a network to a server that processes the request and sends the answer/data back to the client. Thus it is a compromise solution that spreads the processing out over the whole system.[29]

The advantages of this architecture can be several. By splitting the processing, both the client and the server can be developed, enhanced, and optimized to concentrate on their specific activity. An individual server can support more users, because the client workstations perform user interface activities. Data is more available to any given user, because it is only locked during the few moments of access. Workstations with different hardware, software and operating systems can be used to access data from a single powerful server. Users can be free to select and develop their own interfaces, while database administration and security remain under centralized control.[30]

As might be expected, there are some disadvantages to this architecture. Client-server systems can be expected to be complex to set up; as the server, the client and the communication interface each have their own requirements for administration and knowledge. They are heavily dependent on the network/communications system used, and should generally not be used for applications that would put severe loads on the communication system. "The best case for distributing data [or processing] is when there are few I/O's per input; for example, a single complex SQL call. The worst case is when hundreds of records are accessed per transaction."[31] Communications overhead can become a central issue in system performance. This became apparent in this study.

ORACLE OVERVIEW

ORACLE Corporation has developed software products that allow an *ORACLE* tool set or module to reside on a microcomputer, while the data used in the execution of the application is retrieved from an *ORACLE* RDBMS kernel residing on a different machine that could perhaps be located miles away. *ORACLE PC Tools* communicate from the microcomputer over a network through a proprietary communications protocol and accesses the *ORACLE* data on a microcomputer, minicomputer, or mainframe host. This technique is an example of the client-server architecture and it enables distributed databases and distributed processing to be used. This discussion will focus on distributed processing.

ORACLE's distributed products become more sophisticated and permit many additional enhancements with multitasking operating systems such as *OS/2*. However, the test bed used to examine and discuss distributed processing in this study was built with *MS-DOS* and a *MicroVAX II*. The *MS-DOS* operating system limits its capabilities to a single client application and single database kernel. This means that a *MS-DOS* machine cannot be both a server and a client at the same time. The database server can, however, have multiple *MS-DOS* users accessing it at the same time over a communications network.

The *ORACLE* distributed software includes the following modules:

1. A physical communication medium such as *Ethernet*;

2. An industry-standard protocol for the server and each client, in this case *DECnet* and *DECnet-DOS*;

3. *SQL*NET* for the server machine and for each *MS-DOS* client machine; and

4. The *ORACLE* RDBMS kernel for the server.

In addition each microcomputer must have a compatible communications board, in this test *3Com Etherlink* boards. Figure 1 presents product version numbers and a diagram of the test bed environment. Note that the test environment did not include local databases on the microcomputer clients because extended memory would be required to run this local database and a major interest was in the performance of the product below a 640K DOS memory limit. The following is a description of the individual *ORACLE* product modules used to achieve and study distributed processing.

