DFSORT

# Tuning Guide

*Release 14*

SC26-3111-02

DFSORT

# Tuning Guide

*Release 14*

SC26-3111-02

**Note!**

Before using this information and the product it supports, be sure to read the general information under “Notices” on page ix.

**Third Edition (September 1998)**

| This edition replaces and makes obsolete the previous edition, SC26-3111-01. The technical changes for this edition
| are summarized under ″Summary of Changes″, and are indicated by a vertical bar to the left of a change.

| This edition applies to Release 14 of DFSORT, 5740-SM1, and to any subsequent releases until otherwise indicated
| in new editions. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Corporation
RCF Processing Department
G26/050
5600 Cottle Road
San Jose, CA 95193-0000
U.S.A.

| Or, you can send us comments about this book electronically:
| IBMLink from US and IBM Network: STARPUBS at SJEVM5
| IBMLink from Canada: STARPUBS at TORIBM
| IBM Mail Exchange: USIB3VVD at IBMMAIL
| Internet: starpubs@sjevm5.vnet.ibm.com or, starpubs at sjevm5.vnet.ibm.com
| Fax (US): 1–800–426–6209

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

# Contents

# Figures

# Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. 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 of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594, U.S.A.

## Programming Interface Information

This book primarily documents information that is NOT intended to be used as a Programming Interface of DFSORT.

This book also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of DFSORT. This information is identified where it occurs, either by an introductory statement to a chapter or by the following marking:

**Programming Interface information**

Programming Interface information

**End of Programming Interface information**

## Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

DB2
DFSMS/MVS
DFSORT
ESA/390
Enterprise Systems Architecture/370
Enterprise Systems Architecture/390
ESCON
GDDM
Hipersorting
Hiperspace
IBM
MVS/ESA
NetView
OS/390
RACF
Resource Measurement Facility
RMF
SmartBatch
System/370
3090

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of other companies.

Sorting is one of the most frequently used functions at most data processing sites.

This book provides information about tuning IBM DFSORT Release 14, offering suggestions for reducing its use of system resources and achieving better turnaround time for the many applications that use DFSORT without adversely affecting system performance.

This book is intended for systems engineers, performance analysts, system programmers, and application programmers, who have some experience with DFSORT. For those unfamiliar with DFSORT, *DFSORT Installation and Customization* (SC33-4034) and *DFSORT Application Programming Guide* (SC33–4035) are the prerequisites for reading this book.

In this book, one megabyte (MB) is equal to 1024 kilobytes (KB), which is equal to 1048576 bytes.

## Performance Comparisons

The performance comparisons shown in this book are derived from a single run of each component of the comparison in a controlled environment. The applications used are not meant to be representative of any particular user's environment. The performance comparisons shown are examples of the effects of various tuning techniques in particular situations.

Unless otherwise stated, applications were run:

- On an IBM 9672 Model R73 with 1GB of central storage and 1GB of expanded storage, in a stand-alone OS/390 V2R5 environment. All of the input, output, and work data sets resided on IBM RAMAC 9391-A31 DASD connected to an IBM 3990-6 control unit. 1GB of storage controller cache was available for Cache Fast Write usage.
- With DFSORT Release 14 using the IBM-supplied installation defaults.
- Using 150MB input data sets with RECFM=FB, LRECL=160, BLKSIZE=27840 and 20-byte randomly generated keys.

The actual performance of a particularsort application is dependent on many factors including record length, data set size, region size, total central and expanded storage available, type and number of auxiliary storage devices, and specific editing functions and exitsused.

CPU time represents the sum of the following five fields: TCB, SRB, RCT, HPT, and IIP. See “Processor Utilization” on page 4 for a description of these fields.

Elapsed time results for sorting in a multi-tasking environment are application profile and workload dependent. Therefore, the results might differ from user to user.

IBM does not represent nor warrant that your applications will achieve the same performance data as the examples in this book.

## About This Book

This manual contains the following sections:

- "Chapter 1. Introduction" on page 1, describes the importance of tuning DFSORT, an example of successful tuning, system resources and tuning of DFSORT, and the performance indicators for DFSORT.
- "Chapter 2. DFSORT Performance Features" on page 7, describes the performance features of DFSORT.
- "Chapter 3. Environment Considerations" on page 17, describes the relationship between DFSORT and the environment in which it runs including the storage hierarchy, virtual storage, and system-managed storage.
- "Chapter 4. Installation Considerations" on page 25, describes how installation options, run-time options, DFSORT capabilities, installation defaults, site-wide options, and installation exits affect the performance of DFSORT applications.
- "Chapter 5. Run-Time Considerations" on page 41, describes DFSORT features and how to use them to improve the run-time performance of DFSORT.
- "Chapter 6. Application Considerations" on page 57, describes how new and existing applications can make efficient and effective use of the DFSORT facilities.
- "Chapter 7. DFSORT Performance Data" on page 69, describes the actions you can take to tune DFSORT, the type and location of information you need to tune DFSORT, and the methods you can use to collect the information.
- "Appendix A. Sample ICETEXIT" on page 85, includes sample source code for an ICETEXIT routine which creates a summary performance record each time DFSORT is used, and a SVC to write the resulting user records to SMF.
- "Appendix B. Estimating Elapsed Time" on page 95, shows how you can calculate a rough estimate of the best possible elapsed time for a particular sort application.

## Related Publications

The information presented in the following publications can help you to use the options and facilities of DFSORT.

## DFSORT Library

*DFSORT Tuning Guide* is a part of a more extensive DFSORT library. These books can help you work with DFSORT more effectively.

| Task | Publication | Order Number |
|---|---|---|
| Planning for and customizing DFSORT | *DFSORT Installation and Customization Release 14* | SC33-4034 |
| Learning about DFSORT | *Getting Started with DFSORT Release 14* | SC26-4109 |
| Application programming | *DFSORT Application Programming Guide Release 14* | SC33-4035 |
| Interpreting messages and codes, and diagnosing failures. | *DFSORT Messages, Codes and Diagnosis Guide Release 14* | SC26-7050 |
| Quick reference | *DFSORT Reference Summary Release 14* | SX33-8001 |
| Learning to use DFSORT panels | *DFSORT Panels Guide* | GC26-7037 |

You can order a complete set of DFSORT publications with the order number SBOF-1243, except for *DFSORT Licensed Program Specifications* (GC33-4032), which must be ordered separately.

## DFSORT Library Softcopy Information

A softcopy version of the DFSORT library is available on two CD-ROMs as shown in the table that follows. Each of the CD-ROMs contains all of the DFSORT books for Release 13 and Release 14 with the exception of the *DFSORT Reference Summary*.

| Order Number | Title |
|---|---|
| SK2T-6700 | *IBM Online Library: OS/390 Collection* |
| SK2T-0710 | *IBM Online Library: MVS Collection* |

## OS/390 Publications

For up-to-date descriptions of all of the books that support OS/390, refer to the *OS/390 Information Roadmap*, (GC28–1727).

## Storage Management Library (SML)

| Task | Publication | Order Number |
|---|---|---|
| Locating the appropriate storage management publication | *Storage Management Reader's Guide* | GC26-4403 |
| Learning about storage management | *Focus on Storage Management* | GC26-4404 |
| Leading a storage administration group | *Leading an Effective Storage Administration Group* | SC26-4405 |
| Migrating to system-managed storage | *Storage Management Subsystem Migration Planning Guide* | SC26-4406 |
| Managing storage pools | *Managing Storage Pools* | SC26-4407 |
| Managing data sets | *Managing Data Sets* | SC26-4408 |
| Evaluating current storage configurations | *Configuring Storage Subsystems* | SC26-4409 |

## Storage Subsystem Library (SSL)

| Task | Publication | Order Number |
|---|---|---|
| Locating information within the SSL | *Storage Subsystem Library Master Bibliography, Index, and Glossary* | GC26-4496 |
| Evaluating and learning about the 3990 storage control unit | *IBM 3990 Storage Control Introduction* | GA32-0098 |
| Evaluating and learning about the 3390 DASD | *IBM 3390 Direct Access Storage Introduction* | GC26-4573 |
| Evaluating and learning about the RAMAC array DASD | *IBM RAMAC Array DASD Introduction* | GC26-7012 |
| Planning for, installing, and administering storage with the 3990 storage control unit | *IBM 3990 Storage Control Planning, Installation, and Storage Administration Guide* | GA32-0100 |
| Using the 3390 in an MVS environment | *Using the IBM 3390 Direct Access Storage in an MVS Environment* | GC26-4574 |
| Using the RAMAC array DASD in an MVS, VM, and VSE environment | *Using the IBM RAMAC Array DASD in an MVS, VM, and VSE Environment* | GC26-7013 |

| Task | Publication | Order Number |
|---|---|---|
| Reference for the 3990 storage control unit | *IBM 3990 Storage Control Reference* | GA32-0099 |
| Diagnosing and correcting disk media errors Maintaining storage subsystems | *Maintaining IBM Storage Subsystem Media* | GC26-4495 |

## Other Documentation

| Task | Publication | Order Number |
|---|---|---|
| Assembler programming for IBM processors | *IBM Enterprise Systems Architecture/390 Principles of Operation* | SA22-7201 |
| | *IBM Enterprise Systems Architecture/370 Principles of Operation* | SA22-7200 |
| Application programming with COBOL | *VS COBOL II Application Programming Guide* | SC26-4045 |
| | *COBOL for MVS & VM LPS* | GC26-4761 |
| | *COBOL for OS/390 & VM V2.1 Programming Guide* | SC26-9049 |

