z/OS

IBM

Network File System
Performance Tuning Guide

z/OS

**Network File System
Performance Tuning Guide**

Third Edition, December 2001

Contents

Figures

Tables

About this book

This book provides system programmers and operators with methods to assess the performance of their NFS environment, determine their z/OS NFS performance requirements, and evaluate the trade-offs between cost and performance. It also provides examples of methods to evaluate the performance of z/OS NFS.

In this book, NFS client or NFS server refers to NFS client or NFS server in general, which includes z/OS NFS. When necessary, z/OS will be specifically specified.

Required product knowledge

To use this book effectively, you should be familiar with IBM MVS operating system, IBM Time Sharing Option (TSO), and their commands. In addition, you should be familiar with the basic concepts of the NFS protocol and networking (Transmission Control Protocol/Internet Protocol (TCP/IP)), as well as the NFS client operating systems accessing the z/OS NFS.

Referenced publications

The following publications have additional information:

Table 1. Additional information

Publication Title	Order Number
AIX for RISC System/6000 Performance Monitoring and Tuning Guide	SC23-2365
AIX Operating System TCP/IP User's Guide	SC23-2309
AIX Commands Reference for RISC System/6000, Volume 1	GC23-2376
AIX Commands Reference for RISC System/6000, Volume 2	GC23-2366
AIX Commands Reference for RISC System/6000, Volume 3	GC23-2367
AIX Commands Reference for RISC System/6000, Volume 4	GC23-2393
AIX Communications Concepts and Procedures for IBM RISC System/6000	GC23-2203
Character Data Representation Architecture Reference and Registry	SC09-2190
Character Data Representation Architecture Overview	GC09-2207
DFSMS/MVS Online Product Library	LK2T-8731
IBM Online Library: AIX Base Collection Kit	SK2T-2166
IBM Online Library Omnibus Edition: MVS Collection	SK27-0710
IBM Networking Softcopy Collection Kit	SK2T-6012
Introducing IBM's Transmission Control Protocol/Internet Protocol Products for OS/2®, MVS, VM	GC31-6080
OS/390 Support for Unicode™: Using Conversion Services	SC33-7050
SMP/E R8.1 Reference	SC28-1107
TCP/IP: Performance Tuning Guide	SC31-7188
TCP/IP for MVS: Programmer's Reference	SC31-7135
TCP/IP for MVS: Customization and Administration Guide	SC31-7134
TCP/IP for MVS: User's Guide	SC31-7136

Table 1. Additional information (continued)

Publication Title	Order Number
z/OS Communications Server IP Configuration Reference	SC31-8776
z/OS DFSMSdfp Advanced Services	SC26-7400
z/OS DFSMSdfp Diagnosis Reference	GY27-7618
z/OS DFSMSdfp Storage Administration Reference	SC26-7402
z/OS DFSMShsm Diagnosis Guide	LY35-0114
z/OS MVS Programming: Authorized Assembler Services Guide	SA22-7608
z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN ALE-DYN	SA22-7609
z/OS MVS Programming: Authorized Assembler Services Reference ENF-IXG ENF-IXG	SA22-7610
z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU LLA-SDU	SA22-7611
z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO SET-WTO	SA22-7612
z/OS DFSMS Introduction	SC26-7397
z/OS DFSMS Macro Instructions for Data Sets	SC26-7408
z/OS DFSMS Migration	GC26-7398
z/OS DFSMS: Using Data Sets	SC26-7410
z/OS MVS System Management Facilities (SMF)	SA22-7630
z/OS MVS Installation Exits	SA22-7593
z/OS MVS JCL Reference	SA22-7597
z/OS MVS System Messages, Vol 1 (ABA-ASA)	SA22-7631
z/OS MVS System Messages, Vol 2 (ASB-EWX)	SA22-7632
z/OS MVS System Messages, Vol 3 (GDE-IEB)	SA22-7633
z/OS MVS System Messages, Vol 4 (IEC-IFD)	SA22-7634
z/OS MVS System Messages, Vol 5 (IGD-IZP)	SA22-7635
z/OS MVS Programming: Authorized Assembler Services Guide	SA22-7608
z/OS Network File System Customization and Operation	SC26-7417
z/OS Network File System User's Guide	SC26-7419
z/OS TSO/E User's Guide	SA22-7794
z/OS UNIX System Services File System Interface Reference	SA22-7808
z/OS Information Roadmap	SA22-7500
z/OS UNIX System Services Messages and Codes	SA22-7807
z/OS SMP/E User's Guide	SA22-7773
z/OS UNIX System Services Programming: Assembler Callable Services Reference	SA22-7803
z/OS UNIX System Services Programming Tools	SA22-7805
z/OS UNIX System Services Command Reference	SA22-7802
z/OS UNIX System Services User's Guide	SA22-7801
z/OS UNIX System Services Planning	GA22-7800

Using LookAt to look up message explanations

LookAt is an online facility that allows you to look up explanations for z/OS messages, system abends, and some codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can access LookAt from the Internet at:

`http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/lookat.html`

or from anywhere in z/OS where you can access a TSO command line (for example, TSO prompt, ISPF, z/OS UNIX System Services running OMVS).

To find a message explanation on the Internet, go to the LookAt Web site and simply enter the message identifier (for example, `IAT1836` or `IAT*`). You can select a specific release to narrow your search. You can also download code from the *IBM Online Library Omnibus Edition z/OS Collection*, SK2T-6700 and the LookAt Web site so you can access LookAt from a PalmPilot (Palm VIIx suggested).

To use LookAt as a TSO command, you must have LookAt installed on your host system. You can obtain the LookAt code for TSO from a disk on your *IBM Online Library Omnibus Edition z/OS Collection*, SK2T-6700 or from the LookAt Web site. To obtain the code from the LookAt Web site, do the following:

1. Go to http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/lookat.html.
2. Click the **News** button.
3. Scroll to **Download LookAt Code for TSO and VM**.
4. Click the ftp link, which will take you to a list of operating systems. Select the appropriate operating system. Then select the appropriate release.
5. Find the **lookat.me** file and follow its detailed instructions.

To find a message explanation from a TSO command line, simply enter: **lookat** *message-id*. LookAt will display the message explanation for the message requested.

Note: Some messages have information in more than one book. For example, IEC192I has routing and descriptor codes listed in *z/OS MVS Routing and Descriptor Codes*. For such messages, LookAt prompts you to choose which book to open.

Accessing z/OS books on the internet

In addition to making softcopy books available on CD-ROM, IBM provides access to unlicensed z/OS softcopy books on the Internet. To find z/OS books on the Internet, first go to the z/OS home page:

`http://www.ibm.com/servers/eserver/zseries/zos/`

From this Web site, you can link directly to the z/OS softcopy books by selecting the **Library** icon. You can also link to IBM Direct to order hardcopy books.

Accessing licensed books on the web

z/OS licensed documentation in PDF format is available on the Internet at the IBM Resource Link™ Web site at:

`http://www.ibm.com/servers/resourcelink`

Licensed books are available only to customers with a z/OS license. Access to these books requires an IBM Resource Link Web userid and password, and a key code. With your z/OS order you received a memo that includes this key code.

To obtain your IBM Resource Link Web userid and password, log on to:

`http://www.ibm.com/servers/resourcelink`

To register for access to the z/OS licensed books follow these steps:

1. Log on to Resource Link using your Resource Link userid and password, follow these steps:
2. Click on **User Profiles** located on the left-hand navigation bar.
3. Click on **Access Profile.**
4. Click on **Request Access to Licensed books.**
5. Supply your key code where requested and click on the **Submit** button.

If you supplied the correct key code you will receive confirmation that your request is being processed. After your request is processed you will receive an e-mail confirmation.

Note: You cannot access the z/OS licensed books unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

To access the licensed books, follow these steps:

1. Log on to Resource Link using your Resource Link userid and password.
2. Click on **Library**.
3. Click on **zSeries**.
4. Click on **Software**.
5. Click on **z/OS**.
6. Access the licensed book by selecting the appropriate element.

How to send your comments

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this book or other NFS documentation, you can select one of the following options:

- Visit our home page at:

  `http://www.ibm.com/servers/eserver/zseries/zos/nfs/`

  From this Web site you can click on the **Contact Us** icon to enter and submit your comments, or click on the **Library** icon to access NFS books.

- Send your comments about this book, other NFS books, or any DFSMS documentation by e-mail to:
  - IBMLink from US: starpubs@us.ibm.com
  - IBMLink from Canada: STARPUBS at TORIBM
  - IBM Mail Exchange: USIB3VVD at IBMMAIL
  - Internet: starpubs@us.ibm.com

  Be sure to include the name of the book, the part number of the book, version and product name, and if applicable, the specific location of the text you are commenting on (for example, a page number or a table number).

- Fill out one of the forms at the back of this book and return it by mail or by giving it to an IBM representative. If the form has been removed, address your comments to IBM Corporation, RCF Processing Department M86/050, 5600 Cottle Road, San Jose, California 95193-0001, U.S.A.

Related protocol specifications

IBM is committed to industry standards. The internet protocol suite is still evolving through Requests for Comments (RFC). New protocols are being designed and implemented by researchers, and are brought to the attention of the internet community in the form of RFCs. Some of these are so useful that they become a recommended protocol. That is, all future implementations for TCP/IP are recommended to implement this particular function or protocol. These become the *de facto* standards on which the TCP/IP protocol suite is built.

The NFS is implemented as a set of RPC procedures that use External Data Representation (XDR) encoding to pass arguments between clients and servers. The NFS is based on the following RFCs:

Internet Protocol	RFC 791, J.B. Postel
NFS: Network File System Version 2 Protocol Specification	RFC 1094, Sun Microsystems, Incorporated
NFS: Network File System Version 3 Protocol Specification	RFC 1813, Sun Microsystems, Incorporated
Open Group Technical Standard Protocols for Interworking: XNFS, Version 3W	Document Number: C702
RPC: Remote Procedure Call Protocol Specification Version 2	RFC 1057, Sun Microsystems Incorporated
RPC: Remote Procedure Call Protocol Specification Version 2	RFC 1831, R. Srinivasan
User Datagram Protocol	RFC 768, J.B. Postel
WebNFS Client Specification	RFC 2054, B. Callaghan
WebNFS Server Specification	RFC 2055, B. Callaghan
XDR: External Data Representation Standard	RFC 1014, Sun Microsystems, Incorporated
XDR: External Data Representation Standard	RFC 1832, R. Srinivasan

For more information about Request for Comments (RFC), see the Internet Engineering Task Force (IETF) home page:

`http://www.ietf.org/`

RFC documentation is available at the Internet Engineering Task Force (IETF) Web site:

`http://www.ietf.org/rfc.html`

Summary of changes

The summary of changes informs you of changes to this book. Revision bars (|) in the left margin of the book indicate changes from the previous edition.

Summary of changes for SC26-7418-02, z/OS Version 1 Release 2

This book contains information that was previously presented in *z/OS Network File System Performance Tuning Guide*, SC26-7418-01. The following sections summarize the changes to that information.

New information

This edition includes new information applicable to this release.

- Reference documentation; see Table 1 on page ix.

Changed information

This edition includes information about locating messages; see "Using LookAt to look up message explanations" on page xi.

Moved information

This edition does not include moved information.

Deleted information

This edition does not delete any information.

This book contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Starting with z/OS V1R2, you may notice changes in the style and structure of some content in this book. For example, headings that use uppercase for the first letter of initial words only, and text that has a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our books.

Summary of changes for SC26-7418-01, z/OS Version 1 Release 2

Attention: This was a major revision of the Network File System (NFS) features in OS/390® (5647-A01 (V2R6, V2R7, V2R8, V2R9, V2R10)) and z/OS (5694-A01 (V1R1, V1R2)); it provides information for NFS features that are applicable to these releases.

This book contains information that was previously presented in *z/OS Network File System Performance Tuning Guide*, SC26-7418-00. The following sections summarize the changes to that information.

New information

This edition includes the following new information:
- The Network File System (NFS) features; see "Chapter 1. Introduction" on page 1.
- Avoiding storage shortages; see "Storage considerations" on page 29.
- Parameter description; see "DynamicSizeAdj" on page 31.

- Comment requests; see "Related protocol specifications" on page xiii.
- New glossary terms; see "Glossary" on page 37.

Changed information

This edition includes the following changed information:

- Performance tuning symptom and action information; see "Network performance tuning" on page 11.
- *nfstasks* site attribute information; see "Subtasking" on page 29.
- As part of the name change of OpenEdition® to z/OS UNIX® System Services, occurrences of z/OS OpenEdition have been changed to z/OS UNIX System Services or its abbreviated name, z/OS UNIX. OpenEdition may continue to appear in messages, panel text, and other code with z/OS UNIX System Services.

Moved information

This edition does not include moved information.

Deleted information

This edition deleted the User datagram protocol (UDP) information from "MVS constraints" on page 14.

This book contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.

Summary of changes for SC26-7418-00, Version 1 Release 1

This book applies to Version 1 Release 1 of z/OS and contains information that was previously presented in *OS/390 Network File System Performance Tuning Guide*, SC26-7255.

Chapter 1. Introduction

This chapter provides an overview of the remaining chapters and the possible ways to tune the performance of the IBM z/OS Network File System (z/OS NFS). It also briefly explains the client-server relationship. The client-server relationship and related terminology are provided to clarify references that might be misunderstood and to establish term usage within the context of this document.