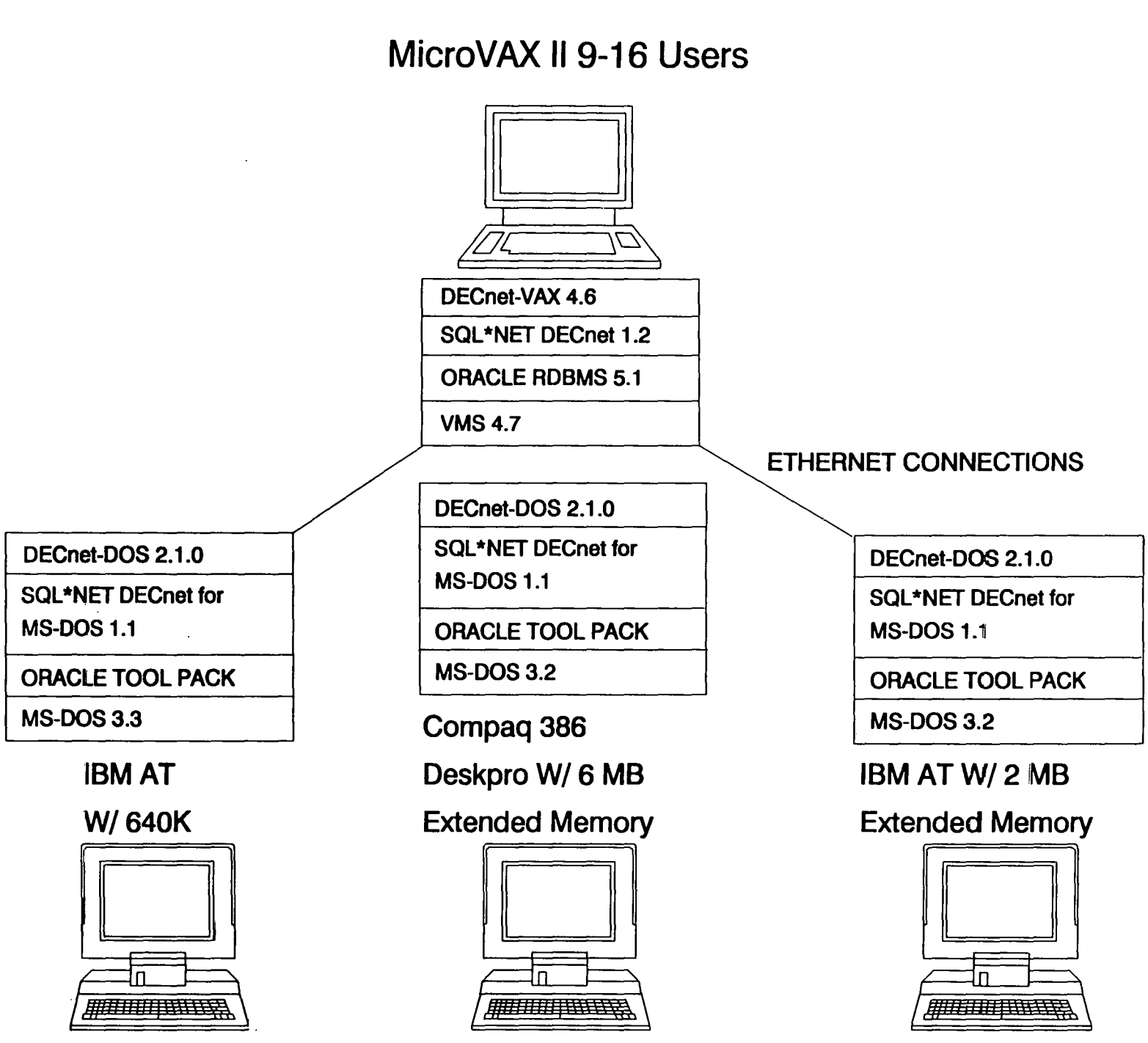

Fig. 1. Hardware/software environment.

ORACLE SYSTEM MODULES

The *ORACLE* Relational Database Management System (RDBMS) works as an interface between the actual storage of the data and the user to provide a view of the data to the user. The RDBMS allows sophisticated, versatile, and an easy method for handling information with powerful control over the data stored. The RDBMS with capability for associating data between tables, permits data relationships that appear to the user as if his requested data came from one large datafile instead of combining from several tables. Flexibility in information retrieval is increased and this allows many groups to share the same data rather than have many datafiles or independent databases housing redundant data. System maintenance, concurrency, and integrity issues suddenly become less costly. *ORACLE* is considered to be a full featured implementation of the relational database model.

Data manipulation is accomplished with the *SQL*PLUS* product. *ORACLE* uses American National Standards Institute (ANSI) Standard Query Language (SQL) as the standard relational access language between the user and the kernel. *ORACLE* also has added SQL extensions (commands) to enable easier requesting and manipulating of data in the RDBMS. The advantage of SQL and *SQL*PLUS* is that many data records can be returned to the user with one command line in a comparatively short amount of time, especially when using *ORACLE*'s indexing. The bulky procedural coding and long programming hours used by older query methods are no longer required to produce reports and review data.