## Referenced Publications

| Short Title | Publication | Order Number |
|---|---|---|
| Analyzing RMF Reports | *Analyzing RMF Monitor I and II Reports* | LY28-1007 |
| Application Programming Guide | *DFSORT Application Programming Guide Release 14* | SC33-4035 |
| COBOL Application Programming Guide | *VS COBOL II Application Programming Guide* | SC26-4045 |
| SmartBatch/MVS | *SmartBatch V1R1 Overview* | GC28-16227 |
| | *SmartBatch V1R1 Users Guide and Reference* | GC28-1640 |
| Diagnosis Guide | *DFSORT Messages, Codes and Diagnosis Guide Release 14* | SC26-7050 |
| Extended Addressability | *MVS/ESA Application Development Guide: Extended Addressability* (for MVS/ESA SP Version 4) | GC28-1652 |
| | *OS/390 V2 R4.0 MVS Extended Addressability Guide* | GC28-1769 |
| Getting Started | *Getting Started with DFSORT Release 14* | SC26-4109 |
| Initialization and Tuning | *MVS/ESA Initialization and Tuning Reference* (for MVS/ESA SP Version 5) | GC28-1635 |
| | *OS/190 V2 R5.0 MVS Initialization and Tuning Reference* | SC28-1752 |
| Installation and Customization | *DFSORT Installation and Customization Release 14* | SC33-4034 |
| | *DFSORT Panels Guide* | GC26-7037 |
| Installation Exits | *MVS/ESA Installation Exits* (for MVS/ESA SP Version 4) | GC28-1637 |
| | *OS/390 V2 R5.0 MVS Installation Exits* | SC28-1753 |

| Short Title | Publication | Order Number |
|---|---|---|
| Messages and Codes | *DFSORT Messages, Codes and Diagnosis Guide* | SC26-7050 |
| Performance Information | *Cache Performance Management* | GC66-3214 |
| | *Performance Management in a DFSMS/MVS World* | GC66-3252 |
| 3390 DAS Introduction | *IBM 3390 Direct Access Storage Introduction* | GC26-4573 |
| *MVS/ESA System Programming Library: System Modifications* (for MVS/ESA SP Version 3) | GC28-1831 | |
| System Management Facilities | *MVS/ESA System Management Facilities (SMF)* (for MVS/ESA Version 4) | GC28-1628 |
| | *OS/390 V2 R5.0 MVS System Management Facilities (SMF)* | GC28-1783 |

# Summary of Changes

## Third Edition, September 1998

## New Programming Support for Release 14

### Symbols for Fields and Constants

DFSORT now provides a simple and flexible method for using symbols in DFSORT and ICETOOL statements. You can define and use a symbol for any field or constant that is recognized in a DFSORT control statement or ICETOOL operator. This makes it easy to create and reuse collections of symbols (that is, mappings) for your frequently used data.

In addition, you can obtain and use collections of DFSORT symbols created specifically for data associated with other products (for example, RACF, DFSMSrmm and DCOLLECT) or by your site.

DFSORT symbols can increase your productivity by automatically providing the positions, lengths and formats of the fields and the literals, numbers, and bit flags of the constants, associated with the particular records you are processing with DFSORT or ICETOOL.

### Improvements in Performance, Capacity and Storage Usage

Blockset copy and merge applications can now use storage above 16MB virtual, providing improved performance and virtual storage constraint relief.

Blockset copy and merge modules will now reside above 16MB virtual, providing virtual storage constraint relief.

DFSORT can now handle a significantly larger number of INCLUDE and OMIT conditions.

DFSORT can now handle a significantly larger number of SUM fields.

The upper limit for the number of JCL and dynamically allocated work data sets that can be specified and used by DFSORT's Blockset technique has been raised from 100 to 255. The use of more work data sets increases the maximum amount of data DFSORT can process in a single sort application. Any valid ddname of the form SORTWKdd or SORTWKd can now be used for DASD work data sets (for example, SORTWK01, SORTWKC3, SORTWK2, SORTWK#5, SORTWKA, SORTWKXY and so on).

The upper limit for the number of input data sets that can be specified and used for a Blockset merge application has been raised from 16 to 100. The use of more merge input data sets increases the maximum amount of data DFSORT can process in a single merge application.

### Time-of-Day Option Controls

New time-of-day installation modules (ICETD1-4) allow different sets of installation defaults to be used, based on the day and time DFSORT applications run. Each environment installation module (ICEAM1-4) can enable one or more time-of-day installation modules. This capability allows new levels of control for installation defaults. For example, larger storage, hiperspace and data space limits could be used only for batch program-invoked DFSORT applications that run off-shift during the week, and all weekend.

## Repackaging

The product has been repackaged to simplify installation and customization:

- IBM's DFSORT, DFSMSdfp, and MVS/DFP teams have simplified the process of replacing IEBGENER with ICEGENER. You now only need to apply a DFSMS or DFP PTF that supplies an alias of ″IEBGENR″ for IEBGENER and place ICEGENER with an alias of ″IEBGENER″ ahead of IEBGENER in the system's search order for programs.
- The number of FMIDs has been reduced from 10 to 3.
- The number of libraries required to install DFSORT has been reduced from 40 to 26.
- DFSORT R14 now supports a single installation of the product for both resident and nonresident features. This allows you to decide how to use DFSORT independent of the installation method, thus reducing the number of decisions you have to make at installation time.
- All FMIDs in DFSORT R14 can be installed together, including the FMIDs for both English and Japanese messages and panels.

## OUTFIL Processing Enhancements

OUTFIL now supports creation of multiple output records using the fields of the input record. This allows you to split each record into pieces, include a field in more than one record, include different fields in different records, and more.

OUTFIL now supports processing of variable-length input records which are too short to contain all specified OUTFIL OUTREC fields. OUTFIL's new VLFILL=byte operand can be used to replace missing bytes in OUTFIL OUTREC fields with the specified fill byte so the filled fields can be processed.

## ICETOOL Enhancement

A new DISCARD(savedd) operand of ICETOOL's SELECT operator allows you to save the records that are not selected, in the savedd data set. Thus, in one pass, you can create an outdd data set with the records that meet your specified criteria, and a savedd data set with the records that do not meet your specified criteria. DISCARD(savedd) can be used to save the records discarded by ALLDUPS, NODUPS, HIGHER(x), LOWER(y), EQUAL(v), FIRST or LAST.

## Installation and Run-Time Option Enhancements

A new p% value for the EXPRES, EXPOLD, and EXPMAX installation options and the HIPRMAX installation and run-time options is now available. p% can be used to vary the limit DFSORT calculates for the corresponding option as a percentage of the configured expanded storage on the system at run time. If the configured expanded storage on a system changes, p% will cause a corresponding change in the run-time limit calculated for the corresponding option. When sharing DFSORT installation options between systems, such as in a sysplex, p% can be used to tailor the limit DFSORT calculates for the corresponding option to the system on which the application runs.

A new SPANINC installation and run-time option allows you to specify what you want DFSORT to do if it detects incomplete spanned records. This gives you control over the action (continue by eliminating incomplete spanned records and recovering valid records, or terminate), type of message (informational or error) and return code (0, 4 or 16) for incomplete spanned records.

A new OVFLO installation and run-time option allows you to specify what you want DFSORT to do when BI, FI, PD or ZD summary fields overflow. This gives you

control over the action (continue or terminate), type of message (informational or error) and return code (0, 4 or 16) for summary overflow.

A new PAD installation and run-time option allows you to specify what you want DFSORT to do when the SORTOUT LRECL is larger than the SORTIN/SORTINnn LRECL. This gives you control over the action (continue or terminate), type of message (informational or error) and return code (0, 4 or 16) for LRECL padding.

A new TRUNC installation and run-time option allows you to specify what you want DFSORT to do when the SORTOUT LRECL is smaller than the SORTIN/SORTINnn LRECL. This gives you control over the action (continue or terminate), type of message (informational or error) and return code (0, 4 or 16) for LRECL truncation.

The IBM-supplied default for ICEMAC option DSA has been changed from 16MB to 32MB.

The IBM-supplied default for ICEMAC option GENER has been changed from IEBGENER to IEBGENR.

The maximum value for ICEMAC option OVERRGN has been changed from 64KB to 16128KB.

### Other Enhancements

New messages ICE178I and ICE179A provide information about reallocation of VIO work data sets.

The option-in-effect messages (ICE127I-ICE133I) are now printed for Blockset copy and merge applications.

The user exit address constant can now be passed to E32 user exits for Blockset merge applications.

Null segments in variable spanned input records are now processed by DFSORT and no longer result in termination. A null segment means that there are no more segments in the block.

### OS/390 and MVS/ESA Only

DFSORT Release 14 only supports the OS/390 and MVS/ESA environments. MVS/XA and VIRTDSP processing for MVS/XA are no longer supported.

## New Programming Support for Release 13 (PTFs after April, 1996)

### Additional Year 2000 Features

A new Y2S format can order and transform two-digit character or zoned decimal year data according to the century window, while handling binary zeros, blanks and binary ones in the year field as special indicators.

A new Y2B format can order and transform two-digit binary year data according to the century window.

FREE=CLOSE support for DFSPARM makes it possible to override the SORT statements generated by multiple COBOL SORT verbs in the same COBOL program.

### OS/390 Registration

With OS/390 R2 and above, a check is performed to ensure that the DFSORT product is licensed for use, either as a feature of OS/390 or as a separate program product.

## New Programming Support for Release 13 (PTFs - April, 1996)

### Year 2000 Features

New Y2C, Y2Z, Y2P and Y2D formats, in conjunction with a new Y2PAST installation and run-time option, allow you to handle two-digit year data in the following ways:

- Set the appropriate century window for your applications (for example, 1915-2014 or 1950-2049).
- Order two-digit character, zoned decimal, packed decimal or decimal year data according to the century window using Blockset SORT or MERGE (for example, order 96 representing 1996 before 00 representing 2000 in ascending sequence, or order 00 before 96 in descending sequence).
- Transform two-digit character, zoned decimal, packed decimal or decimal year data to four-digit character year data according to the century window using OUTFIL OUTREC (for example, transform 96 to 1996 and 00 to 2000).

A new PD0 format allows you to order and transform parts of packed decimal fields (for example, month and day in date fields) using SORT, MERGE and OUTFIL.

### Performance Improvements for FLR and VLR Blockset Sorts

Performance improvements for FLR and VLR Blockset sorts include the following:

- Dataspace sorting can now be used for variable-length record sort applications.
- DFSORT data processing methods have been improved.
- Dynamic storage adjustment is a new feature that allows DFSORT to automatically use more storage than the TMAXLIM value for a Blockset sort application if DFSORT determines that doing so should improve performance. New installation option DSA=n has been added to enable you to specify the dynamic storage adjustment limit.
- The upper limit for the amount of main storage that can be specified and used by DFSORT has been raised from 32M to 2000M. Specifying more main storage can provide the following benefits:
  - It allows DFSORT to sort very large data sets more efficiently.
  - It allows more sort applications to be done entirely in main storage, eliminating the need for intermediate work space and greatly reducing the EXCP counts for those applications.
  - It increases the maximum amount of data DFSORT can process in a single sort application.
- New installation option IOMAXBF=n has been added to enable you to specify the upper limit for the amount of storage to be used for SORTIN and SORTOUT data set buffers, which in turn limits the amount of data that can be transferred in a single I/O operation.
- The upper limit for the number of JCL and dynamically allocated work data sets that can be specified and used by DFSORT's Blockset technique has been raised from 32 to 100. The use of more work data sets increases the maximum amount of data DFSORT can process in a single sort application.
- Changes to the DFSORT SVC provide caching selection enhancements that improve storage control caching performance, especially for SORTIN and SORTOUT devices.