Purpose and structure

This book is geared towards z/OS V1R2 Network File System (V1R2 NFS). Most of the information presented in this book is relevant to previous releases. Information pertaining only to z/OS V1R2 NFS will be marked with parameters where appropriate.

This book contains information that is organized into the following chapters:

Description	Page
Chapter 2. Performance tuning in the NFS environment–Explains performance tuning within the context of the NFS Client-Server environment.	5
Chapter 3. Optimizing the NFS environment–Explores areas that may impact overall performance within the NFS environment.	11
Chapter 4. Evaluating z/OS NFS performance–Explains the techniques to evaluate the performance of z/OS NFS.	17
Chapter 5. Tuning the z/OS NFS server–Explains tuning considerations specific to the z/OS NFS server.	21
Chapter 6. Tuning the z/OS NFS client–Explains tuning considerations specific to the z/OS NFS client.	31

Chapters 2 though 4 provide a brief overview of performance tuning from a total system perspective. Chapters 5 and 6 discuss the performance tuning at the client and server level.

Client-server relationship and terminology

This section is intended to provide a basic understanding of the NFS client-server relationship and the terminology used in later sections. Terminology specific to a limited area will be introduced as necessary. More details about the NFS can be found in *z/OS Network File System User's Guide* and *z/OS Network File System Customization and Operation*.

A *client* is a computer or process that requests services on the network. A *server* is a computer or process that responds to a request for service from a client. A *user* accesses a service, which allows the use of data or other resources.

Figure 1 on page 2 illustrates the client-server relationship. The upper right portion of the figure shows the z/OS NFS server. The lower right portion of the figure shows the z/OS NFS client. The left portion of the figure shows various NFS clients and servers which can interact with the z/OS NFS server and client. The center of the figure shows the Transmission Control Protocol/Internet Protocol (TCP/IP) network used to communicate between the clients and servers.

Directory List (ls -l)

Command Response   Time Comparison

| DFSMS-managed | Non-managed |
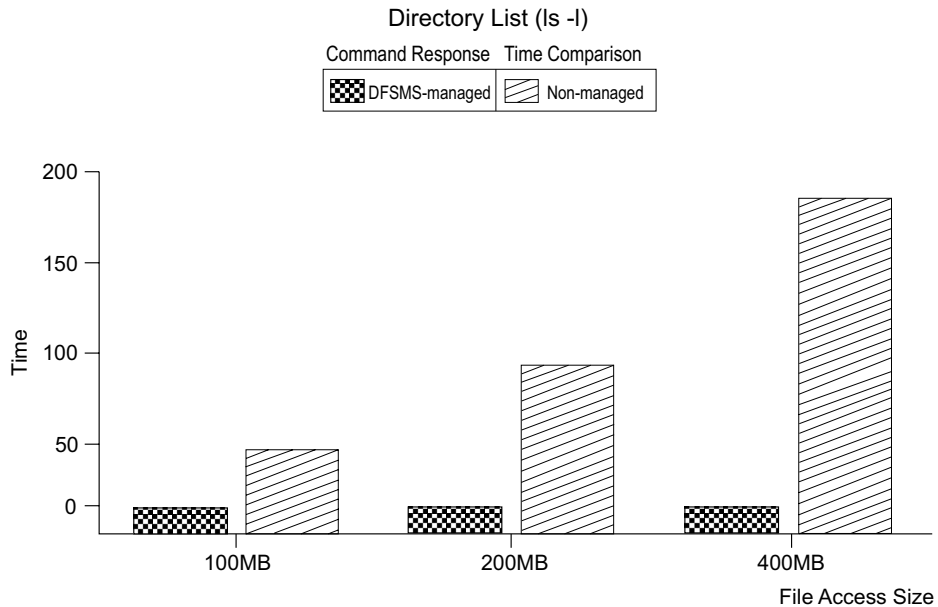
Figure 1. z/OS NFS client-server relationship

With the z/OS NFS server, you can remotely access MVS/ESA™ conventional data sets or z/OS UNIX files from workstations, personal computers, and other systems that run client software for the Sun NFS version 2 protocols, the Sun NFS version 3 protocols, and the WebNFS protocols over TCP/IP network.

The z/OS NFS server acts as an intermediary to read, write, create or delete z/OS UNIX files and MVS data sets that are maintained on an MVS host system. The remote MVS data sets or z/OS UNIX files are mounted from the host processor to appear as local directories and files on the client system. This server makes the strengths of an MVS host processor–storage management, high-performance disk storage, security, and centralized data–available to the client platforms.

With the z/OS NFS client you can allow basic sequential access method (BSAM), queued sequential access method (QSAM), virtual storage access method (VSAM), and z/OS UNIX users and applications transparent access to data on systems which support the Sun NFS Version 2 protocols and the Sun NFS Version 3 protocols. The remote NFS Server can be an MVS, UNIX, AIX®, or other system. The z/OS NFS client is implemented on z/OS UNIX and implements the client portion of the Sun NFS Version 2 protocols and the Sun NFS Version 3 Protocols.

To access the NFS server a **mount** command must be executed on the NFS client system to provide a temporary link (until unmounted) between a local directory (preferably empty), or unused logical drive, and a remote file identifier available on the NFS server. The exact syntax and file naming conventions will depend on the NFS client and server implementations in use. With z/OS NFS, the data set high level qualifier(s) (HLQ) would be used to access conventional MVS data. The hierarchical file system (HFS) path name, preceded by a special configurable keyword, would be used to access z/OS UNIX files.

In addition to providing a link between NFS client and server, the **mount** command also has parameters associated with it. Some of these parameters, such as buffer sizes and timing values, can affect performance. This topic will be discussed further in Section "NFS client system performance tuning" on page 12. Since the defaults for these parameters vary from one implementation to another, specific details

should be obtained from the appropriate client documentation. With the z/OS NFS implementation there are opportunities to override the MVS file allocation parameters and processing options on the **mount** command as well. Client enablement code is provided with the z/OS NFS product for many NFS client implementations. File allocation and processing options will be further discussed in Sections "Data set creation attributes" on page 21 and "Processing attributes" on page 23.

Chapter 2. Performance tuning in the NFS environment

This chapter explains performance tuning within the context of the NFS client-server environment. It provides guidelines on the performance expectation of the underlying hardware and software products on which an NFS client-server implementation is dependent. These guidelines specifically note the limitations to performance tuning and describe processes to tune z/OS NFS.

What is performance tuning?

In general, performance tuning improves the price to performance ratio for a system or set of services by reallocating the available computing, network, or storage resources. The reallocation of these resources not only improves the performance for a particular load of work, but also accommodates an increase in the amount of work to be performed with minimal acquisition of additional resources. The information acquired from performance tuning can also be an important basis for long range capacity planning.

Some possible reasons to do performance tuning might be to:
- Access more data over existing networks
- Improve response time for particular applications or groups of users
- Better utilize the available storage capacity
- Minimize cost for additional services or functional capability

How is performance characterized?

Performance is the manner in which a process, system, processor, network, or device behaves for a particular load or unit of work. To measure, or quantify, performance, we monitor the length of time for a unit of work to complete. If units of work are being shared, we monitor the amount of time waiting for a resource to be available to perform a unit of work. A unit of work is a specific activity or action that we expect a process, system, processor, network, or device to perform. This could be something as granular as an I/O request, sending or receiving a buffer of data over a network, or processing an NFS request. We are frequently interested in the performance of a particular set of work activities which we will refer to as a load of work, or workload.

When a particular workload has been identified for performance measurement, we can determine the performance metrics, or units of measurement, that are relevant to that workload. Some examples of the performance metrics that will be used in reference to performance tuning for z/OS NFS are:

throughput	Number of units of work completed per unit of time.
response time	Elapsed time necessary to complete a unit of work.
CPU time	Amount of time the processor spends executing instructions.
instructions per byte	Total number of CPU instructions to process data divided by the number of bytes of data processed.
aggregate throughput	Sum of the throughput measurements for multiple processes.
NFS operations per second (NFSops)	Number of NFS requests processed by an NFS server divided by the total time in seconds to process the requests.

Measuring performance metrics can be as simple as using your watch to time the execution of a particular command or as complex as using specialized hardware

and software tools to monitor and extract a diversity of performance metrics. "Chapter 4. Evaluating z/OS NFS performance" on page 17 will address some methods that can be used to evaluate the performance of z/OS NFS. The method selected will depend on the complexity of the workload and the monitoring tools available at your installation.

What is the NFS environment?

The NFS environment includes the NFS client system(s), the mix of networks available, the NFS server system(s), and the manner in which they are configured. While this guide is intended for z/OS NFS, it is important to know the performance limitations within the NFS environment to determine the necessity of tuning z/OS NFS.

NFS client systems range from single user desktop personal computers to large scale processors with many users. These NFS client systems typically support multiple applications as well. Clearly, the NFS client system resources will be shared between its users and/or applications. These resources include available physical storage, memory, processing capability, and network access. The NFS client is one application, with a possibility for many users, that must share the memory, processing, and network resources in exchange for providing access to additional physical storage on other systems. The degree to which the NFS client application must share such resources will affect performance for NFS client users and applications.

The NFS server system, like the NFS client system, must share resources with other users and applications. NFS server application performance will be affected by the amount of contention over system resources and by the priority established for the NFS server application. The overall performance of an NFS server is also influenced by the number of NFS clients for which it provides services.

The network(s) over which NFS clients access NFS servers also affect overall performance in the NFS environment. Such networks can be homogenous, consisting of a single network medium, or heterogeneous, consisting of a mix of network mediums. Each network medium type has an expected maximum capacity, or *bandwidth*. For instance, the capacity of a Fast Ethernet Ring network may be 16 megabits per second (Mbps) or 4 Mbps, and the capacity of a Fiber Distributed Data Interface (FDDI) network is 100 Mbps. When different network mediums are combined in a more complex network environment, the capacity for a fixed route over the network is limited by the network segment with the smallest capacity. For example, a route over a network consisting of both 4 and 16 Mbps Token Rings and a FDDI backbone will have a maximum capacity equivalent to that of the Token Ring, or 4 Mbps. When bridges, routers, and gateways are included in a network configuration, their capacity must also be considered. Such devices must also be considered when tuning performance in a network environment, particularly if a device does not support increased network buffer sizes.

How to tune for performance

Given the complexity of the NFS environment, it is important to establish a methodology for tuning performance. The following steps provide a guideline that highlights particular areas relevant to the NFS environment. Implementing the guideline may involve more than one person or support organization.

Identify Performance Requirements: Before you begin performance tuning in general, and particularly z/OS NFS, determine those areas where performance is unsatisfactory. This is a good time to establish more precise performance

requirements. As users and application requirements are identified, it can be advantageous to rank or group them according to their requirements.

Know the NFS Environment: In the previous section, the NFS environment was discussed. It is very important to fully understand the performance of the existing NFS environment, particularly that of the network. Such analysis can eliminate unnecessary tuning of the NFS client and server systems.

Establish Performance Objectives: Once the performance requirements and the NFS environment are known, you are in a position to define and prioritize your performance objectives. The performance objectives should be specified in a manner that is quantifiable and measurable. Keep in mind that you will need an executable workload or test scenario to evaluate the effectiveness of your performance tuning.

Define Workloads and Test Scenarios: You may already have workloads or test scenarios depending on how requirements and/or objectives were defined. However, these may be unwieldy or impractical to use for your performance tuning purposes. Therefore, it is advantageous to spend some time initially to define some simplified test cases that can be executed in a repeatable fashion and with as much control as is feasible. Simple test cases are also useful in diagnosing performance problems, particularly to locate an area within the NFS environment that may be impacting performance.

Select Monitoring Tools: While you may only be interested in the performance of z/OS NFS, you will find it useful to have access and familiarity with a variety of performance hardware and software monitoring tools. Not only will such knowledge assist you in collecting data to evaluate performance, but it will also help you to identify areas that may be impacting the performance of z/OS NFS. Minimally, a set of monitoring tools must be identified to collect the data upon which performance tuning decisions will be made and to determine the effectiveness of tuning.

Collect Performance Data: At this point you should be ready to begin collecting performance data. Initial measurements will be the starting point, or *baseline*, that will be used to evaluate the effect of your tuning. Since there can be a significant variation in network and system performance, it is prudent to repeat a performance measurement to establish the degree of variation inherent in the measurement. Doing this will provide a sense of whether or not future tuning is really affecting performance or simply normal variation.

While it may be convenient to collect performance data when systems and networks are idle, this is probably not practical. However, it is useful to collect data during peak and low activity periods. If user or application requirements are related to a specific time period, or sensitive to other system and network activity, data should be collected for these periods of time, as well. Remember that redistributing the workload may be the most cost effective approach to performance tuning.

Evaluate Performance Data: This may seem like an obvious step. However, keep in mind that the NFS environment may be quite complex. The more complex the NFS environment and your test cases are, the more data there will be to evaluate. You may also have both client and server data to evaluate as well as network data.

As you begin evaluating performance data you have collected, look for areas or opportunities where performance could be improved. Before attempting to tune z/OS NFS, you should determine if performance is primarily impacted by the NFS server system, the NFS client system, or the network itself. In fact, you may

determine that additional data must be collected prior to any performance tuning. You may also discover evidence that configurations and parameter settings within the evaluation environment are not optimal.