The *SQL*FORMS* module allows a user to retrieve and manipulate data with ease using full screen forms. Using nonprocedural methods, simple data manipulation screens are created using function keys, or the user can use the full screen painter to customize and detail screens. *SQL*FORMS* enables quick applications building for the system designer as well as a user-friendly interface to the database for the non-expert operator.

To integrate *SQL*FORMS* applications together and allow access from a main module or program, *ORACLE* created *SQL*MENU*. Just as the RDBMS is the central module of the applications, *SQL*MENU* is the heart of all the data manipulation applications. *SQL*MENU* provides a menu that allows a user to access any of the developed applications instead of having the separate applications execute individually.

To allow easy and fast production of sophisticated reports without procedural code, the *ORACLE SQL*ReportWriter* is used. The non-procedural, menu-driven product allows ease in producing good report formats. A user wanting to do ad hoc reporting must have some knowledge of *SQL*PLUS*, since as with all *ORACLE* products, the *ReportWriter* relies on SQL and *SQL*PLUS* to extract the data from the database.

All of the above *ORACLE* products, with the exception of the RDBMS kernel make up the *ORACLE TOOL PACK* that each microcomputer must have for distributed processing to take place. This software package resides on the microcomputer client. No longer are software tools housed on the same machine as the data base management system. The database server only hosts and maintains the database kernel.

The heart of *ORACLE* distributed processing is the *SQL*NET* product. *SQL*NET* permits the user to access data on a remote machine(s) as if the data resides on the user's microcomputer workstation as a local database. The product's main concern while being executed is to connect the client application to *ORACLE* database on the server side. *SQL*NET* is referred to as an open system since it is independent of industry communication protocols, such as *DECnet*. It conforms or operates with many protocols. Like the other *ORACLE* software tools, *SQL*NET* is portable across various environments. *SQL*NET* is comprised of two sections: the generic layer and the custom layer. The generic layer is common to all protocols, and the custom layer is tailored to the specifically required protocol and operating system environment. *SQL*NET* acts as the interpreter for the *ORACLE* request. The microcomputer client-PC-generated data request is passed from *SQL*NET* to *DECnet-DOS* which performs its process to process communication and sends the data request to the server *DECnet* module. Here *SQL*NET* evaluates the command and interfaces with the RDBMS. The answer to the command is returned to the client workstation. Security can be an issue of concern, because, although the local database data is secure, the data passed across the network lines are not. Later versions of the distributed product plan to incorporate data encryption across the network lines. The *ORACLE* tools such as *SQL*FORMS* and *SQL*PLUS* support *SQL*NET* in that an entrance to the database kernel is permitted with the invoking of the product. Included in the command line for executing the product are the *ORACLE* database username and password, the server's *ORACLE* account username and password, the filename declaring the remote database's name, and the server machine's node number. This cumbersome command line will be alleviated in later versions. At this time, aliases and database server machine names are not permitted.

ORACLE has taken into consideration the International Standards Organization's (ISO) seven layer model of communication, Open Systems Interconnect (OSI), in developing their distributed products. This model creates an environment standard that allows one vendor's product to talk to another vendor's products. This permits users eventually to be able to retrieve data from many kinds of databases. Most of the current industry protocols, do not match the OSI model, since many protocols were developed before the standards were created; however, protocols such as *DECnet* are moving their architecture to the OSI model. When *SQL*NET* is referred to as an open system, it reflects being compatible with the OSI.

THE COMMUNICATIONS ENVIRONMENT

A brief description of the in-house LAN used in the distributed test bed is necessary to give the reader a clear understanding of the distributed environment studied. The architecture of the LAN is one of computing resources resident on widely separated LANs connected via the intra-plant broadband. Figure 2 demonstrates the setup description. Degradation was experienced in data retrieval when the data request went through the baseband to broadband bridges. This bridge is limited to relatively small packet throughput rates and is sensitive to network loading which causes transmission delays. The bridge throughput limitation is the probable cause for varied times in executing a database query.