- DFSORT can now use NOEQUALS for VLR Blockset applications if EQUALS=NO is specified at installation or NOEQUALS is specified at run-time. The use of NOEQUALS can improve performance and is recommended for applications for which the order of records that collate identically need not be preserved from input to output. To minimize migration concerns, the IBM-supplied default for the ICEMAC EQUALS option is the new value VLBLKSET, which is equivalent to EQUALS=YES for VLR Blockset applications and to EQUALS=NO for all other applications.

## Floating Point for SUM

FL format can now be used with the SUM control statement for short (4-byte), long (8-byte) and extended (16-byte) floating point data.

## Security Improvements

Changes to the DFSORT SVC provide security improvements that bring DFSORT up to B1 security standards.

## EXCPVR Processing Removed

To enhance DFSORT's protection of system integrity, EXCPVR processing will no longer be used. EXCPVR parameter values will continue to be accepted, but will have no effect on DFSORT processing. In general, the performance improvements provided by EXCPVR processing have diminished with newer technologies and will be more than offset by the performance improvements listed above. Please ignore any references to EXCPVR in this book; all such references will be deleted when the book is updated.

## New Device Support for Release 13 (PTFs)

The IBM 3590 Magnetic Tape Subsystem is supported for input, output and work data sets.

# Chapter 1. Introduction

This book offers information and recommendations on tuning DFSORT. It offers advice to the system programmer on installing DFSORT and setting its default parameters as well as to the application programmer on improving the performance of individual DFSORT applications. It describes the main indicators used to measure performance and emphasizes the advantages of using DFSORT's Blockset technique for improved performance. Since sort applications tend to be more complex and time-consuming than merge or copy applications, this book concentrates on techniques for improving the performance of sort applications. However, many of these techniques are also appropriate for copy and merge applications.

This book also assumes that DFSORT's most efficient technique is used. See "Blockset Technique" on page 7 for more information on ensuring that Blockset is used.

**Note:** Although a DFSORT "application" can comprise one or more parts of a job, for simplicity's sake the terms "application" and "job" are used interchangeably in this book.

This chapter describes the following:

- The importance of tuning DFSORT
- Examples of successful tuning of DFSORT
- System resources and tuning DFSORT
- The performance indicators for DFSORT

## DFSORT on the World Wide Web

For articles, online books, news, tips, techniques, examples, and more, visit the DFSORT/MVS home page at URL:

`http://www.ibm.com/storage/dfsort/`

## DFSORT FTP Site

You can obtain DFSORT articles and examples via anonymous FTP to:

`index.storsys.ibm.com/dfsort/mvs/`

## The Importance of Tuning

If you are unsure whether your site can benefit from the advice in this book, consider the following questions:

- Do you know how frequently DFSORT is invoked at your site?
- Is there growing pressure to reduce your batch window?
- Are you confident that you and your users are using DFSORT efficiently?
- If better tuning could result in significant savings in elapsed time, CPU time, device connect time, or EXCP counts, would it be worthwhile?

The tuning of DFSORT and the way applications use it is important because sorting and copying are two of the most frequently used functions at OS/390 and MVS/ESA sites. DFSORT applications typically consume from 10 to 25 percent of the total processor resources and 15 to 30 percent of the total I/O channel resources.

### Introduction

More efficient use of DFSORT can lead to:
- Reduced use of system resources
- Reduced job turnaround time
- Improved productivity

All of these benefits can result in cost savings.

Because of the internal optimizations that DFSORT performs (for instance, selecting the best work data set block size for the run), users often do not take the time to tune DFSORT or their applications. Experience shows that many sites only begin to focus on DFSORT performance when there is a crisis: perhaps as data volumes increase, or the batch window becomes smaller. While this book offers help for these situations, it is also intended to help avoid them.

## Examples of Successful Tuning

Some of the practical advice in this book is based on the experience of DFSORT customers.

One customer, a large U.S. retailing organization, was facing increased batch processing times because of growing data volumes. After analysis and subsequent tuning of its 2500 weekly applications that use DFSORT, the organization was able to operate in a smaller batch window with a reduction in machine and system resources used by DFSORT applications. For some applications, the benefits were substantial. They were able to reduce some frequently-used applications from 4 minutes to 30 seconds. The site was able to reduce the elapsed time for its monthly payroll application by more than 3 hours.

Another customer, a worldwide company that provides software to financial institutions, needed to reduce the sort time of an application batch job. After increasing virtual storage from 1MB to 4MB (TMAXLIM or MAINSIZE option) and turning on Hipersorting (HIPRMAX=64), the sort time was reduced from 2.5 hours to 20 minutes.

This book offers many recommendations, ranging from advice on setting the most appropriate option values to guidelines for optimizing virtual storage and work space. You should decide which recommendations are best for your site, based on site requirements and the amount of resources available to implement them.

## System Resources

The purpose of tuning DFSORT is to use system resources more efficiently. This is important at most sites, since there are usually too many demands made for limited resources. Although DFSORT automatically optimizes many of its tuning decisions, there are additional actions which a site and its DFSORT users can take to further improve DFSORT performance. These actions depend on the priority given to various performance objectives.

Different sites and programmers define *efficient performance* in different ways. A site with a primarily batch environment or an application programmer with a single task to complete probably measures performance based on elapsed time. Alternatively, a site experiencing high CPU usage or a programmer who is charged based on CPU time is more likely to evaluate efficiency in terms of reduced CPU time.

While improved performance is the objective of tuning, often it is necessary to compromise. That is, improving the use of one system resource can have a negative impact on other resources, in much the same way that giving more resources to one application can make it perform much better at the expense of degrading the performance of other applications that are competing for the same resources.

The main trade-offs that you should consider are among:

**Processor Load**

Accounting charges are often based on the number of CPU service units used by a job or address space. The more CPU time used, the higher the charges. Reducing a job's CPU time not only reduces these charges, but also enables other jobs competing for the same CPU resource to complete sooner.

**Paging Activity**

System paging activity reflects how the system is managing virtual storage. It is made up of three functions: paging, page movement, and migration, all of which move virtual storage pages from real storage in order to make room for other pages needed by a running program. *Paging* is the process of moving virtual storage pages from central storage to auxiliary storage, and takes a large amount of elapsed time to perform. *Page movement* is the process of moving virtual storage pages between central storage and expanded storage, and requires much less elapsed time to perform than paging. *Migration* is the process of moving pages from expanded storage to auxiliary storage through central storage, and takes the most elapsed time of the three to perform.

System paging activity can increase elapsed time for user programs. If the activity is too high, the system reduces its workload by reducing the number of jobs running, and might suspend processing of some jobs (known as *swapping*) until the paging activity returns to an acceptable level. The result is that the system spends more time managing virtual storage, while many user programs take longer to complete.

**I/O Activity**

Some accounting charges are based on the I/O performed by a job. This is frequently measured by execute channel program (EXCP) counts. While EXCP counts might not represent a completely accurate usage of I/O resources, they are important to many users and sites, and steps can be taken to reduce them. See *Cache Performance Management* and *Performance Management in a DFSMS/MVS World* for more detailed information.

**Elapsed Time**

Elapsed time is likely to be most important for sites with a limited batch window, where processing of particular applications has to be completed within a certain period. It is also important to users whose productivity depends on having their applications complete as soon as possible.

**DASD Utilization**

In environments where DASD is constrained, the amount of auxiliary storage required by an application is of paramount concern. For DFSORT applications, this usually involves the efficient use of output and work data sets.

## Performance Indicators

This section describes how you can measure the performance factors listed above for DFSORT. Often, performance is evaluated as a combination of some or all of them. The priorities given to each depend upon site objectives.

## Processor Utilization

CPU fields are used to measure the amount of work performed by the processor, as opposed to work performed by the I/O and storage subsystems. CPU time consists of the sum of the following five components.

**Task Control Block (TCB)**
The CPU time used to perform user program activity on behalf of user tasks for a job step.

**Service Request Block (SRB)**
The CPU time used to perform system service requests on behalf of user tasks for a job step.

**Hiperspace Processing Time (HPT)**
If the user task reads from or writes to a Hiperspace using the HSPSERV macro, this is the amount of CPU time used to service these reads and writes.

**I/O Interrupt Processing (IIP)**
The CPU time used to handle input/output (I/O) interrupts on behalf of user tasks for a job step.

**Region Control Task (RCT)**
If the user task is swapped out while running, this is the amount of CPU time used to swap the task out and back in again.

## System Paging

To be meaningful, paging and swapping activity measurements are usually associated with particular workloads, or groups of applications that are run simultaneously on a given processor. As such, system paging activity is more a measure of the performance and throughput of the entire system than of the performance of individual applications being run on the system.

Paging activity is often measured by the page-in, page-out, and page-reclaim rates of different address space types, such as Hiperspace, VIO, or non-VIO address spaces. For central storage paging, the migration target is also important; moving pages to auxiliary storage is a lot more expensive in terms of performance than moving pages to expanded storage. For swapping activity, there is also the additional distinction between logical and physical swaps.

System paging and swapping activity data is usually gathered by the Resource Measurement Facility (RMF) or a similar system tool. See *Analyzing RMF Reports* for more information on RMF.

## I/O Activity

This represents the movement of data between the processor and DASD or tape devices. The effective use of I/O resources is important to a site and I/O activity is often an important component of site accounting methodologies.

Because performing input to and output from DASD and tape devices typically takes much longer than manipulating the data in real storage, the amount of I/O

performed is a key component of an application's elapsed time. Therefore, reducing I/O generally improves an application's elapsed time.