The output of this step is a list of changes that you believe will positively affect performance. As you make this list, identify the impact or cost associated with a change. It is also useful to identify any resources that are heavily utilized or that have contending requirements. This will help you to prioritize and ultimately select the changes you will make or recommend.

Tune Your NFS Environment: Make one change from your list of possible tuning changes at a time. Measure and evaluate the effect of that change before making any other changes. Pay particular attention to any impact on heavily utilized resources that may have been previously identified in addition to newly exposed resources that now appear to be impacted. Since performance tuning typically involves a trade off in resource utilization, make sure you have not inadvertently caused a performance problem elsewhere. Also, before deciding to implement a change, consider whether any observed changes are due to the tuning change you've made or simply a result of normal measurement variation. Repeat this step as necessary.

Impact of the NFS protocol on performance

Command response time is of particular importance to NFS client users. The longer a user waits for the results of a particular command, the more important this will become. The nature of transparent access with the NFS protocol results in users not necessarily being aware of the impact on performance caused by the network and the NFS server system.

Also, while users are generally knowledgeable of the commands supported by the NFS client operating system, they may have no knowledge of the NFS commands, or procedures, that are executed as a result of one simple user command. In fact, one user command typically results in execution of multiple NFS commands.

Another impact on command response time is the NFS protocol itself which is intended to be as stateless as possible. This means that a stateless server operates correctly without maintaining any protocol state information for its clients. A stateless protocol was originally selected to minimize the probability of data losses due to a server crash. The z/OS NFS strives to be as stateless as possible.

The stateless nature of the NFS server places the responsibility of keeping track of NFS commands on the NFS client. To do this NFS client implementations generally wait a period of time for a response to a particular NFS command. If a response is not received within this period of time, the NFS client will retransmit the NFS command. This process is repeated for a fixed number of times or until a response is received.

NFS servers and clients have typical methods of queuing requests and responses as part of the underlying protocol layers. When these queues are full, incoming requests or responses are dropped. The NFS client and server do not know when responses or requests are dropped. Both rely on the stateless protocol whereby the client will eventually retransmit the NFS request again. Under these conditions the NFS client is waiting for a response that will never be received. Clearly, waiting to retransmit an NFS command, particularly multiple times, will negatively affect the response time for the initial user command.

To determine whether or not NFS client users are being impacted by the situation previously described, most NFS client implementations provide a **nfsstat** command to monitor NFS statistics. Figure 2 shows the output from the **nfsstat -c** command. When using the **nfsstat** command, users should be aware that results are

```
USER1:/u/user1:>nfsstat -c

Client rpc:
calls            badcalls           retrans          timeout
51               0                  0                0

Client nfs_V2:
calls        badcalls
45           0
null         getattr      setattr        root         lookup
0       0%   21     47%   1        2%    0       0%    13          29%
readlink     read         writecache     write        create
0       0%   0      0%    0        0%    0       0%    1           2%
remove       rename       link           symlink      mkdir
4       9%   0      0%    0        0%    0       0%    0           0%
rmdir        readdir      fsstat
0       0%   3      7%    2        4%


Client nfs_V3:

calls           badcalls
4               0
null         getattr      setattr        lookup       access
0       0%   0      0%    0        0%    0       0%    0           0%
readlink     read         write          create       mkdir
0       0%   0      0%    0        0%    0       0%    0           0%
symlink      mknod        remove         rmdir        rename
0       0%   0      0%    0        0%    0       0%    0           0%
link         readdir      readdirplus    fsstat       fsinfo
0       0%   0      0%    0        0%    2       50%   1           25%
pathconf     commit
1       25%  0      0%
```

Figure 2. Displaying NFS client rpc and NFS statistical information

cumulative. These statistics may be reset with the *-z* option of the **nfsstat** command. Also, the **nfsstat -c** command provides statistics for all NFS client activity, which makes it possible to access files on more than one NFS server. When using this command to query the NFS client statistics for z/OS NFS, make sure that all NFS client access is for z/OS NFS only.

The **nfsstat -c** command provides two types of statistics, *Client rpc* and *Client nfs*. The *Client rpc* statistics provide an indication of NFS performance from the perspective of that particular NFS client. In general, if the *timeout* value is more than 0.2% of the total number of *rpc* calls, then some performance tuning is necessary.

If the *timeout* value is essentially equivalent to the *retrans* value, the NFS client is waiting on the NFS server. While you can increase the *timeout* option of the **mount** command to reduce the number of retransmissions, this will not improve the perceived responsiveness of the NFS server.

The *badcall* and *badxid* statistics are generated when information is lost or dropped somewhere between the NFS client and the NFS server. This can happen when

processes are interrupted and are not necessarily indicative of a performance problem unless they are disproportionately high or persist. On the NFS client system the *netstat -s* command provides additional statistics that may be useful in tuning at the network level. For the z/OS NFS, the MVS TCP/IP **netstat** command provides information that may be useful for tuning as well.

The client NFS statistics provide a distribution of the NFS procedure calls made by users and applications on that particular NFS client. For a typical NFS client you will probably see a larger percentage of *getattr* and *lookup* calls. These calls are made whenever an NFS file is initially accessed. Directory listing is a common user activity that will generate these NFS calls.

The write and read statistics can also provide insight into the manner of NFS access on an NFS client, such as read or write biases. If the percentage of read and write NFS calls is higher than the percentage of getattr and lookup NFS calls, the client is probably accessing relatively large files as opposed to accessing many smaller files. Conversely, if the percentage of getattr and lookup NFS calls is higher than the percentage of read and write NFS calls, you might want to investigate whether users are querying directories or accessing relatively small files. In the latter case you might further determine if such files should be stored on an NFS server or the NFS client system.

Chapter 3. Optimizing the NFS environment

This chapter explores areas that may impact the overall performance within an NFS environment. The focus will be on various network and NFS client and server system parameters which may affect NFS performance. Specific product documentation should be consulted for additional detail.

Network performance tuning

Table 2 contains network performance tuning information.

Table 2. Network performance tuning symptom and action information

| Symptom	Action
Data transfer rates significantly less than network capacity	**ACTION 1:** *Know your network topology* - "What is the NFS environment?" on page 6 briefly introduces the role of the network in the NFS environment. Clearly, transferring data over congested networks results in poor performance. Therefore, when a performance problem is encountered in the NFS environment, it is useful to determine whether or not the network is the source of the problem prior to investigating other alternatives. While there are products specifically designed to monitor and report on network activity, knowing your particular network topology may be sufficient for initial tuning. Figure 3 shows a simple network topology with two 100 Mbps Fast Ethernet Rings connected to a Gbps Fast Ethernet backbone network. In this example, access from NFS client A, on a 100 Mbps Fast Ethernet Ring, to NFS server B, directly connected to the Gbps Fast Ethernet backbone, would be as limited by network bandwidth as access to NFS server A. On the other hand, access from NFS client B, directly connected to the Gbps Fast Ethernet backbone, would be limited by the load on the Gbps Fast Ethernet network and the capacity of NFS server B.

NFS client A

NFS server A

100 Mbps Fast Ethernet Ring   100 Mbps Fast Ethernet Ring

1 Gbps Fast Ethernet Ring

NFS client B

NFS server B
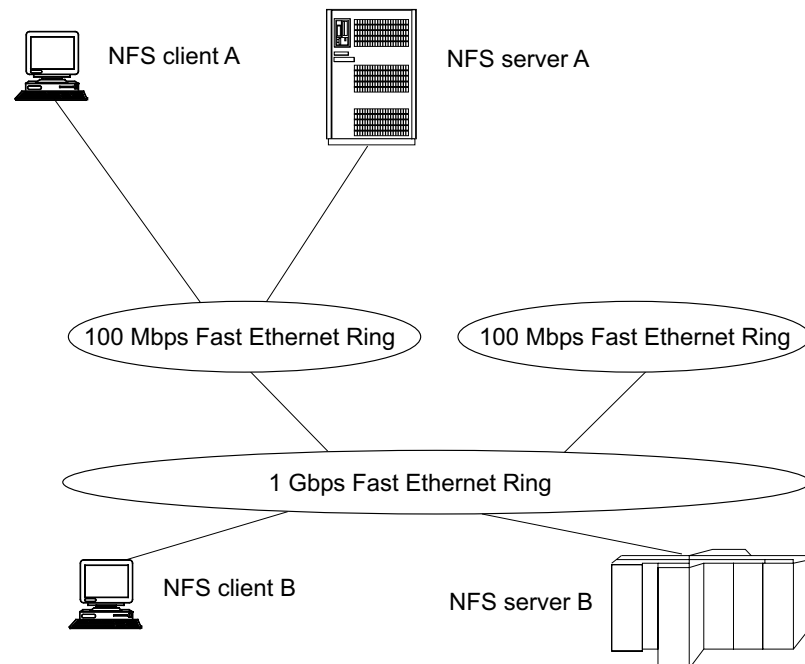
Figure 3. Sample network topology

There are other factors to consider in addition to the theoretical capacity of a network configuration. For instance, there are differences in the capability of network adaptors and network controllers. Overall network performance can also be influenced by the devices connecting network segments or subnets, particularly in terms of packet size limits and UDP checksum processing. While it may initially appear that there is no network constraint, there may be a device, adaptor, or network segment that is not performing at expected levels.

With this type of investigation, you may already have discovered that your primary constraint is a congested network. If so, you may elect to pursue such alternatives as:
- Increasing network bandwidth
- Changing network topology
- Scheduling applications during low system usage
- Modifying applications to reduce data transfer

ACTION 2: _Monitor network activity_ - It may be necessary to monitor network activity over a period of time to determine what is causing a performance problem. In addition to whatever monitoring your network administrator may be able to provide, the **netstat -s** command, available on many NFS client platforms, may provide some insight as well. This command reports such statistics as bad checksums, dropped fragments, non-forwardable packets, various timeouts, and socket buffer overflows. In some cases, it may be necessary to use a network analyzer to determine exactly what is being sent over a network segment. While it can be difficult to analyze congested networks, network analyzers occasionally capture evidence that may indicate a problem with the NFS client or server system.

NFS client system performance tuning

Table 3 contains NFS client system performance tuning symptom and action information.

Table 3. NFS client system performance tuning symptom and action information

Symptom	Action
Excessive NFS retransmissions	A high number of NFS retransmissions can be detected with the **nfsstat -c** command. See section "Impact of the NFS protocol on performance" on page 8 for more information. A general rule of thumb is that the number of retransmissions should not be more than 2% of the total calls. Use traceroute <hostname> command to trace the route that IP packets take to a network host. This helps isolate where the bottleneck may occur.

Symptom	Action

ACTION 1: *Increase network queue length* - If the **nfsstat -c** command shows evidence of bad calls in addition to timeouts, or the **nfsstat -s** indicates that packets or fragments are being dropped, increasing the network queue length may be advisable. Occasionally, a network analyzer will detect that the server has responded to an NFS request before the client retransmits the request. This indicates that the network queue on the NFS client system was full and the response was dropped.

ACTION 2: *Increase maximum transmission unit* - The **nfsstat -in** command will display the maximum transmission unit (MTU) for each network interface active on the NFS client system. The MTU can be reset with the **ifconfig** command. This command is available on most UNIX NFS clients. For the z/OS NFS client, MTU can be reset in the TCP/IP profile. Increasing the MTU can improve overall performance and alleviate excessive IP fragmentation. However, with heterogeneous networks the MTU should not be set larger than the smallest acceptable MTU over the entire network route. For instance, when the network consists of a mix of Token Rings and Ethernets, the MTU is typically set at 1492 that is less than the 1500 MTU for an Ethernet and a 2000 MTU for Token Ring. Changing the network topology may allow you to increase the NFS client system MTU.

ACTION 3: *Change socket buffer size* - Changing the buffer size is another way to tune the NFS client system. The AIX/6000 **no -a** command will display the current settings, in particular the *udp_sendspace* and *udp_recvspace*. The AIX/6000 **no -o udp_sendspace=32768** and **no -o udp_recvspace=32768** commands can be used to reset the *udp_sendspace* and *udp_recvspace*, respectively. For the z/OS NFS client, the *udpsendbfrsize* and *udprecvbfrsize* in the TCP/IP profile should be set respectively. The recommended value for Sun NFS V3 protocol is 65536. When processing with TCP protocol, it may be necessary to modify the *tcpsendbfrsize* and *tcprecvbfrsize*.

ACTION 4: *Increase number of Biods* - If data on an NFS server is typically accessed sequentially, or if there are many users on an NFS client system, it may be advantageous to increase the number of block I/O daemons (Biods). This increases the NFS client processor utilization and should be weighed against other processor requirements on a multiple user system. If access is typically at random offsets within a file or file sizes are very small, you may even consider decreasing the number of Biods. It should also be noted that some NFS client implementations honor Biods differently to provide read or write bias.

ACTION 5: *Increase timeout parameters* - If you have determined that NFS retransmissions are not caused by dropped NFS responses, you may consider increasing the timeout value. *This should be one of the last alternatives you select since this can have adverse effects.* If information is lost or dropped between the NFS client and the NFS server, increasing the timeout value can make performance worse. In this case the client would wait longer to decide to resend a request. In a heavily used NFS environment, increasing the timeout value also increases the likelihood that internal client or server buffers will become stagnant or stolen to service other network requests. You would probably only want to attempt this if you have strong evidence, probably from a network analyzer, that the NFS server is sending responses *after* the NFS client retransmits requests. The timeout values can be reset with the **mount** command.