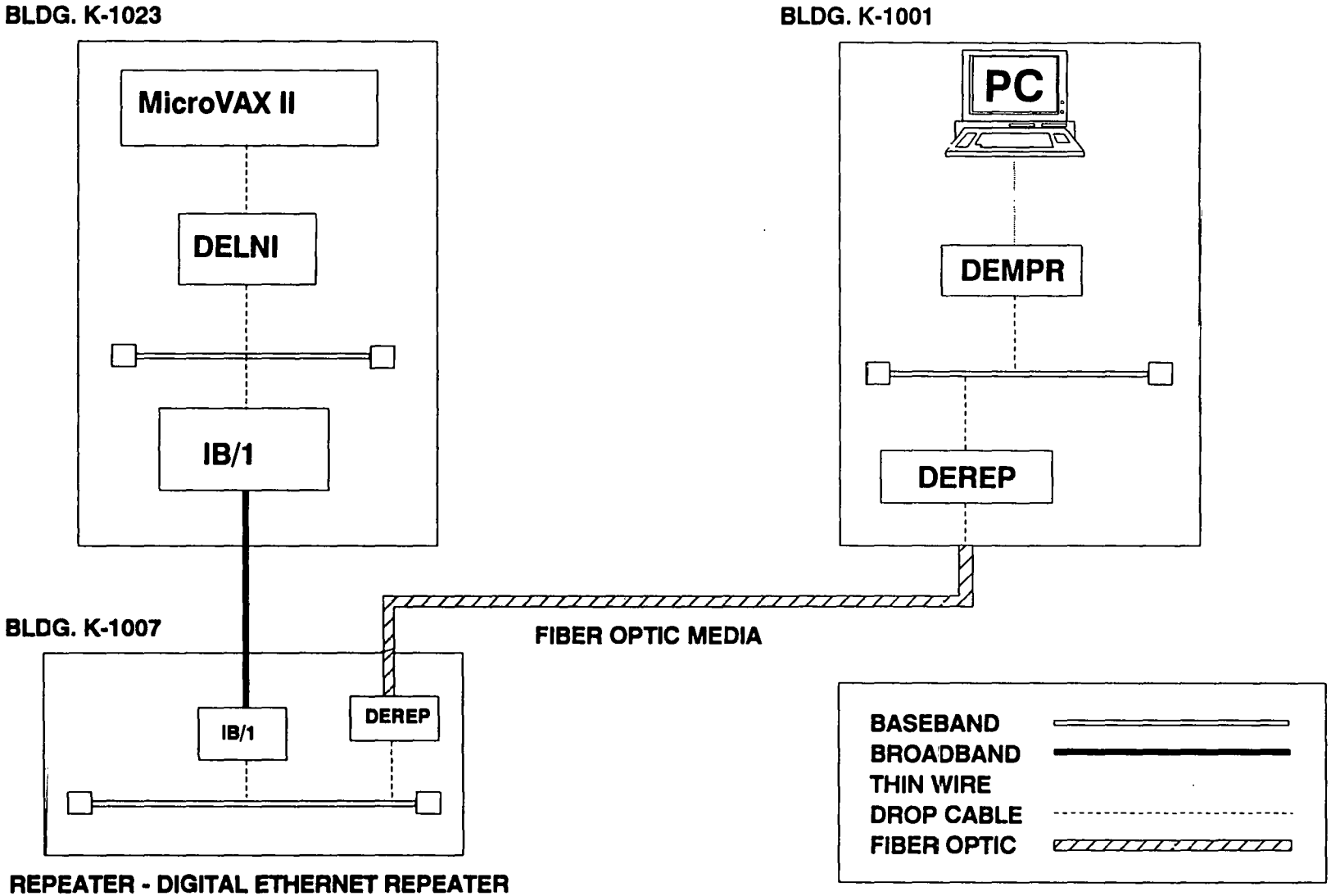

Fig. 2. The communications environment of Test Bed.