I/O activity is primarily measured in the following ways:

**EXCPs**
The number of execute channel program (EXCP) commands issued (logical I/Os)

**SSCHs**
The number of start subchannel (SSCH) commands issued (physical I/Os)

**Device Connect Time**
The amount of time a particular device is dedicated for the I/O transfer

**Channel Usage**
The percentage of time a channel is busy initiating, transferring, or completing the movement of data between a device and the CPU

While EXCPs are often used as a measure of I/O activity, their counts can be extremely misleading. For example, one EXCP can be used to transfer a few bytes or dozens of megabytes! SSCHs measure the number of physical I/Os to a data set and thus can be more useful than EXCPs. A complete analysis of total I/O performance should consider device connect time, channel usage and SSCHs.

## Elapsed Time

Elapsed time refers to the amount of "wall clock" time from initiation to termination of the application. For typical sorting applications, elapsed time is composed primarily of I/O time, with CPU time and I/O queueing time also contributing significantly.

The elapsed time for an application can differ from run to run, depending upon the amount of competition from other applications for system resources. Accurate elapsed time comparisons can be done only if the system has no other applications running.

## DASD Utilization

In certain environments, the DASD space usage is a more important characteristic of an application's performance than CPU time or elapsed time. Inefficient DASD usage is usually measured in terms of the amount of DASD space that is allocated, compared to the amount of DASD actually needed. Frequently, large amounts of DASD space are wasted as a result of inefficient blocking.

# Chapter 2. DFSORT Performance Features

The performance of a DFSORT application is largely determined by the use of a special set of product features. This chapter provides an introduction to the performance features of DFSORT including:

- Blockset technique
- OUTFIL
- Hipersorting
- Sorting with data space
- Dynamic Storage Adjustment
- Cache fast write
- ICEGENER
- Compression
- Striping
- Dynamic allocation of work data sets
- System-determined block size
- IDCAMS BLDINDEX
- DFSORT's Performance Booster for The SAS** System
- SmartBatch Pipes
- VIO in expanded storage

How to use these features to gain the most effective performance from DFSORT is described in "Chapter 4. Installation Considerations" on page 25 and in "Chapter 5. Run-Time Considerations" on page 41.

## Blockset Technique

The Blockset technique is DFSORT's most efficient method for sorting, merging, and copying, using optimized algorithms and using IBM hardware efficiently. DFSORT uses Blockset whenever possible. DFSORT's other techniques, Peerage/Vale and Conventional, are not as efficient as Blockset and are only used when Blockset cannot be used.

The Blockset technique can reduce CPU time, I/O activity, and elapsed time. It is strongly recommended that you remove any obstacles to using Blockset whenever possible. For the purposes of this book, any recommendations to improve performance assume that Blockset is the DFSORT technique used.

It is worth checking to see if Blockset was used for a given application; message ICE143I in the SYSOUT data set shows which technique was used. If ICE143I is not shown or shows that a technique other than Blockset was selected, resubmit the application with a SORTDIAG DD statement (unnecessary if the application already had a SORTDIAG DD statement or if your site has specified installation option DIAGSIM=YES). Additional DFSORT messages and diagnostic information are then shown, including:

- ICE802I, which shows the technique used
- ICE800I or error messages

ICE800I gives a code indicating why Blockset was not used. If the code is 1, one or more error messages are also shown.

**Note:** Blockset messages are suppressed if another technique is selected, unless a SORTDIAG DD statement is present, or installation option DIAGSIM=YES has been specified for your site.

A common reason for not selecting Blockset is the use of work data sets on tape devices. Blockset only supports tape devices for input and output data sets. It is strongly recommended that you use DASD rather than tape for work data sets.

## OUTFIL

OUTFIL is a control statement that allows you to create one or more output data sets for a sort, copy, or merge application with only one pass over an input data set. You can use multiple OUTFIL statements, with each statement specifying the OUTFIL processing to be performed for one or more output data sets.

With a single pass over an input data set, OUTFIL can create:

- Multiple output data sets containing unedited or edited records
- Multiple output data sets containing different ranges or subsets of records. Those records that are not selected for any subset can be saved in another output data set.

OUTFIL has many additional features as well; see *Application Programming Guide* for more details.

### Benefits

Because OUTFIL allows you to create output data sets with only one pass over an input data set, OUTFIL can improve performance. This is especially true for elapsed time and EXCPs when you compare an OUTFIL job to create many output data sets with many non-OUTFIL jobs each creating one output data set.

## Hipersorting

Hipersorting is DFSORT's ability to use Hiperspaces for sorting. When virtual storage (main storage and data space) is smaller than the input data set to be sorted, DFSORT uses temporary intermediate storage to perform the sort. This intermediate storage can be of two types; it can be a Hiperspace or it can be DASD work data sets. DFSORT's use of Hiperspaces for intermediate storage, alone or in combination with work data sets, is called Hipersorting.

See *MVS/ESA Application Development Guide - Extended Addressability* for more detailed information about Hiperspaces and data spaces.

### Benefits

A Hiperspace is a high-performance data space that resides in expanded storage and is backed by auxiliary storage (if necessary). Because Hiperspace data generally resides in expanded storage, it is worth comparing how programs access data from expanded storage and from DASD. Unlike data space pages, Hiperspace pages are never in central storage.

**Expanded Storage**
A program's request for data in expanded storage results in a synchronous or other high-speed transfer into central storage. Upon completion, the program continues.

**DASD Storage**

If the request is for data from DASD, an asynchronous I/O operation is started, the program interrupted, and control returned to the dispatcher to dispatch another program. When the I/O completes, the program waits to be dispatched again, since it is now able to process its data.

A Hiperspace read or write operation involves the transfer of data between expanded storage and central storage, whereas a DASD I/O operation involves the transfer of data between DASD and central storage. Since data movement from expanded storage is faster than from DASD, Hipersorting improves elapsed time. Note that DASD volumes connected to cached controllers can provide significant, but smaller, elapsed time improvements compared to Hipersorting. Hipersorting eliminates the need for work data set I/O, therefore eliminating EXCPs and channel usage for the work devices.

## Operation

In addition to DFSORT, expanded storage can be used by many other applications and system components, such as DB2 hiperpools, VSAM hiperspace, hiperbatch, virtual lookaside facility (VLF), VIO, and the paging subsystem. To avoid overusing expanded storage, which can lead to paging data set space shortages and increased system paging activity, DFSORT uses several safeguards.

To begin with, DFSORT is always aware of the future expanded storage needs of other concurrent Hipersorting applications. A DFSORT application never attempts to back its Hiperspace data with expanded storage that is needed by another Hipersorting application.

Next, DFSORT dynamically determines the amount of available expanded storage throughout the run. “Available” expanded storage is the expanded storage used to back new Hiperspace data. It consists of two types:

- Free expanded storage is expanded storage that is not being used by any application.
- Old expanded storage is expanded storage that is being used by other applications, but whose data has been unreferenced for a long period of time. This time period is long enough that the system migrates the data to auxiliary storage to make room for new hiperspace data.

Knowing both the amount of expanded storage needed by other Hipersorting applications and the amount of available expanded storage, along with the values of DFSORT installation options EXPMAX, EXPOLD, and EXPRES, DFSORT can decide at any time in a Hipersorting run to switch from using Hiperspace to using DASD work data sets. This greatly reduces the likelihood of overusing expanded storage, especially as a result of multiple large concurrent Hipersorting applications. It also allows sites to customize their total use of expanded storage by Hipersorting applications through the EXPMAX, EXPOLD, and EXPRES installation options.

EXPMAX limits the total amount of available expanded storage that can be used at any one time by all Hipersorting applications on a system. EXPOLD limits the total amount of old expanded storage that can be used at any one time by all Hipersorting applications on a system. EXPRES reserves a specified amount of available expanded storage for use by non-Hipersorting applications.

Hipersorting can be used in two modes:

- If sufficient expanded storage is available for the entire input data set (plus the required overhead), all of the intermediate data is written to Hiperspace, and DASD work data sets are not needed. This is referred to as Hiperspace-only mode.
- If the above conditions are not met, but sufficient expanded storage is available to provide a performance benefit for the job, then Hiperspace is used in conjunction with DASD work data sets. This is referred to as Hiperspace-mixed mode.

When Hipersorting cannot be used, DFSORT uses DASD work data sets to store its intermediate data, which is referred to as DASD-only mode. Note that Hiperspace-only mode usually provides the best performance when compared to Hiperspace-mixed and DASD-only modes. However, this is not always true for Hiperspace-mixed mode when compared to DASD-only mode. Due to the additional Hiperspace overhead, the use of DASD-only rather than Hiperspace-mixed mode is often more advantageous in terms of performance, and therefore DFSORT may choose not to use Hipersorting.

When making performance comparisons for Hipersorting, remember that the same job sorting the same data set, run on the same system but at different times, may exhibit different performance characteristics. This is due to different modes being used as a result of variations in the system's expanded storage paging activity. These variations can make it difficult to measure the benefits of using Hipersorting for a site. If you are interested in obtaining a realistic measure of the benefits that Hipersorting is providing on your system, you need to concentrate on the combined performance characteristics of a set of sort jobs, rather than on those of individual sorts. For example, you could run a specific set of sort jobs (a workload) repeatedly at different times over the span of a few days and alternately turn on or turn off Hipersorting. The resulting total elapsed times and EXCP counts for all the jobs run with Hipersorting compared to those run without Hipersorting will give you a good measure of the benefits, since this kind of workload measurement does not depend on any single job using Hipersorting.

## Sorting with Data Space

Dataspace sorting is a DFSORT capability that uses data space. DFSORT can use dataspace sorting to improve the elapsed time and CPU time performance for selected sort applications.

## Benefits

A data space is a large contiguous area of virtual storage that is backed by central, expanded, or auxiliary storage, whichever is necessary, as determined by the system. The benefit of sorting with data space is twofold:

- Due to the potentially large size of a data space, dataspace sorting enables a greater percentage of sort jobs to be processed completely within main storage, without the need to write intermediate data to DASD. In-main-storage sorts are usually the most efficient of all sorts. Consequently, dataspace sorting can dramatically reduce CPU and elapsed times, as well as EXCP counts and channel usage.
- If an in-main-storage sort is not possible, dataspace sorting usually picks the optimal amount of virtual storage for its data space. This is frequently larger than the default amount of main storage and enables DFSORT to sort larger amounts of data at a time before writing them to the work data sets. This often provides a significant savings in elapsed time, I/O activity, and CPU time.