NFS server system performance tuning

Regardless of whatever network and NFS client system tuning is done, some level of tuning or resource balancing should be addressed on the z/OS NFS server system. Typically, these systems have tuning and capacity planning procedures in place along with a high level of experience. Consequently, this section will be limited to a surface level discussion of the resource components to be considered when tuning the NFS server system.

Network controller constraints

Table 4 contains network controller constraints symptom and action information.

Table 4. Network controller constraints symptom and action information

Symptom	Action
Insufficient network capacity	**ACTION 1:** *Change controller configuration* - The performance of z/OS NFS is dependent on how well the MVS TCP/IP application and network controllers perform. This section will use the IBM 3172 Interconnect Controller, hereafter referred to as 3172, as an example of some items to consider for network controllers. The 3172 is available in a variety of hardware configurations. The 3172, also available with different memory and processor capability, supports Ethernet, Token Ring, and Gbps Fast Ethernet network adaptors as well as parallel and ESCON® channel attachment. In addition to variations in 3172 hardware configurations, there are alternatives for the software configuration as well. The 3172 can run the Interconnect Control Program (ICP) or an optional feature supporting TCP/IP Offload. Running the 3172 with TCP/IP Offload reduces the CPU time consumed by the TCP/IP address space. This reduction in CPU consumption is exchanged for a reduction in data transfer rates from the client system to the server system. The degree of impact is dependent on the amount of memory and processor capability of the 3172. Data transfer rates from the server system to the client system are generally not impacted. Tuning of the 3172 running TCP/IP Offload should be done in conjunction with tuning of the MVS TCP/IP application. **ACTION 2:** *Tune the network controller* - When the 3172 is running in ICP mode, there are two parameters to consider: the BLOCK DELAY TIME and the MAXIMUM RESPONSE LENGTH. The block delay time is the amount of delay allowed to block network frames for transmission. The recommended value for best performance is 10 milliseconds. The response length is the length of the longest frame to be sent to the MVS system without blocking. The recommended value for the maximum response length is 500 bytes. **ACTION 3:** *Upgrade the network adapter* - The S/390® Open System Adapter 2 (OSA2) is an integrated S/390 hardware feature plugged in place of a channel card. The OSA2 becomes an integral component of the I/O subsystem to enable convenient LAN attachment. The OSA2 is available in a variety of configuration modes. Using OSA2 may eliminate constraints with 3172 controller. Tune OSA2 on MVS TCP/IP if TCP/IP passthru mode is used.

MVS constraints

Table 5 on page 15 contains MVS constraints information. These actions should be weighed against any increase in MVS storage they may require.

Table 5. MVS constraints symptom and action information

Symptom	Action
High CPU utilization with low aggregate throughput	**ACTION 1:** *Increase MVS TCP/IP send and receive buffers* - There are other applications besides z/OS NFS which are dependent on the 3172 and the MVS TCP/IP application. Some applications have different tuning requirements. The default values for *udpsendbfrsize* (or *tcpsendbfrsize*) and *udp udprcvbrfsize* (or *tcprcvbrfsize*) is 16 KB. It may be useful to increase these parameters up to 64 KB for better throughput with large files. **ACTION 2:** *Increase MVS TCP/IP UDP queue* - Another area to consider is the MVS TCP/IP UDP queue. As discussed in section "NFS client system performance tuning" on page 12, NFS responses are dropped when network adaptor queues are full. A similar situation occurs when the UDP queue of MVS TCP/IP is full; in this case incoming NFS requests are dropped. The *noudpqueuelimit* keyword in the *assortedparms* section of the TCP/IP profile data set can be specified to enable the MVS TCP/IP server to accept incoming UDP datagrams. Without specifying this keyword, the default queue length of 30 may not be sufficient. **ACTION 3:** *Modify NFS and TCP/IP dispatching priorities* - It is recommended that the z/OS NFS procedure have a relative dispatching priority less than that of TCP/IP. With MVS/ESA Version 5 Release 1.0 or later this is especially important since the System Resource Manager (SRM) algorithms have changed. Prior to this release of MVS mean time to wait algorithms resulted in the z/OS NFS receiving a lower dispatching priority than TCP/IP due to the order in which the address spaces were started. With this newer release of MVS mean time to wait dispatching priorities are adjusted based on increased I/O activity. Since TCP/IP has higher network I/O than z/OS NFS, the TCP/IP dispatching priority is lowered. Assigning fixed dispatching priorities, with TCP/IP dispatched at a higher relative value than the z/OS NFS, can alleviate this situation. **ACTION 4:** *Run TCP/IP Offload on 3172* - Running TCP/IP Offload on the 3172 may reduce CPU time in the MVS TCP/IP address space. Smaller buffer sizes typically recommended for TCP/IP Offload may improve MVS storage usage as well. However, transfer rates will be reduced when the capacity of the 3172 has been reached. Transfers from the client system to the server system are particularly prone to reduced transfer rates. **ACTION 5:** *Select transport protocols* - z/OS NFS supports TCP and UDP as transport protocols for the server, for both NFS Version 2 and Version 3 protocols. UDP is primarily used for its efficiency on high bandwidth, low latency networks, such as LANS. TCP is used for its efficiency on low bandwidth, high latency networks, such as WANS.
Constrained throughput with low CPU utilization	**ACTION 1:** - See "Subtasking" on page 29.

Table 5. MVS constraints symptom and action information (continued)

Symptom	Action
High DASD utilization and NFS command response time	**ACTION 1:** *Evaluate placement of MVS catalog data sets* - The performance of z/OS NFS will be impacted when serving files located on heavily utilized storage devices or devices behind congested storage controllers. Before deciding whether or not to move data or upgrade storage systems, make sure system catalogs are not on heavily utilized devices. Many of the NFS commands that are executed for conventional MVS data sets involve catalog access. In fact, listing directories, one of the most commonly executed commands on NFS client systems, accounts for a significant portion of the NFS *get attribute* and *lookup* commands. Such commands not only cause the MVS catalog to be accessed but might also cause the entire file to be read depending on: • z/OS NFS processing options • Prior access via z/OS NFS • Whether or not files are DFSMS-managed **ACTION 2:** *DFSMS-managed data accessed from the network* - Files that are primarily accessed via z/OS NFS should be DFSMS-managed for improved performance. The z/OS NFS maintains file attributes for DFSMS-managed data sets when the z/OS NFS *nofastfilesize* processing option is specified. The *nofastfilesize* processing option provides exact file size determination rather than approximating file size as with the *fastfilesize* processing option. DFSMS management also provides improvements in terms of better storage utilization and specification of service levels. **ACTION 3:** *Evaluate placement of data accessed from the network* - While deciding whether or not to access DFSMS-managed files via the z/OS NFS, consider also the placement of such files. Files with critical performance requirements should be placed on the appropriate devices via the storage class parameters. If reallocation is necessary, you might also consider allocating sequential access method striped data sets for larger files or an HFS for smaller files. If you elect to maintain data set organization, you may choose to reblock the data set to a more optimal block size, such as half track blocking. You may even determine for some files that the best alternative is to store the files locally on the NFS client system.

Chapter 4. Evaluating z/OS NFS performance

This chapter assists both NFS client users and NFS server system administrators in evaluating the performance of z/OS NFS. Test methods are provided to assist in collecting performance data for evaluation. The appropriateness and usage of these test cases and methods depend on the following conditions:
- Workload requirements to be evaluated
- Level of expertise of the evaluator
- Monitoring tools available

Evaluating throughput

Throughput refers to the rate at which data is transferred between the NFS client and server systems. Establishing throughput baselines for the Client-Server environments of interest will assist in determining any benefits achieved from tuning. Baselines and measurement techniques are also useful in diagnosing performance problems.

As discussed in "How is performance characterized?" on page 5, a set of work activities or workloads should be identified for measurement. For throughput the type of workload selected should reflect the type of NFS read and write access expected by typical users or specific applications. Consider the following points:
- Average file size or range of file sizes
- Sequential or random access
- Data set types and organizations to be accessed
- Requirements for ASCII/EBCDIC translation

Single process throughput

The easiest place to begin evaluating NFS read and write throughput is on a single client and a single process basis. While measuring in a controlled and isolated laboratory situation is advantageous, it may not be practical. Measuring the actual environment of interest can provide more meaningful information as well as establishing a methodology for monitoring performance in the future. Under such uncontrollable situations, it is important to determine range and normal variation in measurements with particular attention to peak and non-peak periods of activity.

Once the NFS environment has been selected for measurement, some simple techniques can be used to initially evaluate NFS read and write throughput. One of the easiest methods is to use commands available on the NFS client system, such as **copy** or **cp**, to generate NFS reads and writes. Copy commands along with **date**, **time**, or **timex** commands can be used to determine the elapsed time for the copy. Throughput can be readily calculated from the elapsed time and the size of the file.

Some additional considerations when measuring NFS throughput are the effects of graphical user interfaces (GUI), any overhead associated with opening files, and I/O to and from local physical storage. Measuring with and without a GUI will show the effect on NFS throughput caused by the GUI. Using a network analyzer can assist with isolating the time spent opening files from the time spent executing NFS reads or writes. Techniques such as copying to */dev/null* can eliminate local physical I/O for NFS reads. For more complex requirements such as reading and writing at random offsets, you may use an existing application to provide such access or write a program to execute NFS procedure calls at random offsets within a file. Program

generated NFS procedure calls can also be useful when available local physical storage is insufficient for the file sizes to be measured.

Multiple process throughput

For NFS client systems with multiple users or applications you may want to determine aggregate throughput on a multiple process basis. If throughput on a single process basis has achieved the network capacity, then multiple process throughput will be limited by the same network capacity as well as any NFS client system limitations. A simple method for evaluating multiple process throughput is to simply execute multiple single process measurements simultaneously.

Multiple client throughput

Evaluating multiple client throughput requires the ability to propagate the measurement workload over more than one NFS client system, preferably the same type of system. Such measurements are more complex in that they require controlled execution and collection of results from multiple systems. Depending on your network configuration, remotely executing your single process workload on multiple systems is a relatively simple method to use to initially evaluate multiple client aggregate throughput.

Evaluating NFS command response time

A user is probably more interested in the response time for typical user commands, such as listing a directory, making or removing a directory, and creating or removing a file. However, when such commands are executed for NFS mounted file systems, the NFS command response times become a factor in user command response time. The transparent nature of the NFS protocol allows a user to define a set of commands as a measurement workload with a minimum amount of effort.

For those desiring more detail on NFS command response time, there are public domain programs as well as industry standard benchmarks. Both of these options require more experience on the part of the user and may have limitations on their usage. Such programs are typically written for the UNIX environment and may not support all NFS client and server systems.

Evaluating CPU utilization

Evaluating CPU utilization on an MVS system can be quite complex, particularly for a production system. The activity on such systems is already generally monitored and z/OS NFS is simply one more application to be monitored. High level monitoring can be accomplished from the Display Active (DA) display of the System Display and Search Facility (SDSF). This display provides an overview of the total activity on the system. Of particular importance are the CPU usage and relative dispatching priorities of TCP/IP and z/OS NFS. Monitoring tools such as the Resource Measurement Facility (RMF™) Monitor I and Monitor II can provide more detail on the performance characteristics of the MVS system.

Collecting server usage data

You can use the SMF records that z/OS NFS produces to keep track of how files are accessed and how long each NFS user session lasts. The following SMF records can be produced:

Record type 42 subtype 7	This record, written when a file times out, provides file usage statistics.

Record type 42 subtype 8	This record, written when a client user logs out of the z/OS NFS, provides user session statistics.

Chapter 5. Tuning the z/OS NFS server

Previous chapters discussed performance tuning from a total system perspective. This chapter discusses performance tuning at the z/OS NFS server level. In particular, this chapter focuses on the z/OS NFS data set creation, processing, and site attributes.

Data set creation attributes

When creating new MVS data sets, the allocation parameters are the most readily accessible means for the end user to influence the performance of z/OS NFS server. They are also not well known outside of the MVS user community. This is primarily due to the fact that MVS has a record oriented data structure and most NFS client systems have a byte oriented data structure. This section looks at how a variety of allocation parameters affect performance.

Block size and record length

Block size (BLKSIZE) specifies the maximum length, in bytes, of a physical block of storage in MVS. If BLKSIZE(0) is specified, the system will determine the optimal block size based on the maximum record length (LRECL) and the physical characteristics of the disk, or approximately half of a physical track. Half track blocking is optimal in that it reduces the amount of wasted storage on the disk.

The BLKSIZE is also important to the performance of the z/OS NFS, which performs physical I/O on a block basis. While half track blocking is generally optimal, for z/OS NFS server you should be aware of the relationship between the block size, the file record length, the network packet size and the NFS client **mount** write and read buffer sizes, *wsize* and *rsize*.

A network packet size will be negotiated between the NFS client and the NFS server when the NFS **mount** is processed. In general, the NFS server will determine the packet size for write operations, and the NFS client will determine the packet size for read operations. For z/OS NFS, this packet size is 8192 bytes, when processing with Sun NFS V2 protocol. For several NFS client implementations the default packet size is 8192. However, there are some implementations with a 4096 byte maximum packet size. The maximum packet size is 65536 bytes when processing with Sun NFS V3 protocol.