IMPLEMENTING *ORACLE*'S DISTRIBUTED PROCESSING

The distributed *ORACLE* environment was setup with a *MicroVAX II/VMS* and three client *MS-DOS* microcomputers. It should be stated that the distributed software was tested over a network that experiences moderately heavy user traffic. Also, there was not a local database on any of the client microcomputers. The *MicroVAX II* was to be used solely as a remote database for the developed applications on each client workstation.

It should also be noted that the installation of the software, particularly the *TOOL PACKS* on the client microcomputers have certain restrictions. During *TOOL PACK* installation, a program entitled **MACHTYPE** is executed. This program requires the user performing the installation to declare the type of microcomputer that will be used as a client. *ORACLE* customizes their *TOOL PACK* to run on certain types of hardware. Just because a machine is *IBM PC*-compatible, does not mean that choosing one of the IBM-compatible models will allow the software to successfully run. The *TOOL PACK* installation was initially attempted on a *NCR 286* machine which is *IBM PC*-compatible. *ORACLE* does not support the NCR machines so the software would not successfully load. The installation appeared to be successful, but the *SQL*NET* drivers were unable to be loaded into memory, which is required for the remote database to be accessed through *SQL*NET* and *DECnet*. After discovering the NCR machines could not be used, they were replaced with two *IBM PC/ATs* and a *COMPAQ 386 Deskpro*. These machines are included in *ORACLE*'s list of supported hardware.

The user installing the software should also make note that *SQL*FORMS* developed in a large machine environment cannot be ported to another smaller computer platform and execute properly without the proper preparation. *ORACLE* recommended that certain procedures be executed to insure the applications would run properly on the microcomputer. First, one needs to make sure the **VT100.CRT** file is copied from the *SQL*FORMS* Installation Disk 2 into the **ORACLE5\DBS** directory on the microcomputer regardless of whether the microcomputer is the target or source system in the transferring process. This file is not installed during installation because the microcomputer cannot use it to run a form. Second, if a database exists on a source system, all the forms being transferred must be saved in the database of the source system. Third, the forms on the source system need to be converted to *VT100* format. Use the Interactive Application Converter (IAC) command to translate the forms, in this case on the *VAX*, to *VT100* format. The command is executed as follows:

```
IAC -c VT100.CRT inputfilename formnameinDB username/password
```

It is very important to note that the input filename that will contain the form formatted to *VT100* format, should be different from the form name in the database; otherwise, the form in the database will be written over. Proceed by performing a binary file transfer for each file to be downloaded. Once all files have been ported to the target system, the forms need to be saved in the database of that target system, if one exists. The IAC command with the "-i" option loads the application forms into the target database.[32]

The *5.1B application TOOL PACK* includes the newer versions of *SQL*FORMS* and *SQL*ReportWriter*. If there is not a local database, the database to be accessed is referred to as remote. To allow development capability with *SQL*FORMS* and *SQL*ReportWriter*, the system tables required for application and report development need to be installed or in this case upgraded to allow the new versions to execute properly. To accomplish the upgrade, use *SQL*PLUS* to connect to the remote database and execute the appropriate SQL scripts indicated in the user's manuals.

A file must be created on the server machine that identifies or names the database. This file should be defined per instructions in the *SQL*NET* manual for *DECnet* on the server machine. This file is of great importance in that *SQL*NET* looks for this file to access the database. This file is referenced by name in the command line when executing a tool from the client microcomputer. *SQL*NET* looks for this same filename on the server machine to grant the user access to the correct database. An example command line to connect to the database is shown below.

```
SQL*PLUS "ORACLE USERID/PASSWORD@D:SERVER NODE NUMBER\"VMS USERNAME VMS
        PASSWORD\"::\"TASK=DATABASE FILENAME\""
```

The backslashes preceding the quotation marks in the command string are to prevent *MS-DOS* from stripping the quotes before passing the string to the program. *VMS* requires that quotes be a part of the command line so that the request can be executed.[33]