Not every sorting application can use dataspace sorting, and, even for those sorts that can use dataspace sorting, it may be more advantageous not to use it under certain circumstances. DFSORT dynamically determines the possible performance benefits of using data space for each run, and chooses not to use dataspace sorting if there is no performance advantage to be gained from it.

### Operation

The use of dataspace sorting is affected by the system's IEFUSI exit, which determines the system's default values for Hiperspaces and data spaces. In addition, dataspace sorting can be controlled with DFSORT's DSPSIZE parameter which can be specified as an installation-wide default through ICEMAC, overridden at run-time with an EXEC PARM option or OPTION control statement, or overridden with an installation-wide ICEIEXIT exit routine.

Dataspace sorting makes heavy use of the system's central storage resources, which can affect system paging. To minimize the impact on the paging subsystem, DFSORT queries the system about paging activity before starting the sorting process. In the case of heavy system activity, dataspace sorting is not used since DFSORT's use of a data space may have an adverse effect on system paging.

See "Chapter 4. Installation Considerations" on page 25 and "Chapter 5. Run-Time Considerations" on page 41 for further details on how best to tune DFSORT's use of data space sorting.

## Dynamic Storage Adjustment

Dynamic Storage Adjustment (DSA) is DFSORT's ability to automatically increase the amount of virtual storage committed to a sort when doing so should significantly improve performance. Like dataspace sorting, DSA lets DFSORT tune the right amount of virtual storage for a sort application.

### Benefits

Increasing the amount of virtual storage available to DFSORT can significantly improve the performance of many sort applications, especially large sorts. The optimal amount of virtual storage varies with the amount of data being sorted. Consequently, the best performance and most efficient use of virtual storage can only be obtained by tuning the virtual storage of each individual sort. DSA does most of this tuning automatically, relieving system and application programmers of the task. By using the optimal amount of virtual storage, DFSORT maximizes its performance while minimizing the impact of DFSORT applications on the system.

### Operation

DFSORT's use of virtual storage from the primary address space is limited by several factors, most notably system (for example, REGION, IEFUSI) and DFSORT (for example, TMAXLIM, SIZE) parameters, defaults, and exits. While DFSORT can never exceed the system limits, those limits usually exceed the virtual storage DFSORT needs for most applications. As a result, the DFSORT limits usually control the amount of virtual storage used by DFSORT.

Dynamic Storage Adjustment permits a range to be specified for the default amount of virtual storage for sorts, provided SIZE/MAINSIZE=MAX is in effect. This range starts with the value specified for TMAXLIM and goes up to the value specified for DSA. For most sorts, DFSORT limits its use of virtual storage to TMAXLIM.

However, for large sorts, DFSORT automatically optimizes performance by increasing the amount of virtual storage it uses, up to but not exceeding the DSA value.

See “Chapter 4. Installation Considerations” on page 25 for further details on the best way to tune DFSORT’s use of DSA.

## Cache Fast Write (CFW)

The term cache fast write (CFW) is normally used to refer to the capability of the cached 3990 storage control units to use cache memory to improve average I/O times. In this book, cache fast write is also used to refer to DFSORT’s ability to take advantage of the storage control unit’s cache fast write function, by writing data sets to and reading data sets from cache only.

### Benefits

You can gain significant elapsed time savings by using cache fast write. The benefits of CFW are twofold:

- When writing data to the work data sets, the write operations can complete at cache speed rather than at DASD speed, as long as overall cache activity permits it. If the cache usage is very heavy, then there may be very little or no benefit to using CFW, since the majority of write operations will be immediately directed to DASD.
- When reading data back from the work data sets, the read operations can complete at cache speed rather than at DASD speed, as long as the data to be read still resides in cache. If the required data does not reside in cache, a DASD access is required and no performance benefit results. The higher the cache activity, the less the performance improvement with CFW, because it is more likely that written data will be destaged to DASD before it can be read back.

### Operation

DFSORT’s use of cache fast write for the work data sets can be controlled with installation option CFW, and run-time option CFW or NOCFW on the DEBUG statement. Note that cache fast write can only be used if the DFSORT SVC has been installed.

## ICEGENER

DFSORT’s ICEGENER facility allows you to more efficiently process IEBGENER jobs. Qualifying IEBGENER jobs are processed by the equivalent, but more efficient, DFSORT copy function. If the DFSORT copy function cannot be used (for example, IEBGENER control statements are specified), control is automatically transferred to the IEBGENER program.

If your site has installed ICEGENER as an automatic replacement for IEBGENER, you need not make any changes to your jobs to use ICEGENER. If your site has not chosen automatic use of ICEGENER, you can use ICEGENER by substituting the name ICEGENER for IEBGENER on any EXEC statement or LINK macro call.

The use of ICEGENER rather than IEBGENER for copy applications can result in significant savings in CPU time and EXCP counts. Figure 1 on page 13 shows an example of the tremendous performance advantages of ICEGENER over IEBGENER.

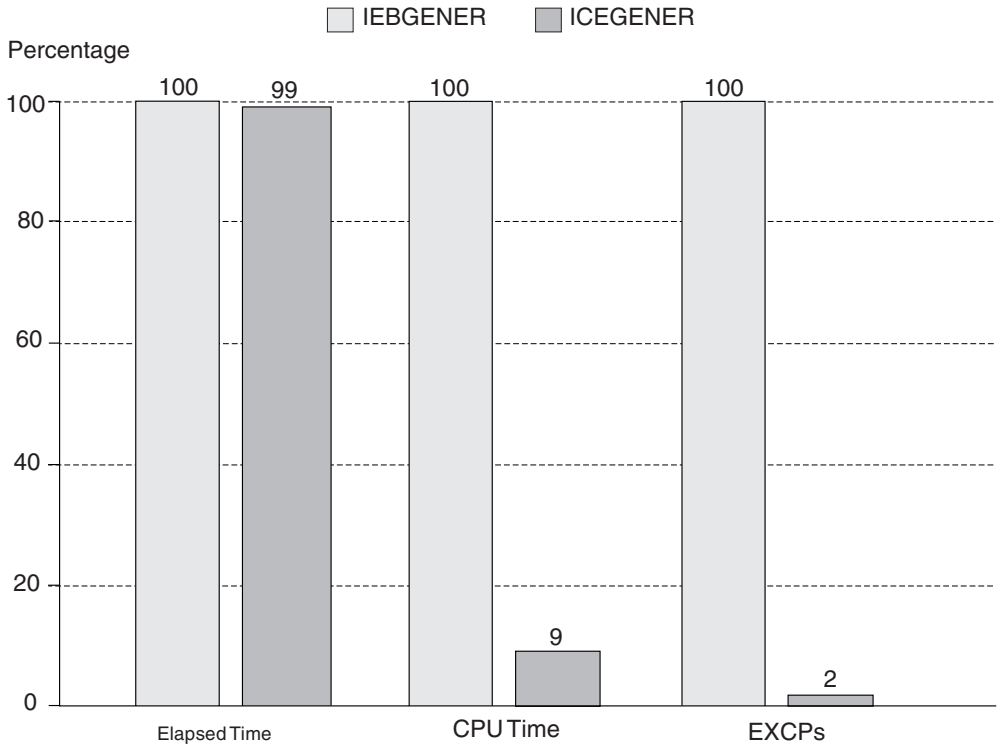



*Figure 1. ICEGENER versus IEBGENER Copy Comparison*

## Compression

You can get significant DASD storage reduction for many types of data by using sequential or VSAM compression. This reduction in DASD storage can result in an equivalent savings in elapsed time for DFSORT to read or write the data. It is recommended that you use compressed sequential or VSAM data sets for input or output as a way to improve elapsed time performance.

## Striping

The use of sequential striping can significantly reduce the elapsed time DFSORT spends reading and writing data. It is recommended that you use sequential striping for your DFSORT input and output data sets as a way to improve elapsed time performance.

## Dynamic Allocation of Work Data Sets

When a sort application cannot be performed entirely in virtual storage, DFSORT must use work space. Dynamic allocation of work data sets makes more efficient use of this work space.

You set the DYNAUTO installation option to control whether dynamic allocation is used automatically, or only when requested by the DYNALLOC run-time option. DYNAUTO also controls whether dynamic allocation or JCL allocation takes precedence when JCL work data sets are specified.

DYNAUTO can be set as follows:

- If DYNAUTO=IGNWKDD, dynamic allocation takes precedence over JCL allocation. If you want the opposite result for selected applications, use the USEWKDD run-time option.
- If DYNAUTO=YES, JCL allocation takes precedence over dynamic allocation. If you want the opposite result, remove all JCL allocation statements.
- If DYNAUTO=NO, dynamic allocation of work data sets is not used unless you specify the DYNALLOC run-time option. JCL allocation takes precedence over dynamic allocation.

Dynamic allocation of work data sets has the following advantages:

- As the characteristics (for example, file size and virtual storage size) of an application change over time, DFSORT can automatically optimize the amount of dynamically allocated work space for the application. This eliminates unneeded allocation of DASD space.
- As the amount of Hiperspace available to the application varies from run to run, DFSORT can automatically adjust the amount of space it dynamically allocates to complement the amount of Hiperspace. This eliminates unneeded allocation of DASD space.
- The amount of work space actually used is often less than the amount allocated. DFSORT tries to minimize dynamic over-allocation while making certain that the application does not fail due to lack of space. With JCL allocation, you could minimize the amount of allocated space manually, but this might require changes to JCL allocation as the characteristics of the application change.

## System Determined Block Size (SDB)

Use the system-determined block size (SDB) facility whenever possible to allow the system to select optimal block sizes for your output data sets. SDB provides better use of output DASD and improved elapsed time. DFSORT's use of SDB can be controlled using installation options SDB and SDBMSG.

## IDCAMS BLDINDEX

DFSORT provides support that enables IDCAMS BLDINDEX to automatically use DFSORT to improve the performance of most BLDINDEX jobs that require BLDINDEX external sorting.

## DFSORT's Performance Booster for The SAS System

DFSORT provides significant CPU time improvements for The SAS System applications. To take advantage of this feature, contact SAS Institute Inc. for details of the support they provide to enable this enhancement.

## SmartBatch Pipes

The use of SmartBatch input and output pipes can significantly reduce elapsed time as a result of the parallelism inherent in piping the data from "writer" to concurrent "reader" jobs. For example, if a SORTOUT data set is piped, DFSORT output processing can be overlapped with the receiving job's input processing. In addition, because a pipe is a virtual storage queue rather than a DASD or tape data set, data transfer time and elapsed time can be reduced significantly.