The packet size can also be automatically reset to a smaller value when a constraint is detected. Such a reduction in packet size causes an increase in the number of NFS requests for the server to process the same amount of data. This increases the load on the server and may or may not have been the initial constraint causing the packet size reduction. This can have an adverse effect on z/OS NFS server performance, particularly when processing write requests.

Regardless of the packet size, the z/OS NFS server must reassemble the packets, possibly arriving out of sequence, into records. z/OS NFS server buffers (see "Ordering out-of-sequence data" on page 27) and orders these packets until a block of data can be written to disk. Write requests, especially those for large files, are processed more efficiently when the likelihood of packets spanning records and blocks is reduced.

When z/OS NFS server is processing read requests, the physical I/O is again on a block basis. Consequently, read throughput is improved by increasing the block size up to half track blocking.

If files are allocated with a default block size (BLKSIZE=0) on RAMAC® 3 or Enterprise Storage Server (ESS™) 2105 DASD with 3390-3 format, the actual block sizes will probably vary between 24 and 27 KB, depending on the specified record length and record format. Table 6 shows the actual allocation block sizes for physical sequential data sets allocated on a RAMAC 3 or ESS 2105 DASD with a variety of record lengths and with both fixed and variable length records.

Table 6. Default block sizes for RAMAC 3 or ESS 2105 DASD

Record Format	Record Length (bytes)	Block Size (bytes)
FB	80	27,920
FB	1024	27,648
FB	4096	24,576
FB	8192	24,576
VB	84	27,998
VB	1028	27,648
VB	4100	27,998
VB	8196	27,998

Record format

Record format (RECFM) specifies the format and characteristics of the records in the data set. This section will be limited to a discussion of the performance considerations associated with fixed and variable length records.

There seems to be a natural affinity toward allocating MVS data sets with variable length records from the perspective of a byte oriented NFS client operating system. Such allocations enhance the feeling of transparent access and eliminate requirements to determine a fixed record length. However, the placement of incoming packets into records of varying lengths is more complex than placing packets into records with fixed lengths. Consequently, write performance for z/OS NFS is generally more efficient when MVS data sets are allocated with fixed length records.

Data set organization and data set type

Data set organization (DSORG) specifies the organization of an MVS data set. The z/OS NFS data set organization attribute can be physical sequential (ps), direct access (da), or VSAM. z/OS NFS server supports three types of VSAM files: key-sequenced (KSDS), entry-sequenced (ESDS), and relative record (RRDS). However, keyed access and relative-number access are not supported by the NFS protocol.

The DSORG attribute is ignored for directory-oriented NFS client commands. The data set type (DSNTYPE) specifies whether a PDSE or PDS is to be created when the z/OS NFS receives a **mkdir** command from the NFS client. A PDSE is created when library is specified, and a PDS is created when PDS is specified. You cannot create another PDS (or PDSE) within a PDS (or PDSE).

Another MVS data set type is extended (EXT) which identifies an extended format data set. With z/OS NFS these DFSMS-managed data sets are allocated based on

specification of a data class (DATACLAS). An appropriate data class must have been defined on the MVS system for an extended format data set to be allocated. One of the values specified when creating an MVS data class is the volume count which determines the number of stripes allocated for an extended format data set. If automatic class selection (ACS) routines have been written to allocate extended format data sets based on such criteria as naming conventions, it would not be necessary to specify a DATACLAS to allocate an extended format data set via the z/OS NFS server.

The hierarchical file system (HFS) is another data set type. The HFS must be DFSMS-managed and be mounted within the z/OS UNIX subsystem. While an HFS cannot be allocated using z/OS NFS, an HFS can be accessed through the z/OS NFS server.

Data set organizations and data set types are generally selected based on user and application requirements. However, performance should be considered as one of these requirements. An important factor to consider when evaluating performance is the size of the files to be accessed. For smaller file sizes, we can differentiate between the overhead costs associated with creating or opening a file for output. As the file size increases, this impact becomes less of a factor in the transfer rates.

Processing attributes

Processing attributes are used to control how files are accessed from the NFS client system. Default values can be specified when the server is started. These defaulted values can be overridden at the NFS **mount** level or at the individual command level. The processing attributes, like the data set creation attributes, can affect the performance of the Network File System (NFS). This section will be limited to those processing attributes which have the most influence on performance.

Character translation

The processing attributes that specify whether or not data conversion occurs between ASCII and EBCDIC formats may have the most influence on the performance of z/OS NFS. The binary attribute, which specifies no data conversion, is the most efficient manner in which to access data via z/OS NFS. However, data conversion may be a requirement, particularly when data is shared with MVS users or applications. Under such circumstances the text processing attribute would be specified instead of the binary processing attribute. The actual impact associated with data conversion will depend on the data to be converted.

File size determination

The attributes discussed previously have primarily affected data transfer rates. Whether or not to determine the exact file size in bytes has more of an effect on user command response time. This is because the file size in bytes is one of the file attributes obtained by an NFS Get Attribute procedure. The NFS Get Attribute procedure is executed for every user command accessing an NFS mounted file system.

z/OS NFS provides processing attributes to specify estimated file size determination, *fastfilesize*, or exact file size determination, *nofastfilesize*. The accuracy of the file size with *fastfilesize* processing is best for binary processing of data sets with fixed length records. For other situations, or when exact file size is a requirement, the *nofastfilesize* processing attribute should be specified.

When the *nofastfilesize* processing attribute is specified, there are performance factors to consider. The most important factor is that file size in bytes is maintained by z/OS NFS for DFSMS-managed data sets with many data set organizations.

Figure 4 compares the differences in response time between the z/OS NFS attribute support for DFSMS-managed data sets and the case when attributes are not cached for non-managed data sets. In this example, all data sets were members of a PDSE data set allocated with a variable block record format. The ls -l command was executed with the z/OS NFS *nofastfilesize* and binary processing attributes.

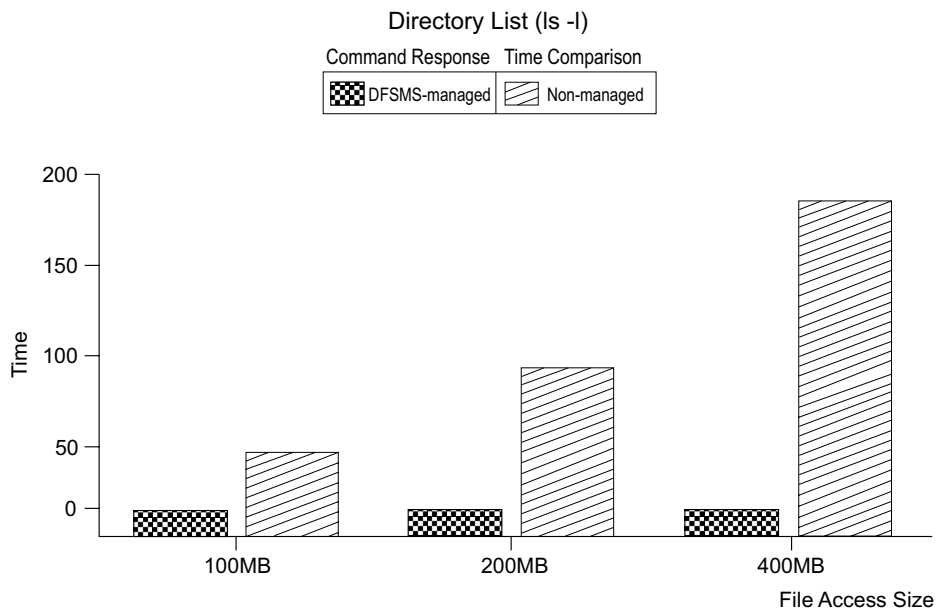Directory List (ls -l)

Command Response Time Comparison

Figure 4. Directory list comparison between DFSMS-managed and non-managed

Data sets processed with the *nofastfilesize* attribute that are not DFSMS-managed will need to be read entirely to determine the exact file size in bytes when initially accessed via z/OS NFS. z/OS NFS will cache this information until the data set is modified (see section "Buffer usage and caching" on page 26) or the attribute time value has expired. The larger the file is, the greater the impact to command response time is. Figure 4 shows how the response time for a directory list command can be impacted when the file has to be read to determine the exact file size in bytes.

Maintaining the file size attribute by z/OS NFS also affects read throughput when accessing larger DFSMS-managed data sets with the *nofastfilesize* processing attribute. In the *nofastfilesize* case, reading entire file to determine byte file size is not necessary. The impact on the end-to-end response time for the read is lessened by reducing the response time for the initial access to obtain file attributes.

Data set timeout specification

There are three timeout processing attributes that specify when z/OS NFS will release a data set following certain NFS operations. The *readtimeout* and *writetimeout* processing attributes specify how long z/OS NFS will wait after respective read and write operations before releasing a data set. The *attrtimeout* specifies how long a data set will remain allocated after an NFS Get Attribute or Lookup operation. When accessing data sets that are not DFSMS-managed with

the *nofastfilesize* processing attribute, it may be advantageous to increase the *attrtimeout* value so that z/OS NFS caches data set attributes for a longer period of time. However, all three processing timeout attributes must be within the range of the *maxtimeout* and *mintimeout* site attributes.

Accessing migrated files

While migrating less frequently accessed data sets to tape improves MVS storage utilization, retrieving migrated data sets can impact the performance of z/OS NFS. z/OS NFS provides three processing attributes to specify how migrated data sets are to be handled by the server: *retrieve(wait)*, *retrieve(nowait)*, and *noretrieve*.

The *retrieve(wait)* attribute instructs the server to wait for a migrated data set to be recalled to a direct access storage device. The NFS client user or application process will not receive a response from z/OS NFS until the data set has been recalled. If the migrated file is relatively large, or migrated to tape, this can cause the NFS client to retransmit the request. The NFS client will retransmit the request a fixed number of times as specified by the **retrans** parameter of the **mount** command. If the recall of a migrated data set takes longer than the product of the timeo value on the **mount** command and one plus the **retrans** value for the **mount**, **timeo * (retrans +1)**, the initial command will most likely need to be executed again. Using the default values for the AIX/6000 **mount** command of seven-tenths of a second for *timeo* and three attempts for **retrans**, a recall would need to take less than 3 seconds. An alternative approach is to use the *retrieve(nowait)* processing attribute.

The *retrieve(nowait)* attribute instructs the server to recall a migrated data set and immediately return a ″device not available″ message to the client without waiting for the recall to complete. With this option, users attempting to access the file can continue to use their session for other activity rather than waiting an indeterminate time for the recall of a file. They can attempt to access the file again after allowing some period of time for the recall to have completed.

If it is critical to the user that a file be recalled before further work can continue, the *retrieve(wait)* processing attribute can be specified more selectively as part of the NFS client user command syntax along with the file name. While doing this provides the user more control over command execution, the user command may still timeout and have to be executed again.

Specifying the **retrieve(wait)** processing attribute as the system default may also impact the availability of z/OS NFS subtasks to process such requests. The second parameter of the **nfstasks** site attribute determines the number of z/OS NFS subtasks available to process such recall operations. See "Subtasking" on page 29 for more information about the **nfstasks** site attribute.

Another method of avoiding the unnecessary recall of migrated data sets is to specify the *noretrieve* processing attribute. With the *noretrieve* attribute, the server does not recall a migrated data set and a ″device not available″ message is returned to the client. This processing attribute is particularly useful when listing a directory with unknown contents.

Asynchronous HFS processing

z/OS NFS, with Sun NFS V2, protocol provides two processing attributes that only apply to HFS processing. The *sync* attribute specifies that HFS write requests should be committed to nonvolatile media (for instance, DASD) when received by the server. Some performance improvement can be obtained with the alternative

async processing attribute. This type of processing causes HFS write requests to be cached at a level within the z/OS UNIX architecture prior to the physical I/O. For more information about HFS tuning, see "Ordering out-of-sequence data" on page 27 and "Subtasking" on page 29.

The *async* and *sync* attributes specified in the control file will be ignored when z/OS NFS server is communicating with Sun NFS V3 protocol. The processing attribute is determined in the client application implementation. Asynchronous write is also recommended for faster throughput.

Site attributes

The site attributes control z/OS NFS resources. Tuning of these parameters should be done with caution. They are only specified at the start of the z/OS NFS server and cannot be modified by client users. This section is limited to those attributes which more directly affect performance.