TEST RESULTS

Once these procedures were completed, testing began on the retrieval and manipulating of data in the remote database via *SQL*NET* and *DECnet*. It appears that multi-users trying to query the same record or records can do so easily without interfering with the other user's data requests. In the case of updates and deletes to the database, the user whose process retrieves the record first, issues a lock on that row in the database. That lock stays in effect until his transaction is completed. In the meantime, the users that try to retrieve this locked row or are viewing this record to also perform an update have no manipulation capability. The user's microcomputer appears to be locked up. A message will appear at the bottom of the screen display telling the user that this record is being updated and he should requery to receive the new status of the record. The record lock does allow other users to query other records in the same table as the updated record while the update occurs.

Measuring the time it took for the client workstation to connect to the remote database was difficult, since traffic on the network varied greatly. Observations did reveal that the initial connection time was not exceptionally fast, but once into an *ORACLE* product, such as *SQL*PLUS* or executing a *SQL*FORM*, data retrieval time was just as good as that on a workstation with a *Professional ORACLE* local database or being a terminal tied directly into the *MicroVAX II*. The accessing of a remote database was totally transparent to the client workstation.

The testing environment contained two kinds of microcomputers. The *DESKPRO* and one *IBM PC/AT* were equipped with extended memory. The other *IBM PC/AT* had only 640K of memory. There are differences and restrictions in the capabilities of the *ORACLE* modules depending on available memory.

The microcomputers with extended memory allowed *SQL*FORMS* and *SQL*ReportWriter* development as well as execution. It should be noted that the *SQL*ReportWriter* can only be installed and used in protected mode. This means that at least one megabyte (1MB) of extended memory is required on the microcomputer. With extended memory, all of *ORACLE* products included in the *TOOL PACK* are accessible. In dealing with a remote database and no local database, a user should remember that a new *SQL*FORM* or a report should always be generated before running against remote the database. This is to ensure that every user executes the latest version of a company-wide report. When a report program is updated, the new generated version is saved only to the database and to the user's workstation who made the changes. This can be an advantage to microcomputer workstation clients in that a user can store personal report versions on their own workstation.

The microcomputer without extended memory presented quite a few limitations. First of all, the *SQL*ReportWriter* cannot even be loaded because protected mode is not accessible without extended memory. Therefore, the older version procedural *RPT* report writer or *SQL*PLUS* will have to be used to generate any reports. These two products can only produce considerably less sophisticated reports than the *SQL*ReportWriter*. Without

extended memory, the development of *SQL*FORMS* is impossible. There is a size limitation on a form running under 640K of memory. *SQL*MENU* development also requires extended memory, so this capability is unavailable. The other *ORACLE* products such as *SQL*LOADER* and the *Import/Export Utilities* will execute under 640K.

Possibly the greatest advantage to distributed processing is that while the workstation and the remote server are talking to one another, the data request (query) is not recorded or logged in as an user with an account as a dumb terminal user would be. Using the monitor commands available with the *VMS* operating system, only the transaction or manipulation appears on the screen when it is executed. For example, the BWR process which handles all write operations to the database appears for a split second when data is added, deleted, or updated. This is extremely valuable in that much less CPU is required since the host machine no longer has to manage the execution of the software products, allocate space for each user to be productive, as well as control data retrieval. The host machine or server has only to concentrate on sending the data to the users requesting it and control the locking of data for concurrency and integrity.

CONCLUSIONS

The functionality of the *ORACLE* software when operating in a distributed processing client-server architecture worked according to expectations, with some limitations. The modules did work in an identical manner on both the microcomputer workstations and *MicroVAX* minicomputer hardware platforms, except for expected variances between installations due to differences between the *MS-DOS* and *DEC VMS* operating systems.

Applications could be developed on the microcomputer workstation with a local copy of *SQL*FORMS* and ported to the *MicroVAX* to function completely in that environment. They could also be executed from the microcomputer workstation to access the *MicroVAX* database. It was not necessary to have a local database on the microcomputer, as would be the case when developing forms with the *Professional ORACLE* microcomputer product. However, the limitations of some modules to work under the 640K DOS memory, became evident even here, as only small forms could be developed on the microcomputer. More memory was needed for large forms.