We recommend using SmartBatch pipes for your DFSORT input and output data sets when appropriate, as a way to improve elapsed time and data transfer time.

Refer to the *SmartBatch V1R1 Overview* and *SmartBatch V1R1 Users Guide and Reference* for more detailed information on using SmartBatch pipe data sets.

## VIO in Expanded Storage

Temporary non-VSAM input and output data sets can be held in expanded storage using Virtual I/O (VIO). Because expanded storage is used rather than a DASD or tape data set, data transfer time and elapsed time can be reduced significantly (provided that the entire data set can fit in the available expanded storage). For example, if a SORTOUT set is allocated as a temporary VIO data set in expanded storage, DFSORT's output processing as well as the receiving job's input processing can show improved performance.

We recommend using VIO in expanded storage for your DFSORT input and output data sets, when appropriate, as a way to improve elapsed time and data transfer time.

Note that VIO is generally not recommended for work data sets.

# Chapter 3. Environment Considerations

While DFSORT automatically optimizes many of its tuning decisions to improve performance, the environment in which it runs affects its overall performance and, therefore, the programs and applications that use DFSORT. DFSORT is designed to take advantage of the major components of IBM computer systems.

The purpose of this chapter is to describe how DFSORT modifies its operation to take advantage of these components and the benefits it derives from them.

The majority of information in this chapter applies only to sorting applications. While copying and merging are also commonly used features of DFSORT, it is sorting that uses the most resources and is the most important to tune.

This chapter includes the following information:

- Storage hierarchy
- Virtual storage
- System-managed storage

## Storage Hierarchy

Within an IBM computer system, data resides across a storage hierarchy. Each level of this hierarchy is represented in hardware by a different component, and the amount of each component in a system is usually variable between some minimum and maximum levels. A typical representation of these components (from top to bottom) would be:

1. Processor cache
2. Central storage
3. Expanded storage
4. Storage control cache
5. DASD
6. Tape

The components at the top of the hierarchy (processor cache, central storage) are somewhat expensive on a per-byte basis (and thus relatively small in capacity), but have very fast access times. In contrast, the components at the bottom of the hierarchy (DASD, tape) are relatively inexpensive and have very large capacities, but their access rates are considerably slower. As you go from top to bottom in the hierarchy, the components typically get less expensive per byte, have higher capacities, and have slower access times.

DFSORT attempts to take advantage of all the levels of the storage hierarchy. The following sections briefly describe how DFSORT accomplishes this.

## Processor Cache

Processor cache is the special high-speed memory from which processors access their instructions and data. This memory has a much faster access rate than central storage, and is an integral part of IBM processors. It contains a copy of those portions of central storage that have been recently referenced. When an instruction or piece of data is needed by the processor and is not in the processor cache, a “cache miss” takes place and the processor waits while a copy of the data is brought into the cache. This results in higher CPU times.

### Environment Considerations

DFSORT is designed to make efficient use of the processor cache by reducing cache misses as much as possible.

## Central Storage

Of all the components that affect DFSORT's performance, central (or main) storage is the most crucial. Sorting is a memory-intensive operation. Without enough central storage to back the virtual storage needs of DFSORT, its performance (as well as the performance of other applications on the system) degrades significantly due to excessive system paging activity.

To sort efficiently, DFSORT needs large amounts of virtual storage. Its needs grow with the amount of data being sorted; a data set four times as large as another requires roughly twice as much virtual storage to sort with the same degree of efficiency. If this virtual storage is not backed by central storage when DFSORT is running, there is a noticeable performance degradation on the system.

Central storage also plays an important role in the use of data space sorting. DFSORT bases its use of its very efficient data space sorting method on the amount of central storage it can use without causing excessive paging on the system. If too much central storage paging would result, DFSORT does not attempt to use data space sorting.

See "Sorting with Data Space" on page 10 for more information about data space sorting.

## Expanded Storage

Expanded storage is an extension to central storage. Its access times are much faster than auxiliary storage (DASD and tape). It is the first resource used to back data residing in Hiperspaces, and is also used to off-load pages from central storage that have not been referenced recently.

DFSORT takes advantage of expanded storage through its use of Hipersorting. With Hipersorting, DFSORT uses a Hiperspace to hold some or all of its intermediate data. This enables some or all of the work data set I/O to be replaced with much faster transfers between central storage and expanded storage.

See "Hipersorting" on page 8 for more information about Hipersorting.

## Storage Control Cache

Certain storage control models, such as the 3990, contain a special high-speed (relative to the DASD) memory known as storage control cache. This cache serves two purposes:

- When reading data from the DASD, if the data is already in the storage control cache, the program can access it directly from the cache without having to wait for the (relatively slow) DASD to retrieve it.
- When writing data to the DASD, by writing directly to the cache (through use of cache fast write), applications can complete their write operations significantly faster than if they had written to the DASD directly.

DFSORT takes advantage of the storage control cache by writing to its work data sets with cache fast write enabled. This speeds up the time needed to write to these work data sets as well as to read back from them.

To benefit from DFSORT's ability to use cache fast write (CFW), ensure that the CFW feature is activated on your storage control units (if appropriate).

DFSORT also takes advantage of the storage control cache by selecting the appropriate caching mode for input and output data sets. This reduces elapsed time for DFSORT applications and also helps other non-DFSORT applications make better use of the cache.

## DASD

In most cases, DFSORT's Blockset technique automatically adjusts itself to take advantage of the geometry of the particular IBM DASD with which it is being run. This is especially true for the work data sets, whose block sizes and distribution of data play crucial roles in the performance of DFSORT.

The location of input, output, and work data sets, as well as the speed of the DASDs on which they reside has a significant effect on the performance of a sort application. For best results, work data sets should be placed on different storage subsystems than the input and output data sets. This helps avoid channel, control unit path, and device contentions. To attain the maximum performance benefit, these data sets (or at least the input and output data sets) should be placed on the fastest DASDs so that DFSORT can take advantage of their speed. Avoid using the 3390 Model 9 for work data sets since this device has slower random access performance than other 3390 devices; it is designed to store large amounts of input and output data sets which are accessed sequentially.

In general, while DFSORT has no control over where its data sets are allocated, it does automatically optimize its access patterns based on data set location to achieve the best possible performance.

When allocating DFSORT work data sets on devices attached to non-synchronous storage control units or connected to ESCON channels, elapsed time may be degraded for certain applications. To avoid this degradation, it is especially important to follow the virtual storage guidelines described in "Virtual Storage Guidelines" on page 49 and to ensure that DFSORT has an accurate knowledge of the size of the data set being sorted. See *Application Programming Guide* for more details about non-synchronous storage subsystem considerations.

In addition to data set location, certain data set characteristics, such as block size, are also very important when considering performance. As mentioned before, DFSORT automatically chooses an optimal block size for its work data sets. Using DFSORT's installation option SDB=YES enables DFSORT to choose optimal block sizes for its output data sets on DASD as well. Of less importance, using installation option SDBMSG=YES enables DFSORT to choose optimal block sizes for its message data sets on DASD.

## Tape

Tape is the least expensive media on a per-byte basis. It has the highest capacity for data storage of any component in the storage hierarchy. But it also has a relatively slow access time and must be accessed sequentially. However, automatic tape libraries and IDRC compacted tape make tape a good choice for SORTIN and SORTOUT.

Tape devices are not recommended for use with work data sets since a fast access rate is critical and work data tends to be accessed in a nonsequential manner.

### Environment Considerations

Performance is significantly better when DASD devices are used for work data sets. Note that you cannot use the efficient Blockset technique with tape work storage devices.

Certain data set characteristics, such as block size, are important when considering performance on tape devices. Using DFSORT's installation option SDB=YES enables DFSORT to choose optimal block sizes for its output data sets on tape. Of less importance, using installation option SDBMSG=YES enables DFSORT to choose optimal block sizes for its message data sets on tape as well.

## Virtual Storage

Logically, DFSORT (like any other application) works within a virtual address space. Installation defaults such as TMAXLIM and run-time options such as REGION and MAINSIZE determine the size of this address space. With the exception of data space sorting (see "Sorting with Data Space" on page 10 for a discussion of data space sorting) this size remains constant throughout most of the sorting process.

DFSORT attempts to make the best use of the virtual storage it has available. If you provide DFSORT with enough virtual storage, it might be able to sort the input data set entirely in virtual storage (an "in-main-storage" sort) without need of work data sets or Hiperspace. This is the preferred method of sorting small data sets up to a few megabytes in size.

When an in-main-storage sort is not possible or practical, DFSORT processes a portion of the input data set at a time, then combines all of these processed portions together into the final sorted data set. It is here that DFSORT excels at allocating virtual storage effectively in order to minimize both the number of portions as well as the time spent combining the portions into the output data set.

With data space sorting, DFSORT creates a data space to help carry out its processing. This is a new area of virtual storage, and is in addition to the original (specified or defaulted) virtual storage requested. The size of the data space is sufficient to guarantee an efficient sort (or data space sorting is not used).

In addition, DFSORT adjusts the size of the data space during processing, as necessary, in response to system paging levels. When system paging levels rise, DFSORT reduces its use of virtual storage (as long as this reduction does not significantly degrade DFSORT performance).

DFSORT also tries to move as many of its data areas above 16MB virtual as it can to help provide virtual storage constraint relief for the system.

## Main Storage

Main storage is the portion of virtual storage in the primary address space that DFSORT limits itself to using. In general, the more main storage you make available to DFSORT, the better the performance for larger jobs. To prevent excessive paging, insure that sufficient real storage is available to back up the amount of main storage used. This is especially important with main storage sizes greater than 32MB. The default amount of main storage that will be made available to DFSORT is defined when it is installed.

DFSORT requires a minimum of 88KB, but to get better performance, use a much larger amount of storage. The recommended amount is about 4MB. Improved

performance is most noticeable with large input data sets. Guidelines for setting these values are given in Table 2 on page 49 and Table 3 on page 50.

Although DFSORT can run some applications in the minimum of 88KB (below 16MB virtual), the minimum amount of main storage required for most applications is more than 88KB.