Buffer usage and caching

The *bufhigh* site attribute specifies the maximum size in bytes of allocated buffers before any buffer reclamation is initiated (see Figure 5). When the *bufhigh* limit has been reached, a percentage of the buffers will be reclaimed, and the amount of reclamation is determined by the *percentsteal* attribute. z/OS NFS uses this buffer area to cache file information, thereby satisfying requests more efficiently.
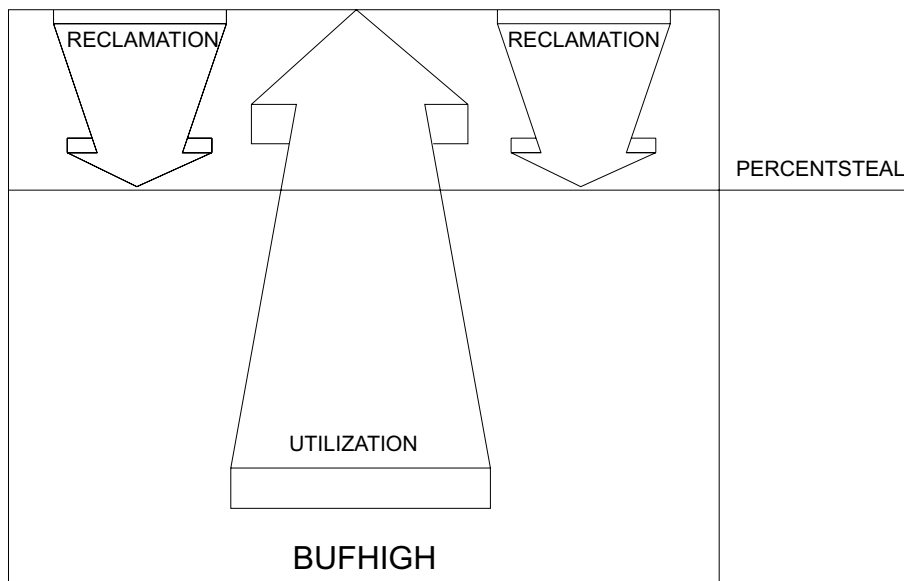

Figure 5. Bufhigh utilization with percentsteal

There are two additional site attributes, *readaheadmax* and *maxrdforszleft*, that also affect *maxrdforszleft* usage. The *readaheadmax* attribute specifies the number of bytes to be read to fill the internal buffers so that additional read requests for that file may be satisfied directly from cache. The *maxrdforszleft* site attribute defines the number of physical block buffers to cache after determining a file's size (see section "File size determination" on page 23). This information is cached to satisfy any subsequent read requests for the same file.

Keep in mind also that some information will be cached on a file handle basis, in other words, for every file accessed within the timeout periods. Tuning of the *bufhigh*, *readaheadmax*, and *percentsteal* values should be determined based on the following conditions:
- Number of files to be accessed
- Available region
- Amount of data to cache per file

For example, suppose that, on average, 1,000 physical sequential (DSORG=PS) files allocated on 3390 Model 3 DASD are accessed within the same timeout period via the z/OS NFS server. Further, assume that we would like two blocks of a file cached internally to satisfy read requests. If we assume an average block size of 25 KB (see "Client-server relationship and terminology" on page 1 for more detail), we would need at least 50 MB of storage for internal buffers. With a 20 percent value for *percentsteal*, a 64 MB value for *bufhigh* would probably be more reasonable. For this example, a reasonable value for the *readaheadmax* attribute is the file block size, or 25 KB. The *readaheadmax* value should not be less than twice the maximum record length of files accessed via the z/OS NFS server.

Ordering out-of-sequence data

While the previous section primarily addressed satisfying read requests from cache, this section discusses the caching used by z/OS NFS to satisfy write requests. z/OS NFS provides two site attributes for this purpose: *cachewindow* and *logicalcache*. As with all site attributes, *cachewindow* and *logicalcache* can only be modified at server startup.

The *cachewindow* attribute specifies the window size, in terms of number of packets, used in logical I/O to buffer client block writes received out of sequence. Figure 6 on page 28 shows how out of order packets are buffered in a cache window until a block of data can be physically written to storage. Since out of sequence packets are an inevitability, a certain amount of buffering is necessary.

In a congested network environment, the nature of the underlying protocols should be taken into consideration when tuning the *cachewindow* attribute. Under such circumstances, NFS, UDP, and IP requests are more likely to timeout. When this situation is detected within the underlying protocols, the packet size can be reduced automatically. The packet size is generally halved until the server and client systems are balanced. The minimum packet size that can be negotiated is 512 bytes. Accessing the z/OS NFS server in such an environment can result in the default 8 KB read and write buffers being reduced to 512 bytes, or 16 packets instead of one packet. *Under these circumstances the recommended value for cachewindow is 16 times the number of Biods.*

Another factor to consider in tuning the *cachewindow* attribute relates to the relationship between packet size, data set record length, and block size. If the packet size is typically greater than or equal to the data set block size, there is less of a need to buffer the packets. On the other hand, if the packet size is smaller than the block size and/or record length, a larger *cachewindow* value will be required to buffer the packets until a block I/O is executed.
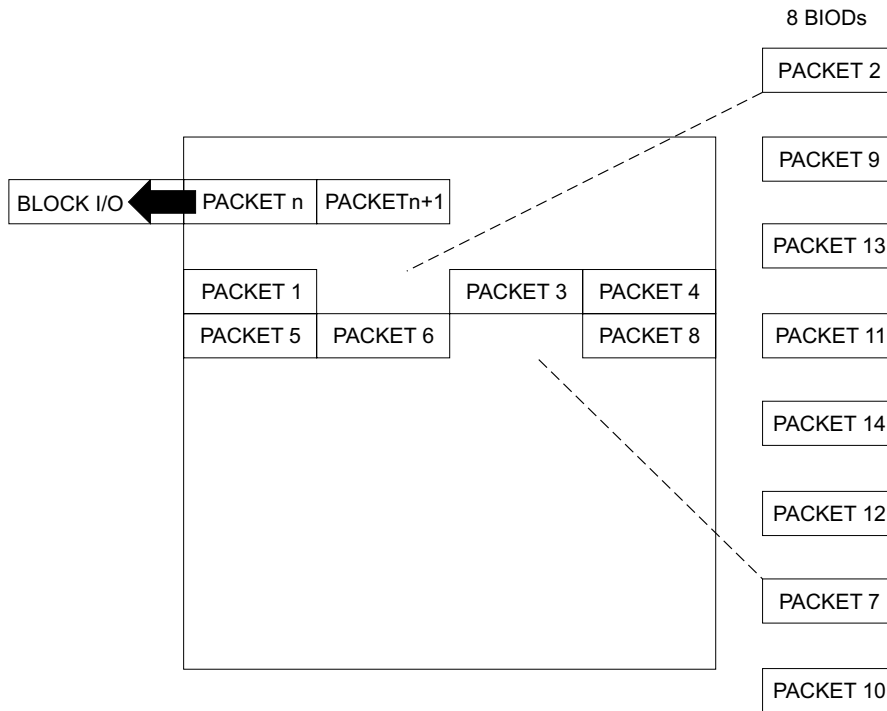
8 BIODs

PACKET 2

PACKET 9

PACKET 13

| BLOCK I/O | PACKET n | PACKETn+1 |

PACKET 11

| PACKET 1 | | PACKET 3 | PACKET 4 |
| PACKET 5 | PACKET 6 | | PACKET 8 |

PACKET 14

PACKET 12

PACKET 7

PACKET 10

Figure 6. Relationship between cachewindow and Biods

The *logicalcache* site attribute specifies the size (in bytes) of allocated buffers for all the cache windows combined (see Figure 7). The number of cache windows in use will depend on the number of file handles accessed within a given timeout period.

LOGICALCACHE

CACHEWINDOW			CACHEWINDOW		
PACKET	PACKET	PACKET	PACKET	PACKET	PACKET
PACKET		PACKET	PACKET	PACKET	
	PACKET	PACKET	PACKET		PACKET
PACKET				PACKET	
CACHEWINDOW			CACHEWINDOW		
PACKET	PACKET	PACKET	PACKET	PACKET	PACKET
PACKET	PACKET		PACKET		PACKET
	PACKET	PACKET	PACKET	PACKET	PACKET
	PACKET		PACKET		

Figure 7. Logicalcache utilization for cachewindows

Tuning of the *logicalcache* and *cachewindow* values should be determined based on the following conditions:
• Number of files accessed for write
• Available region

- Typical packet size
- Data set block size
- Number of client Biods

The *cachewindow* and *logicalcache* attributes do not apply to HFS processing.

For example, suppose you have AIX/6000 NFS client workstations on a lightly loaded network, with the default value of six Biods, writing to physical sequential MVS data sets allocated with 8 KB fixed length records. Further, assume that these files were allocated on 3390 Model 3 DASD with a 24 KB block size and that the network packet size is 8 KB. Then, a value of 12, twice the number of NFS client Biods, for *cachewindow* means that an out of sequence packet could be internally buffered within a 96 KB or four block range. If a packet is received outside this range, z/OS NFS must read data from storage in order to write the packet at the correct offset into the data set. z/OS NFS detects and ignores duplicate requests.

A value of 12 for *cachewindow* and a network packet size of 8 KB in our example results in a 96 KB buffer requirement per client actively writing to MVS data sets via z/OS NFS. A 5 MB value for *logicalcache* is sufficient to satisfy sustained write requests from 50 NFS client workstations at the same time.

Storage considerations

To avoid storage shortages, consider the following conditions:

- If a region size is specified, the `bufhigh + logicalcache` value should not be more than 90 percent of the region size. (The more **bufhigh**; the better performance.) With **bufhigh(1M)**, **logicalcache(1M)**, and the existing NFS core size (currently the core size is about 20M), the minimum region size is 25M.
- If a region size is equal to 0, **bufhigh + logicalcache** value is limited by the supported virtual memory size on the installation. Region size can be set up as large as the size of the local page data sets.

The System Programmer should also consider if the other jobs on the installation use virtual memory.

Subtasking

The **nfstasks (n,m,o)** site attribute specifies the number of server processes to initiate on startup. If **nfstasks(n,m)** is specified, then the valid value range for both n and m is 1 to 24. The sum of n and m must be less than or equal to 25. If **nfstasks(n,m,o)** is specified, then the following is true–the valid value range for 'n' is from 1 to 24. The valid value range for 'm' is from 1 to 100. The valid value range for 'o' is from 1 to 24. The sum of 'n' plus 'o' must be less than or equal to 25.

The absolute and relative value of **n** and **m** should be tuned for the expected system usage. If conventional MVS data sets will be accessed, primarily, then **n** should be relatively high. If z/OS UNIX files will be accessed, primarily, then **m** should be relatively high. The absolute value of these will influence the amount of system resources consumed (higher values will make more system resources available to process NFS request.

The parameter **n** is the number of processes started to handle asynchronous operations (conventional MVS data set access) and short duration synchronous operations (SAF calls). The parameter **m** is the number of processes which handle HFS requests. Increase this value if your server supports lots of active hierarchical file system (HFS) clients. The parameter **o** is the number of processes which handle longer duration synchronous operations (concurrent NFS server recall

requests.) Increase this value if your server supports lots of active recall operations. When ***nfstasks(n,m)*** is specified, m represents both the processes for HFS operations as well as the long duration synchronous operations. It is better to use the nfstasks(n,m,o) format to have better granularity of control as well as have the capability to specify a higher number of HFS processes to handle more HFS operations.

Chapter 6. Tuning the z/OS NFS client

This chapter discusses performance tuning at the z/OS NFS client level. Performance of the z/OS NFS client is affected by the following installation parameters that can be tuned (see Table 7). The default value of each parameter is specified in parentheses.

Table 7. Client installation parameters for tuning

AttrCaching (Y)
Biod (6)
Bufhigh (32)
DataCaching (Y)
DelayWrite (16)
DynamicSizeAdj (Y)
ReadAhead (8)

Most of the installation parameters that are specified in member `BPXPRMxx` in `SYS1.PARMLIB` can be overridden over a mount point by using the **mount** command. The exception to this is ***Biod*** and ***Bufhigh***, which require restarting z/OS UNIX.

Table 8 shows mount parameters that affect performance and that can be tuned. The default values are specified in parentheses.

Table 8. Mount parameters for performance and tuning

AttrCaching (Y)
DataCaching (Y)
DelayWrite (16)
ReadAhead (8)
rsize (8192 for V2) (No default for V3 - system determined)
wsize (8192 for V2) (No default for V3 - system determined)
vers (3)

Caching

Caching of file attributes (***attrcaching=y***) and data (***datacaching=y***) are recommended for performance. By caching the file data, all subsequent references to the cached data is done locally, avoiding the network overhead. Further authorization checking for subsequent access to the cached data or for other client users is done on the z/OS NFS client system. With attribute caching, z/OS NFS client will perform consistency checks to maintain valid cached data.

DynamicSizeAdj

The **DynamicSizeAdj** parameter specifies the capability to have the packet size adjusted automatically when the remote procedure call (RPC) timeout has occurred.

For example, the **mount** command specifies, `rsize=32K`. If the RPC `READ` with a 32 K bytes request times out:
- The RPC request times out.
- The client cuts the size in half and sends the RPC `READ` with a 16 K request.
- The RPC request times out again.
- The client cuts the size in half and sends the RPC `READ` with a 8 K request.

Note: This process continues until the minimum size is reached.

Bufhigh

The **Bufhigh** parameter specifies the maximum size in megabytes of allocated buffers for caching. z/OS NFS client uses this buffer area to cache file data, thereby satisfying requests efficiently.

When a client system is suffering high CPU utilization and low throughput, increase the Bufhigh storage size. Increasing the Bufhigh storage size by twice the default value (within available region) may improve system performance. Also, if the client system is processing mostly large files, it is good to increase the Bufhigh storage size. Any changes to the Bufhigh parameter require restarting z/OS UNIX.

Biod

The z/OS NFS client uses block I/O daemons (Biod) to perform asynchronous read or write when **DataCaching** is enabled. The **Biod** parameter specifies the number of asynchronous block I/O daemons to use. If there are a lot of users on the NFS client system, it may be advantageous to increase the number of Biods. The maximum number of Biods should never exceed two times the number of client system processors. This will avoid excessive TCB switching. Tuning of this parameter should be done with caution. Any change to the Biod parameter requires restarting z/OS UNIX.