This became more evident with the *SQL*ReportWriter* module, which would not run under 640K at all. Only the older *RPT* reportwriter could be used, which is a much less "friendly" product to users. *SQL*Plus* could also be used, but does not contain extensive report formatting capabilities. *SQL*Loader* and the *Import/Export Utilities* can also run under 640K, but are of little help in using or developing applications. This means that for most serious applications development or use, extended memory microcomputers will probably be necessary.

Distributing the screen and forms handling, keyboard I/O, data validation, query development, and applications development to the workstation are major advantages of this architecture, if reduction of processing on the central host machine are desired. In addition to the advantages of preserving database administration of a central "master" database in a centrally managed manner, users can be offered computing tools for developing their own applications, while possibly reducing an expectation of an immediate need for new hardware for the central host computer. The prototype as tested confirms this as a reasonable expectation.

The installation of this software product on two different operating systems (*VMS* and DOS) along with the installation of LAN-based communications network software and drivers is very complex. It demands a thorough knowledge of both the *ORACLE* products and the telecommunications environment that will serve as the vital link between workstations and the host. Performance monitoring of this client-server will require attention to the communications environment, as well as the individual software modules. *ORACLE* software users, applications developers, database administrators, and telecommunications managers will all have to work together for the Client-Server Architecture to be able to deliver its full potential.

REFERENCES

1. S. Ceri and G. Pelagatti, *Distributed Databases: Principles and Systems*, McGraw-Hill Publishing Company, NY, 1984.

2. J. Akoka and P.P.S. Chen, "Optimal Design of Distributed Information Systems", *IEEE Transactions on Computers* **C29**(12), 1068-90 (December, 1980).

3. A. L. Scherr, "Distributed Data Processing," *IBM Systems Journal* **17**(4), 324 (November 4, 1978).

4. C. K. Davis and J. C. Wetherbe, "An Analysis of the Impact of Distributed Processing on Organizations in the 1980's", *MIS Quarterly* **3**(4), 47-56 (1979).

5. J. P. Buchanan and R. G. Linowes, "Understanding Distributed Data Processing", *Harvard Business Review* **58**(4), 143-52 (July-August, 1980).

6. J. Martin, pp. 5, 9 in *Design and Strategy for Distributed Data Processing*, Prentice-Hall Inc., Englewood Cliffs, NJ, 1981.

7. Ref. 7, pp. 9-17.

8. R. G. Crepeau and J. R. Weitzel, "A Manager's Guide to Distributed Data Processing", *Journal of Systems Management* **40**(9), 17-21, (September, 1989).

9. C. Sivula, "Cooperative Processing - The Client-Server Perspective", *Datamation* **35**(19), 47-8 (October 1, 1989).

10. Ref. 7, pp. 87-104.

11. C. J. Date, "What is a Distributed Database", *InfoDB* **2**(2), 3 (Summer 1987).

12. Ref. 12, p. 4.

13. C. J. Date, pp. 589-90 in *An Introduction to Database Systems, Volume 1, 4th edition*, Addison-Wesley Publishing Company, 1986.