## System-Managed Storage

The Storage Management Subsystem (SMS) makes data set allocation very easy and efficient. Having SMS manage the temporary data sets can be a good first step in migrating to system-managed storage. However, the SMS automatic class selection (ACS) routines can unintentionally affect DFSORT data set allocations and job performance, so you might need to coordinate changes to the ACS routines with the site's storage administrator, as described here.

When any data set is allocated on an SMS system, the allocation request passes through the system's data class and storage class ACS routines. If the data set will be system-managed, the request also passes through the system's management class and storage group ACS routines. There is only one set of ACS routines per site, and they are very powerful. They can override DFSORT installation options, such as VIO=NO, and can even override requests for a certain data, storage, or management class, or a certain unit or volume.

As an example of how ACS routines can affect the performance of DFSORT applications, consider a storage class ACS routine that assigns all temporary data sets to the STANDARD storage class, as shown in Figure 2. The storage class is then used to assign the temporary data sets to a storage group. Because the routine does not differentiate between DFSORT temporary data sets and other temporary data sets, the storage group ACS routine cannot selectively prevent DFSORT temporary data sets from being assigned to VIO. Allocating temporary data sets to VIO works well in most cases, but might not be desirable for DFSORT temporary data sets, as explained in "VIO for DFSORT Data Sets" on page 53.

```
PROC 1 STORCLAS
     ......
     ......
     SELECT
        WHEN(&DSTYPE = 'TEMP')
          SET &STORCLAS = 'STANDARD'
     END
     ......
     ......
END /* END OF PROC */
```

*Figure 2. ACS Storage Class Routine. &DSTYPE is used to assign temporary data sets to the STANDARD storage class.*

One way to avoid allocating DFSORT temporary data sets to VIO is to write the ACS storage class routine so it assigns all DFSORT temporary output and work data sets (for example, those with ddnames SORTOUT, SORTOFdd, SORTWKdd, and SORTDKdd) to a special NONVIO storage class. Using the &DD variable is the most efficient way to determine whether or not a data set is a DFSORT data set. Because you cannot use the &DD variable to check the ddname in the storage

group ACS routine, you must check for DFSORT temporary data sets in the storage class ACS routine as shown in Figure 3.

```
PROC 1 STORCLAS

/* DEFINE DFSORT TEMPORARY DATA SETS  */

FILTLIST DFSORTDD INCLUDE(SORTWK*,SORTDK*,SORTOF*,'SORTOUT')
     ......
     ......
/* ASSIGN 'NONVIO' STORAGE CLASS TO DFSORT TEMPORARY DATA SETS */
/* AND 'STANDARD' TO ALL OTHER TEMPORARY DATA SETS               */

     SELECT
        WHEN(&DSTYPE = 'TEMP')
          IF (&DD = &DFSORTDD)
           THEN SET &STORCLAS = 'NONVIO'
           ELSE SET &STORCLAS = 'STANDARD'
        OTHERWISE SET &STORCLAS = '
     END
     ......
     ......
END /* END OF PROC */
```

*Figure 3. Storage Class ACS Routine. &DSTYPE and &DD are used to assign storage class NONVIO for DFSORT temporary data sets. The ddnames of these temporary data sets should be reserved names.*

The storage group ACS routine can then look for the SORT storage class, and assign the data sets to a non-VIO storage group, as shown in Figure 4. The site's storage administrator will need to create the special NONVIO storage class and alter the storage class and storage group ACS routines.

```
PROC 1 STORGRP
     ......
     ......
/* ASSIGN ALL TEMPORARY DATA SETS THAT ARE NOT DFSORT    */
/* DATA SETS TO 'SGVIO'                                  */

     SELECT
        WHEN(&DSTYPE = 'TEMP' && &STORCLAS; ¬= 'SORT')
          SET &STORGRP = 'SGVIO','PRIME'

/* ASSIGN DFSORT TEMPORARY DATA SETS TO 'PRIME'          */

        OTHERWISE SET &STORGRP = 'PRIME'
     END
     ......
     ......
END /* END OF PROC */
```

*Figure 4. Storage Group ACS Routine. &DSTYPE and &STORCLAS¬=SORT are used to prevent VIO allocation for DFSORT temporary data sets. PRIME is a pool storage group used for most data sets.*

Be aware that the ddnames SORTOUT, SORTWKdd, and SORTDKdd can be changed to other names when a program calls DFSORT. If other DFSORT output and work data set ddnames are in common use at your site, you should also include them in the ACS routines. This scheme does not work for OUTFIL data sets specified with the FNAMES operand unless common ddnames are specified.

**Note:** Filtering on SORTIN in the ACS routines cannot prevent DFSORT temporary input data sets from being allocated to VIO. These data sets are allocated in preceding steps and passed to DFSORT. Thus, SORTIN is not the ddname used when these data sets are allocated.

If the only DFSORT temporary data sets are dynamically allocated work data sets, another way to avoid VIO allocation is to use a non-VIO generic device type for the installation option DYNALOC and the run-time option DYNALLOC. The storage group ACS routine could then be written to assign a pool storage group to data sets with that generic device type. Again, the site's storage administrator will need to establish a non-VIO generic device type and alter the storage group ACS routine.

There might be other cases as well where the site's ACS routines unintentionally alter DFSORT performance or data set allocation. Be aware of this, and if you encounter problems, consult with your storage administrator to work out a joint solution.

# Chapter 4. Installation Considerations

Improving the performance of DFSORT consists of a number of activities, including:

- Tuning on a site-wide or system level
- Tuning of individual applications
- Designing efficient applications

To some extent, the success of each depends upon the success of the others. For instance, suppose we have a DFSORT application which has been carefully designed and tuned, and would be a good candidate for data space sorting. It might be adversely affected by the site's decision to set the default DSPSIZE parameter to 0. Such a setting would turn off dataspace sorting (unless the programmer has specifically overridden it in the application).

In a similar fashion, suppose that a significant site-wide tuning effort had been done to find a sufficiently large value for the installation parameter TMAXLIM. This parameter controls DFSORT's default maximum virtual storage. If a programmer overrides this default by specifying an unusually small SIZE or MAINSIZE value in an application, the application might make larger demands on the system's DASD and processor resources. This, in turn, could cause performance problems for that application as well as any other active applications on the system.

Even the best system and application tuning may be wasted on applications that use sorts unnecessarily. For example, an application that sorts two already sorted data sets into one could be replaced with a more efficient merge application. Likewise, an application that uses a sort to extract a subset of the records, but does not rearrange the records in any way, could be replaced with a more efficient copy application.

This chapter offers advice for system programmers and application developers who are responsible for installing and using DFSORT. It includes the following topics:

- Installing DFSORT
- Understanding storage options
- Using DFSORT capabilities
- Changing installation defaults
- Understanding installation performance options
- Using installation exits

## DFSORT Installation

The way DFSORT is installed can affect its performance. This section explains some of the things you should consider.

### Running DFSORT Resident

By running DFSORT resident, you can gain three performance benefits:

- Two or more applications can use the same copy of DFSORT in main storage at the same time. This enables central storage to be used more efficiently and cuts down on system paging.
- The DFSORT load modules do not have to be loaded each time DFSORT is run. This also saves unnecessary paging and time. This is especially noticeable for the smaller DFSORT applications, which tend to make up the bulk of DFSORT jobs at most sites.

### Installation Considerations

- The space for the DFSORT load modules is not charged against the virtual storage limits of individual applications. This saves storage that can be used by DFSORT to do a more efficient sort.

Since DFSORT is invoked so frequently, it is a prime candidate for running resident. When the DFSORT ICEGENER facility is used to replace IEBGENER, as described in "ICEGENER", DFSORT's use greatly increases, making it even more important to run DFSORT resident.

## DFSORT SVC

The DFSORT-supplied SVC enables DFSORT to run authorized functions without itself being authorized. In particular, the following performance-related functions are impaired if DFSORT's SVC is not available:

**SMF type-16**
DFSORT's type-16 SMF record contains useful information for analyzing the performance of DFSORT (see "Using SMF Data" on page 78). Without the SVC, DFSORT cannot write the SMF record to an SMF system data set, although the record can still be obtained through an ICETEXIT routine. If DFSORT's SMF feature is activated (installation or run-time option SMF=SHORT or SMF=FULL) and a properly installed SVC is not available, then all DFSORT applications will abend.

**Cache fast write**
Cache fast write (CFW) enables DFSORT to save elapsed time because DFSORT is able to write its intermediate data into storage control cache, and read it from the cache (see "Cache Fast Write (CFW)" on page 12). Without the SVC, DFSORT cannot use CFW, and issues message ICE191I. Processing continues with possibly degraded elapsed time performance.

**Caching mode**
For storage control units that support cache, DFSORT selects the caching mode that appears to be the best for the circumstances. In particular, this enables DFSORT to bypass the cache altogether when it appears it is not beneficial to use the cache. This helps other applications, including other DFSORT applications, make better use of the cache. Without the SVC, DFSORT cannot set these caching modes, and issues message ICE191I. This results in the default modes being selected, with possibly degraded system and DFSORT elapsed time performance.

In addition to the functions described above, there are other performance enhancements that are available to DFSORT through use of the SVC.

**Note:** It is strongly recommended that you make the DFSORT SVC available for best performance. Make sure that the installation of the SVC has been completed correctly so that the SVC can be used.

How the SVC is installed depends on whether you are replacing an earlier DFSORT release, installing a new release to coexist with an earlier release, or installing DFSORT for the first time. Make sure that the installation option of SVC is set to correspond to the way you install the SVC. See *Installation and Customization* for complete details on installing the DFSORT SVC.

## ICEGENER

At many sites, the copy utility IEBGENER is a frequently used program. Actions that improve its performance greatly benefit user productivity and resource utilization.

DFSORT's ICEGENER facility allows qualifying IEBGENER jobs to be routed to the more efficient DFSORT copy function. In most cases, using the DFSORT copy function instead of IEBGENER requires less CPU time, less elapsed time, and results in fewer EXCPs (see Figure 1 on page 13). If the DFSORT copy function cannot be used, DFSORT automatically transfers control to the IEBGENER utility. You can install ICEGENER so that your existing IEBGENER jobs do not require changes.