ReadAhead

With caching enabled, read requests may be satisfied from cache. The **readahead** parameter specifies the maximum number of disk blocks to read ahead. Thus, additional read requests for the file may be satisfied directly from cache. When application is anticipating mostly sequential reads, increase the readahead to the maximum. If application is mostly processing random reads, use the default or a minimal readahead value.

DelayWrite

With caching enabled, write requests may be cached. The **DelayWrite** parameter specifies the maximum number of disk blocks to cache. The z/OS NFS client will issue the `WRITE RPC` whenever the system load is low or if the cache data are in sequence. DelayWrite increases `WRITE` performance by overlapping the disk I/O. This performance increase is most apparent with many small writes or with many random writes. When there is a high volume of write requests, set the DelayWrite parameter to the maximum value(32). If there is a low volume of write requests, set the DelayWrite to the default (16).

Vers

Sun NFS version 3 protocol support is available on OS/390 V2R6 NFS. It has performance enhancements over Sun NFS version 2 protocol. The parameter does not have to be specified unless you would like to override the default. The z/OS NFS client will communicate at the highest protocol level supported by the server, by default. Verify the protocol level for tuning of read or write buffer size (**wsize**, **rsize**). Use `nfsstat -m /mountpoint` command and look for `vers(n)` in the report or use `rpcinfo -p servername` command and look for `100003 3 udp 2049 nfs` in the report.

Wsize and rsize

wsize and *rsize* specify the buffer size to use for read and write request. For NFS Version 2 protocol, rsize and wsize are a multiple of 512 bytes, up to a maximum of 8192 bytes.

For the NFS server with Version 3 protocol, *rsize* and *wsize* are negotiated between the z/OS NFS client and the NFS server. The maximum buffer size that is supported in z/OS NFS client system is 32 KB. The negotiated read and write buffer size will be the smaller of the buffer size supported by the server and 32 KB (min(server_info, 32KB)).

The z/OS NFS client system will reduce the read or write buffer size when there is constraint on the storage resource.

When using NFS Version 3 protocol the system normally determines the optimal buffer values. However, if the number of retransmissions or the number of time-out from *nfsstat* report is high, reduce the write buffer size (*wsize*) to 8 KB to avoid retransmission of large 32 KB data.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Information Enabling Requests
Dept. DZWA

5600 Cottle Road
San Jose, CA 95193 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Programming interface information

This book documents intended Programming Interfaces that allow the customer write programs to obtain the services of z/OS DFSMS.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX	OpenEdition
ESCON	OS/390
ESS	RAMAC
IBM	Resource Link
IBMLink	RMF
MVS/ESA	S/390
	z/OS

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Glossary

This glossary defines technical terms and abbreviations used in DFSMS documentation. If you do not find the term you are looking for, refer to the index of the appropriate DFSMS manual or view the *IBM Dictionary of Computing Terms* located at:

http://www.ibm.com/networking/nsg/nsgmain.htm

This glossary includes terms and definitions from:

- The *American National Standard Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 11 West 42nd Street, New York, New York 10036. Definitions are identified by the symbol (A) after the definition.

- The *Information Technology Vocabulary* developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions of published part of this vocabulary are identified by the symbol (I) after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.

- The *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

The following cross-reference is used in this glossary:

See: This refers the reader to (a) a related term, (b) a term that is the expanded form of an abbreviation or acronym, or (c) a synonym or more preferred term.

A

access. To obtain computing services.

access method. (1) A mainframe data management routine that moves data between storage and an I/O device in response to requests made by a program. (2) The part of the distributed data management architecture which accepts commands to access and process the records of a file.

access permission. A group of designations that determine who can access a particular AIX or UNIX file and how the user can access the file.

ACS. See *automatic class selection*.

ACS routine. A procedural set of ACS language statements. Based on a set of input variables, the ACS language statements generate the name of a predefined SMS class, or a list of names of predefined storage groups, for an MVS file.

address. The unique identifier assigned to each device or workstation connected to a network.

address space. The complete range of addresses in memory available to a computer program.

Advanced Interactive Executive (AIX). IBM's licensed version of the UNIX operating system.

AIX. See *Advanced Interactive Executive*.

alias. An alternative name for an ICF user catalog, a non-VSAM file, or a member of a partitioned data set (PDS) or PDSE.

alias entry. An entry that relates an alias to the real entry name of a user catalog or nonVSAM data set.

allocation. (1) Generically, the entire process of obtaining a volume and unit of external storage, and setting aside space on that storage for a data set. (2) The process of connecting a program to a data set or devices.

American Standard Code for Information Interchange (ASCII). The standard code used for information exchange among data processing systems, data communication systems, and associated equipment. It uses a coded character set consisting of 7-bit coded characters.

APAR. See *authorized program analysis report*.

APF. Authorized program facility.

API. See *application programming interface*.

application programming interface (API). A formally defined programming language interface between an IBM system control program or a licensed program and the user of a program.

ASCII. See *American National Standard Code for Information Interchange*.

atime. The time when the file was last accessed.

automatic class selection (ACS). A mechanism for assigning Storage Management Subsystem classes and storage groups to data sets.

automatic class selection (ACS) routine. A procedural set of ACS language statements. Based on a set of input variables, the ACS language statements generate the name of a predefined SMS class, or a list of names of predefined storage groups, for a data set.

authorized program analysis report (APAR). A request for correction of a problem caused by a suspected defect in a current unaltered release of a program.

B

basic sequential access method (BSAM). An access method for storing or retrieving data blocks in a continuous sequence, using either a sequential access or a direct access device.

BIOD. The caching daemon that caches directory lookups and file data when remote files are accessed from the host.

block. A string of data elements recorded, processed, or transmitted as a unit. The elements can be characters, words, or physical records.

BSAM. See *Basic sequential access method.*

C

CCSID. See *Coded Character Set Identifier.*

CDRA. Character data representation architecture.

Character Data Representation Architecture (CDRA) API. A set of identifiers, services, supporting resources, and conventions for consistent representation, processing, and interchange of character data.

client. (1) A user. (2) A consumer of resources or services. (3) A functional unit that receives shared services from a server. (4) A system that is dependent on a server to provide it with programs or access to programs. (5) On a network, the computer requesting services or data from another computer.

client-server relationship. Any process that provides resources to other processes on a network is a *server*. Any process that employs these resources is a *client*. A machine can run client and server processes at the same time.

Coded Character Set Identifier (CCSID). A 16-bit number that identifies a specific encoding scheme identifier, character set identifiers, code page identifiers,

and additional coding required information. The CCSID uniquely identifies the coded graphic character representation used.

connection. An association established between functional units for conveying information.

current directory. The currently active directory; the directory that is searched when you enter a file name without indicating the directory that contains the file name. When you specify a file name without specifying a directory, the system assumes that the file is in the current directory.

D

daemon. A background process that is initiated at system start that continuously performs a function required by another process.

DA. Direct access.

DAE. Dump Analysis and Elimination.

DAM. Direct access method.

DASD. Direct access storage device.

DASD volume. A Direct Access Storage Device space identified by a common label and accessed by a set of related addresses. See also *volume*, *primary storage*, *migration level 1*, *migration level 2*.

data control block (DCB). A control block used by access method routines in storing retrieving data.

data set. In DFSMS, the major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access. In z/OS non-UNIX environments, the terms *data set* and *file* are generally equivalent and sometimes are used interchangeably. See also *file*. In z/OS UNIX environments, the terms *data set* and *file* have quite distinct meanings.

data set organization. The type of arrangement of data in a data set. Examples are sequential organization or partitioned organization.

DBCS. Double byte character set.

DCB. See *data control block.*

DES authentication. Requires a client to send credentials (its name, conversation key, window key, and a time stamp) to the server. The server then returns a verifier to the client. Data Encryption Standard (DES) credentials are sometimes called "secure" credentials because they are based on a sender's ability to encrypt data using a common time reference. DES-style credentials require a randomly generated key used to encrypt a common reference time which is then used in

turn to create a *conversation key*. DES-style credentials are less vulnerable to penetration than UNIX-style credentials because of the generated conversation key and the data that is encrypted with the conversation key. Other forms of attack (for example, replaying message traffic) are resisted because of the timestamp in the transmission. The Network File System does not support DES-style credentials.

DFSMSdfp. A DFSMS functional component or base element of z/OS, that provides functions for storage management, data management, program management, device management, and distributed data access.

direct access file. A type of MVS file for storing data on a random access device that is accessed using a record address.

directory. A file that maps the names of other directories and files to their locations.

DOS. Disk operating system.

DSCB . Data set control block.

E

EBCDIC. See *extended binary-coded decimal interchange code*.

end user. A person in a data processing installation who requires the services provided by the computer system.

EOR. End of record.

entry-sequenced data set (ESDS). In VSAM, a data set whose records are loaded without respect to their contents and whose RBAs cannot change. Records are retrieved and stored by addressed access, and new records are added at the end of the data set.

ESDS. See *entry-sequenced data set (ESDS)*.

ESS. Enterprise Storage Server.

ETR. Electronic technical response.

exports data set. An MVS file on the server containing entries for directories that can be exported to Network File System clients. It is used by the server to determine which MVS files and prefixes can be mounted by a client, and to write-protect MVS files on the server.

extended binary-coded decimal interchange code (EBCDIC). A coded character set consisting of 8-bit coded characters.

External Data Representation (XDR). A standard developed by Sun Microsystems, Incorporated for representing data in machine-independent format. XDR is a vendor independent way of representing the data. By using the XDR standard data representation convention, systems do not have to understand and translate every data format that exists on the network; there is only the one convention. Data is translated into XDR format before it is sent over the network and, at the reception point, is translated into the data convention used there. This means that new computer architectures can be integrated into the network without requiring the updating of translation routines. The new architecture simply includes a routine that translates its data format into XDR format and the new member of the network is ready to go.

Using XDR, data can be accessed or exchanged among machines of various hardware and software architectures without any translation or interpretation problems. Word lengths, byte ordering, and floating point representations appear to be the same to all nodes in the network.

F

FDDI. Fiber Distributed Data Interface (FDDI).

file. A collection of information treated as a unit. In non-UNIX System Services MVS environments, the terms *data set* and *file* are generally equivalent and are sometimes used interchangeably. See also *data set*.

file handle. A file handle is used by the client and server sides of the Network File System to specify a particular file or prefix.

file system. In the z/OS UNIX HFS environment, the collection of files and file management structures on a physical or logical mass storage device, such as a diskette or minidisk. See also *HFS data set*.

File Transfer Protocol (FTP). A TCP/IP protocol used for transferring files to and from foreign hosts. FTP also provides the capability to access directories.

FMID. Function modification identifier.

FUB. File usage block.

foreign host. Any host on the network other than the local host.

free space. Space reserved within the control intervals of a key-sequenced data set for inserting new records into the data set in key sequence or for lengthening records already there; also, whole control intervals reserved in a control area for the same purpose.

FTP. File Transfer Protocol.

G

gateway. A functional unit that interconnects two computer networks with the different network

architectures. A gateway connects networks or systems of different architectures. A bridge interconnects networks or systems with the same or similar architectures.

GID. See *group number (GID)*.

group. (1) With respect to partitioned data sets, a member and the member's aliases that exist in a PDS or PDSE, or in an unloaded PDSE. (2) A collection of users who can share access authorities for protected resources.

group number (GID). A unique number assigned to a group of related users. The group number can often be substituted in commands that take a group name as an argument.

GXB . Global exit block.

H

handle. (1) In the Advanced DOS and OS/2 operating systems, a binary value created by the system that identifies a drive, directory, and file so that the file can be found and opened. (2) In the AIX operating system, the data structure that is a temporary local identifier on an object. Allocating a handle creates it. Binding a handle makes it identify an object at a specific location. (3) In the OS/400 application programming interfaces, a variable that represents an object.

HFS. See *hierarchical file system (HFS)*.

hierarchical file system (HFS) data set. A data set that contains a POSIX-compliant hierarchical file system, which is a collection of files and directories organized in a hierarchical structure, that can be accessed using the UNIX System Services MVS facilities.

host. A computer connected to a network that provides an access method to that network. A host provides end-user services.

I

ICF. See *integrated catalog facility (ICF)*.

ICP. Interconnect Control Program (ICP).

IDCAMS. Integrated catalog access methods services.

integrated catalog facility (ICF). In the Data Facility Product (DFP), a facility that provides for integrated catalog facility catalogs.

Internet. A specific internetwork that includes ARPANET, MILNET, and NSFnet. These networks use the TCP/IP protocol suite. See also *internetwork*.

Internet Protocol (IP). The TCP/IP layer between the higher-level host-to-host protocol and the local network protocols. IP uses local area network protocols to carry packets in the form of diagrams to the next gateway or destination host.

internetwork. A collection of packet-switched networks that are connected by gateways. They work as a single network.

interoperability. The ability of hardware and software from different vendors to communicate on a network.

inter-process communication (IPC). Ways for programs to communicate data to each other and to synchronize their activities. Semaphores, signals, and internal message queues are common methods of inter-process communication.

IP. Internet Protocol.

IPC. See *inter-process communication (IPC)*.

J

JCL. See *job control language*.

Job control language (JCL). A problem-oriented language used to identify the job or describe its requirements to an operating system.