14. G. Wai, "Take Your Pick", *BYTE* **14**(7), 215-23 (July, 1989).

15. J. Martin, pp. 36-64 in *Computer Networks and Distributed Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

16. H. Edelstein, "Look for Relational to be 1990's Data Model", *Software Magazine* **10**(1), 59-60 (January, 1990).

17. R. Carnahan, "Richer SQL Poised on the Horizon," *Systems Integration* **23**(1), 23 (January, 1990).

18. J. Soat, S. Leibs, and K. Myers, "The Race for Open Databases", *Information Week* **241**, 12-3 (October 16, 1989).

19. Ref. 17, pp. 59-60.

20. S. Mace, "Distributed Data: Sizing up the Challenge", *InfoWorld* **11**(31), 39-41 (July 31, 1989).

21. R. Francis, "Cooperative Processing, The PC Perspective", *Datamation* **35**(18), 63-6 (September 15, 1989).

22. R. Letson, "Competition Heats Up in Database Servers", *Systems Integration* **23**(1), 32-7 (January, 1990).

23. D. R. Vinzant, "SQL Database Servers", *DataCommunications* **19**(1), 72-85 (January, 1990).

24. K. Watterson, "Soul of a Machine - SQL Server", *Data Based Advisor* **7**(8), 80-4 (August, 1989).

25. K. Watterson, "Getting Under the Hood of SQL Server, Part 1", *Data Based Advisor* **8**(1), 56-67 (January, 1990).

26. K. Watterson, "Getting Under the Hood of SQL Server, Part 2", *Data Based Advisor* **8**(2), 102-12 (February, 1990).

27. N. Petreley, Z. Banapour and L. Slovik, "Dueling Servers", *INFOWORLD* **12**(10), 57-75 (March 5, 1990).

28. K. Watterson, "DB Connections - The System Administrator", *Data Based Advisor* **8**(3), 84-98 (March, 1990).

29. M. L. Van Name and W. Catchings, "Serving Up Data", *BYTE* **14**(9), 259-64 (September, 1989).

30. Ref. 15, pp. 215-23.

31. J. Gantz, "Client/Server Computing Offers Hidden Problems, No Panacea", *InfoWorld* **11**(36), 36 (September 4, 1989).

32. B. Humphrey, "SQL*FORMS Application Transfer," *Oracle Corporation's Customer Support Newsletter: Technical Alerts*, S-2 (October/November, 1989).

33. "DECnet - Database ID String," *Oracle Corporation's Customer Support Newsletter: Technical Alerts*, 5-1 (October/November, 1989).

SELECTED BIBLIOGRAPHY

J. T. Perry and J. G. Lateer, *Understanding ORACLE*, Sybex Corporation, San Francisco, CA, 1989.

*The New Questions and Answers about SQL*STAR, Revision 2*, Oracle Corporation, Belmont, CA, May 1988.

INTERNAL DISTRIBUTION

1. Abercrombie, R. K.
2. Abercrombie, E. F.
3. Arnold, H. G.
4. Ashdown, B. G.
5-9. Beckwith, A. L.
10. Boling, M. E.
11. Elrod, M. H.
12. Green, P. L.
13. Gillespie, S. J.
14. Halsey, P. J.
15. Hammons, C. E.
16. Handler, B. H.
17. Hicks, S. E.
18. Huntley, Jr., A. F.
19. Johnson, G. L.
20. Kegley, W. P.
21. Kidd, G. V.
22. Kimmerly, W. C.
23. Kirk, P.
24. Leinius, R. P.
25. Light, K. L.
26. Loebl, A. S.
27. Lumley, J. J.
28. Lundberg, L. A.
29. Pennewell, W. J.
30-39. Phillips, J. T.
40. Popa, G. L.
41. Ragland, W. R.
42. Ruple, S. L.
43. Sexton, F. L.
44. Shelton, J. D.
45. Smith, A. A.
46. Smith, K. J.
47. Streetman, K. D.
48. Taylor, R. W.
49. Thomas, Jr., B.
50. Tubbs, C. S.
51. Vickers, B. D.
52. Webber, L. S.
53. Wood, W. B.
54. Merriman, J. R.
55. ORGDP Plant Records
56. DSRD Resource Center
57. Applied Technology Library

EXTERNAL DISTRIBUTION

58. R. Blackburn, Code 0320.1, Pacific Missile Test Center, Point Mugu, CA 93042.
59. D. Kral, Code 0321, Pacific Missile Test Center, Point Mugu, CA 93042.
60. Office of Assistant Manager for Energy Research and Development, Department of Energy, Oak Ridge Operations Office, Oak Ridge, TN 37831