K

key-sequenced data set (KSDS). A VSAM data set whose records are loaded in key sequence and controlled by an index. Records are retrieved and stored by keyed access or by addressed access, and new records are inserted in the data set in key sequence because of free space allocated in the data set. Relative byte addresses of records can change because of control interval or control area splits.

KSDS. See *key-sequenced data set (KSDS)*.

L

LFS. Logical file system.

local host. The computer to which a user's terminal is directly connected.

local network. That portion of a network physically connected to the host without intermediate gateways.

M

management class. A collection of management attributes, defined by the storage administrator, used to control the release of allocated but unused space; to control the retention, migration, and backup of data

sets; to control the retention and backup of aggregate groups, and to control the retention, backup, and class transition of objects.

master catalog.   A key-sequenced data set or file with an index containing extensive data set and volume information that VSAM requires to locate data sets or files, to allocate and deallocate storage space, to verify the authorization of a program or operator to gain access to a data set or file, and to accumulate usage statistics for data sets or files.

MBCS.   Multiple byte character set.

migration level 1.   DFSMShsm-owned DASD volumes that contain data sets migrated from primary storage volumes. The data can be compressed. See also *storage hierarchy*. Contrast with *primary storage, migration level 2*.

migration level 2.   DFSMShsm-owned tape or DASD volumes that contain data sets migrated from primary storage volumes or from migration level 1 volumes. The data can be compressed. See also *storage hierarchy*. Contrast with *primary storage, migration level 1*.

mkdir.   An AIX, UNIX, OS/2 and DOS command used to make a new directory.

mount.   A host-linked operation which results in a tape cartridge being physically inserted into a tape drive.

mount handle data set.   A data set used to store the file handles of mount points.

mount point.   A directory established in a workstation or a server local directory that is used during the transparent accessing of a remote file.

mtime.   The time of last data modification.

MTU.   Maximum transmission unit (MTU).

MVS.   Multiple virtual system.

MVS/ESA.   Multiple Virtual Storage/Enterprise Systems Architecture. A z/OS operating system environment that supports ESA/390.

MVS/ESA SP.   An IBM licensed program used to control the z/OS operating system. MVS/ESA SP together with DFSMS compose the base MVS/ESA operating environment. See also *z/OS*.

mvslogin.   A client command to connect your workstation to MVS.

mvslogout (or mvslogut).   A client command to break the connection between your workstation and MVS.

N

network.   (1) An arrangement of nodes and connection branches. (2) A configuration of data processing devices and software connected for information interchange.

NFS.   See *z/OS Network File System*.

NLM.   Network Lock Manager.

NSM.   Network Status Manager.

null credentials.   Null credentials are usually associated with diskless workstations where it is impossible to obtain identifying information because there is no local (that is, local to the workstation) repository of information.

O

object storage hierarchy.   A hierarchy consisting of objects stored in DB2 table spaces on DASD, on optical or tape volumes that reside in a library, and an optical or tape volumes that reside on a shelf. See also *storage hierarchy*.

optical volume.   Storage space on an optical disk, identified by a volume label. See also *volume*.

OpenEdition MVS.   See *z/OS UNIX System Services.*

OpenEdition MVS file system.   See *z/OS UNIX file system.*

optical volume.   Storage space on an optical disk, identified by a volume label. See also *volume*.

Operating System/2 (OS/2).   An operating system used in IBM PC AT, PS/2, and compatible computers.

OSA2.   Open System Adapter 2 (OSA2).

P

partitioned data set (PDS).   A data set on direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

partitioned data set extended (PDSE).   A system-managed data set that contains an indexed directory and members that are similar to the directory and members of partitioned data sets. A PDSE can be used instead of a partitioned data set.

PDS.   See *partitioned data set (PDS)*.

PDS directory.   A set of records in a partitioned data set (PDS) used to relate member names to their locations on a DASD volume.

PDSE.   See *partitioned data set extended (PDSE)*.

permission code.   A three-digit octal code, or a nine-letter alphabetic code, indicating the access permissions. The access permissions are read, write, and execute.

permission field.   One of the three-character fields within the permissions column of a directory listing indicating the read, write, and run permissions for the file or directory owner, group, and all others.

PFS.   Physical file system.

portable operating system interface (POSIX). Portable operating system interface for computer environments. An IEEE operating system standard, closely related to the UNIX system (software writing).

port.   (1) An access point for data entry or exit. (2) A receptacle on a device to which a cable for another device is attached.

Portmapper.   A server that converts RPC program numbers into port numbers acceptable to the protocol. This server must be running to make RPC calls.

POSIX .   See *portable operating system interface (POSIX).*

primary storage.   A DASD volume available to users for data allocation. The volumes in primary storage are called primary volumes. See also *storage hierarchy.* Contrast with *migration level 1, migration level 2.*

protocol.   (1) A set of semantic and syntactic rules that determines the behavior of functional units in achieving communication. (2) A specification for the format and relative timing of information exchanged between communicating parties.

PS.   Physical sequential.

PTF.   Program temporary fix.

Q

QSAM.   See *queued sequential access method (QSAM).*

queued sequential access method (QSAM).   An extended version of the basic sequential access method (BSAM). Input data blocks awaiting processing or output data blocks awaiting transfer to auxiliary storage are queued on the system to minimize delays in I/O operations.

R

RACF.   See *Resource Access Control Facility (RACF).*

RBA.   Relative byte address.

relative byte address (RBA).   The displacement of a data record or a control interval from the beginning of the data set to which it belongs; independent of the manner in which the data set is stored.

relative record data set (RRDS).   A type of VSAM data set whose records have fixed or variable lengths, and are accessed by relative record number.

remote procedure call (RPC).   RPC is an independent set of functions used for accessing remote nodes on a network. Using RPC network services, applications can be created in much the same way a programmer writes software for a single computer using local procedure calls. The RPC protocols extend the concept of local procedure calls across the network, which means that distributed applications can be developed for transparent execution across a network.

Resource Access Control Facility (RACF).   An IBM licensed program that is included in z/OS Security Server and is also available as a separate program for the z/OS and VM environments. RACF provides access control by identifying and verifying the users to the system, authorizing access to protected resources, logging detected unauthorized attempts to enter the system, and logging detected accesses to protected resources.

Resource Measurement Facility (RMF).   An IBM licensed program or optional element of z/OS, that measures selected areas of system activity and presents the data collected in the format of printed reports, system management facilities (SMF) records, or display reports. Use RMF to evaluate system performance and identify reasons for performance problems.

rmdir.   An AIX, UNIX, OS/2 and DOS command used to remove (delete) a directory.

RMF.   See *Resource Management Facility (RMF).*

root.   The user name for the system user with the most authority.

RPC.   Remote procedure call.

RRDS.   Relative record data set.

S

SAF.   System Authorization Facility (SAF).

SDSF.   System Display and Search Facility (SDSF).

sequential file.   A type of MVS file that has its records stored and retrieved according to their physical order within the file. It must be on a direct access volume.

server.   (1) A functional unit that provides shared services to workstations over a network; for example, a file server, a print server, a mail server. (2) On a

network, the computer that contains the data or provides the facilities to be accessed by other computers in the network. (3) A program that handles protocol, queuing, routing, and other tasks necessary for data transfer between devices in a computer system.

sharing. A term used in a computing environment to refer to utilizing a file on a remote system. It is done by mounting the remote file system, then reading or writing files in that remote system.

shell prompt. The character string on an AIX or UNIX command line indicating the system can accept a command (typically the $ character).

showattr. A client command used to display the values of the site, processing, and data set creation attributes.

SMF. See *system management facilities (SMF)*.

SMP/E. See *System Modification Program/Extended (SMP/E)*.

SMS. See *Storage Management Subsystem (SMS)*.

SPF. System productivity facility.

SRM. System Resources Manager (SRM).

SSF. Software support facility.

stale file handle. A file handle is stale when the file handle for a file or prefix is no longer valid.

stateless. A stateless server can function correctly without maintaining any protocol state information about any of its clients. The NFS protocol is intended to be as stateless as possible. This avoids complex crash recovery; a client just resends requests until a response is received. The stateless protocol is chosen to minimize the probability of data losses due to a server crash.

storage hierarchy. An arrangement of storage devices with different speeds and capacities. The levels of the storage hierarchy include main storage (memory, DASD cache), primary storage (DASD containing uncompressed data), migration level 1 (DASD containing data in a space-saving format), and migration level 2 (tape cartridges containing data in a space-saving format). See also *primary storage, migration level 1, migration level 2, object storage hierarchy*.

Storage Management Subsystem (SMS). A DFSMS facility used to automate and centralize the management of storage. Using SMS, a storage administrator describes data allocation characteristics, performance and availability goals, backup and retention requirements, and storage requirements to the system through data class, storage class, management class, storage group, and ACS routine definitions.

system management facilities (SMF). A component of z/OS that collects input/output (I/O) statistics, provided at the data set and storage class levels, which helps you monitor the performance of the direct access storage subsystem.

superuser. The user who can operate without the restrictions designed to prevent data loss or damage to the system (User ID 0).

System Modification Program/Extended. Basic tool for installing software changes in programming systems. It controls these changes at the element (module or macro) level, which helps protect system integrity.

T

tape volume. Storage space on a tape, identified by a volume label, which contains data sets or objects and available free space. A tape volume is the recording space on a single tape cartridge or reel. See also *volume*.

task control block (TCB). An MVS control block that describes an asynchronous process that is called a task.

TCB. See *task control block (TCB)*.

TCP/IP. See *Transmission Control Protocol/Internet Protocol (TCP/IP)*.

TIOT. Task input/output table.

transmission control block (TCB). An internal control block within the TCP/IP address space.

Transmission Control Protocol (TCP). A stream communication protocol that includes error recovery and flow control.

Transmission Control Protocol/Internet Protocol (TCP/IP). The two fundamental protocols of the Internet protocol suite. The abbreviation TCP/IP is frequently used to refer to this protocol suite. TCP/IP provides for the reliable transfer of data, while IP transmits the data through the network in the form of datagrams. Users can send mail, transfer files across the network, or execute commands on other systems.

TSO. Time-sharing option.

TSO/E. Time-sharing option extended.

U

UCB. Unit control block.

UDP. User datagram protocol.

UID. The DFSMShsm authorized-user identification in 1 to 7 characters. See also *user number (UID)*.

Unicode Standard. A universal character encoding standard that supports the interchange, processing, and display of text that is written in any of the languages of the modern world. It can also support many classical and historical texts and is continually being expanded. The Unicode Standard is compatible with ISO/IEC 10646.

UNIX. A highly portable operating system originally developed by Bell Laboratories that features multiprogramming in a multi-user environment. UNIX is implemented in the C language. UNIX was originally developed for use on minicomputers but has been adapted on mainframes and microcomputers. It is especially suitable for multiprocessor, graphics, and vector-processing systems.

UNIX authentication. Requires a client process to send credentials (the client's machine-name, uid, gid, and group-access-list) to the server.

user catalog. An optional catalog used in the same way as the master catalog and pointed to by the master catalog. It lessens the contention for the master catalog and facilitates volume portability.

User Datagram Protocol (UDP). A connectionless datagram protocol that requires minimal overhead, but does not guarantee delivery.

user number (UID). In the AIX operating system, a number that uniquely identifies a user to the system. It is the internal number associated with a user ID.

V

virtual storage access method (VSAM). An access method for direct or sequential processing of fixed and variable–length records on direct access storage devices. The records in a VSAM file can be organized in logical sequence by a key field (key sequence), in the physical sequence in which they are written on the file (entry sequence), or by relative record number.

volume. The storage space on DASD, tape, or optical devices, which is identified by a volume label. See also *DASD volume, optical volume, tape volume*.

VRA. Variable Recording Area (VRA).

VSAM. See *virtual storage access method (VSAM)*.

VSCR. Virtual storage constraint relief.

X

XDR. See *External Data Representation (XDR)*.

Z

z/OS. z/OS is a network computing-ready, integrated operating system consisting of more than 50 base elements and integrated optional features delivered as a configured, tested system. See also *MVS/ESA SP*.

z/OS Network File System. A base element of z/OS, that allows remote access to z/OS host processor data from workstations, personal computers, or any other system on a TCP/IP network that is using client software for the Network File System protocol.

z/OS UNIX System Services file system. In the z/OS UNIX environment, the collection of files and file management structures on a physical or logical mass storage device, such as a diskette, minidisk, or a disk drive. See also *hierarchical file system (HFS) data set*.

z/OS UNIX System Services (z/OS UNIX). The set of functions provided by the SHELL and UTILITIES, kernel, debugger, file system, C/C++ Run-Time Library, Language Environment, and other elements of the z/OS operating system that allow users to write and run application programs that conform to UNIX standards.

Index

Numerics

Readers' Comments — We'd Like to Hear from You

z/OS
Network File System
Performance Tuning Guide

Publication No. SC26-7418-02

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

Company or Organization

Phone No.

IBM®

Fold and Tape   **Please do not staple**   Fold and Tape
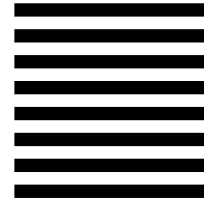
NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL   PERMIT NO. 40   ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Buisness Machines Corporation
RCF Processing Department
M86/050
5600 Cottle Road
San Jose, CA U.S.A.   95193-0001

Fold and Tape   **Please do not staple**   Fold and Tape

IBM®

Program Number: 5694-A01