These are some of the circumstances that prevent the use of ICEGENER:

- A SYSIN DD statement other than SYSIN DD DUMMY is present.
- Detection of an error before DFSORT has started the copy operation
- Any condition listed in DFSORT message ICE160A as follows:

  **1** The SYSUT1 or SYSUT2 data set was BDAM.

  **2** FREE=CLOSE was specified.

  **3** An attempt to open a data set caused a system error.

  **4** The SYSUT1 or SYSUT2 data set resided on an unsupported device.

  **5** ASCII tapes had the following parameters:

  (LABEL=AL or OPTCD=Q) and RECFM=D and BUFOFF¬=L

  or

  (LABEL=AL or OPTCD=Q) and RECFM¬=D and BUFOFF¬=0

  **6** An attempt to read the DSCB for the SYSUT1 data set caused an error.

  **7** An attempt to read the DSCB for the SYSUT2 data set caused an error.

  **8** The SYSUT1 data set had keyed records.

  **9** User labels were present.

  **10** A MODS statement E*xx* operand had SYSIN in the third parameter or T or S in the fourth parameter (that is, dynamic link-editing was requested.)

Under such circumstances, DFSORT transfers control to IEBGENER. However, IEBGENER may not be able to process the copy application either.

See *Installation and Customization* for complete details on installing ICEGENER as an automatic replacement for IEBGENER.

## Storage Options

By using appropriate values for DFSORT installation storage options, you can ensure that the majority of applications have sufficient virtual storage. If the IBM-supplied installation default value for an installation storage option is inappropriate for the majority of DFSORT applications at your site, you should change it to a more appropriate value. For specific applications, the installation values can be overridden using run-time options. See *Installation and Customization* and *Application Programming Guide* for details of DFSORT storage options and the relationships between these options.

## Recommendations for Storage Options

You must provide sufficient virtual storage to DFSORT using the guidelines given in "Virtual Storage Guidelines" on page 49 and "Virtual Storage and Sorting with Data Space" on page 49.

The following installation storage option values are recommended:

**SIZE** The default, SIZE=MAX, is recommended. This enables DFSORT to use as much virtual storage as possible, both above and below 16MB virtual, subject to the limits set by MAXLIM and TMAXLIM.

When installation option SIZE=MAX, EXEC PARM option SIZE=MAX, or run-time option MAINSIZE=MAX is in effect, the TMAXLIM, RESALL, and RESINV options are used. These options are not used when SIZE=n or MAINSIZE=n is in effect.

You should also be aware of how the JCL REGION parameter can affect DFSORT virtual storage allocation. While subject to the constraints of your site's IEFUSI and IEALIMIT exits, the JCL REGION value limits the amount of below 16MB virtual storage.

DFSORT attempts to place as much of its storage as possible above 16MB virtual. DFSORT, however, still needs sufficient storage below 16MB virtual to run effectively. In general, a REGION value of at least 512KB is best. If DFSORT is called by a program, the REGION value should be large enough to allow sufficient storage for DFSORT and the program. If E15 or E35 user exits are used, a larger REGION value might improve performance because these functions use more storage below 16MB virtual.

In general, the minimum of:
- REGION plus OVERRGN
- SIZE or MAINSIZE
- MAXLIM

determines the maximum storage available below 16MB virtual. Thus, with REGION=100K, OVERRGN=65536 (64KB), SIZE=4194304 (4MB), and MAXLIM=1048576 (1MB), a total of 4MB is available to DFSORT, but only 164KB can (and probably will) be available below 16MB virtual. On the other hand, if REGION=2M and SIZE=819200 (800KB), then a total of 800KB is available to DFSORT and all of it can (but probably will not) be allocated below 16MB virtual.

If the available storage below 16MB virtual is insufficient, DFSORT issues message ICE039A, which indicates the minimum additional storage required below 16MB virtual. Although the application might run if you add the minimum storage indicated, it is recommended that you increase either the REGION value, or the SIZE or MAINSIZE value (or both) to provide sufficient virtual storage for the application to run efficiently.

The EXEC PARM option SIZE=n or OPTION statement option MAINSIZE=n can be used to allow more (or less) storage for specific applications for which the installation default is not appropriate. For example, you might want to specify MAINSIZE=8M to improve performance for a critical large application when your installation default is 4MB.

When SIZE=n or MAINSIZE=n is in effect, RESALL and RESINV are not used. Generally, this does not cause a problem, but if it does you should ensure that the user exit size values in your MODS statement, if any, are correct. Because the user exit size in the MODS statement is only an estimate, you can raise it if necessary to allow more reserved storage. Alternatively, you could raise the REGION value or go back to using SIZE=MAX or MAINSIZE=MAX to resolve the problem. See *Installation and Customization* for details of the relationships between SIZE, MAINSIZE, RESALL, RESINV, REGION, and other storage parameters.

**TMAXLIM**

Although different sites have different requirements, experience indicates that the TMAXLIM default of 4MB is a good general purpose value. Raising

TMAXLIM to as high as 16MB might provide additional performance benefits, particularly if a high percentage of your site's DFSORT usage is spent on large sorting applications.

Alternatively, you could specify SIZE=n or MAINSIZE=n at run-time to make more (or less) storage available for specific DFSORT applications.

**DSA** DSA allows DFSORT to change the value of TMAXLIM dynamically if doing so should improve the performance of a sort job. In general, the default of 32 (MB) is sufficient to handle most sorts, but if your site runs very large sorts (multiple GB of input data), you might consider raising this to 64 or 128.

**MAXLIM**

The default of 1MB is generally sufficient. On the other hand, raising MAXLIM above 1MB might provide additional performance benefits for applications that cannot use the maximum amount of storage above 16MB virtual, providing that the REGION value is also raised an equivalent amount. DFSORT performance might be improved by larger MAXLIM and REGION values when E15 or E35 user exits are used.

**MINLIM**

A MINLIM value of at least the default value of 440KB is recommended. The major reason for not raising MINLIM is if your site has applications that fail with higher MINLIM values (and for which there is no easy fix for the failures).

The MINLIM value is important only for jobs which specify a SIZE or MAINSIZE value that is less than the MINLIM value. By not using a value of MAX, these jobs become locked into a specific virtual storage limit. They must also reserve storage without using RESALL and RESINV, usually by making the SIZE or MAINSIZE value less than the REGION value or by coding user exit sizes on a MODS control statement. Such applications are prime candidates for your tuning efforts.

One way to improve the performance of these applications is to raise the MINLIM value (for example, to the MAXLIM value). This enables such applications to run with a larger amount of virtual storage. This strategy, however, might cause some of these applications to fail due to insufficient reserved storage. Using run-time options, you should change the failing applications to use SIZE=MAX or MAINSIZE=MAX and set appropriate values for user exit sizes (on the MODS control statement), RESALL, and RESINV.

**OVERRGN**

The default and recommended values for this option are 64KB for directly-invoked and 16KB for program-invoked applications.

**RESALL**

The default is 4KB and should normally not be modified. If a storage related failure occurs when sufficient virtual storage (REGION and SIZE) has been specified, try setting RESALL to a larger value to correct the problem.

**RESINV**

Normally, a RESINV value of 16KB is sufficient and is recommended.

**ARESALL**

ARESALL is seldom needed and can be kept at its default of 0.

**ARESINV**

ARESINV is seldom needed and can be kept at its default value of 0.

### Installation Considerations

See *Installation and Customization* for more information about these parameters.

# DFSORT Capabilities

DFSORT provides several capabilities that allow you to maximize performance on your system. These capabilities and recommendations for using them efficiently are described in the sections that follow. The capabilities include:

- Sorting with data space
- Hipersorting
- Cache fast write (CFW)

## Sorting with Data Space

Data space sorting is a DFSORT capability that uses data space. For a more detailed description of this capability, see “Sorting with Data Space” on page 10. The section that follows provides recommendations for using data space sorting to improve DFSORT performance.

### Recommendations for Sorting with Data Space

The recommended installation setting is DSPSIZE=MAX. This enables DFSORT to dynamically determine the amount of data space to be used for dataspace sorting, taking into account the size of the file being sorted and the paging activity of the system.

Use of dataspace sorting should provide significant CPU and elapsed time performance improvements for DFSORT applications. Because Hipersorting is never used in conjunction with dataspace sorting, some applications may experience an increase in EXCPs due to the use of work data sets rather than Hiperspace for intermediate data. When your tuning efforts emphasize EXCP counts over CPU and elapsed times, you may wish to turn off dataspace sorting by specifying DSPSIZE=0.

If a number of large applications that use dataspace sorting start at the same time on a system, resource contention for central storage could become a problem. If central storage is overcommitted, paging and swapping rates increase significantly. If this situation is likely to occur frequently at your site, it is recommended that you place further restrictions on the use of dataspace sorting. This can be accomplished by:

- Setting a DSPSIZE=n value as the installation default (rather than DSPSIZE=MAX). The appropriate value to use for n is best determined by experimentation; start with a low value and keep raising it by a few megabytes until system paging rates are not adversely affected. In cases of severe overcommitment of central storage, an installation default of DSPSIZE=0 may be advisable.
- If resource contention is a problem only during specific time intervals at your site, it is recommended that you use DSPSIZE=MAX as the value for your environment installation modules, but use a time-of-day installation module to restrict the DSPSIZE value for all jobs starting during these particular intervals. In this manner, the majority of DFSORT applications at your site can still take advantage of the performance benefits of using dataspace sorting, while at the same time the system is protected from excessive central storage contention. See “DFSORT Installation Defaults” on page 33 for more information on environment and time-of-day installation modules.

In general, DFSORT takes into account the effect on the application's performance and the effect on the system's performance before using data space. If either effect is not desirable, DFSORT chooses not to use dataspace sorting.

# Hipersorting

Hipersorting is a DFSORT capability that uses Hiperspaces for sorting. For a more detailed description of this capability, see "Hipersorting" on page 8. This section recommends ways to use Hipersorting to improve DFSORT performance.

## Recommendations for Hipersorting

The recommended installation settings for Hipersorting are:

- EXPMAX=MAX
- EXPOLD=MAX
- EXPRES=0
- HIPRMAX=OPTIMAL

These settings are described in detail in the sections that follow.

**EXPMAX=MAX**

This setting allows maximum Hipersorting activity on a system, especially during periods when non-Hipersorting applications are making little use of expanded storage. Setting EXPMAX=n, where n is a number of megabytes, or EXPMAX = p%, where p% is a percentage of the configured expanded storage, should only be considered when it is important for a site to limit the total amount of expanded storage used by all Hipersorting applications on the system. In this case, EXPMAX is a far better alternative than trying to use storage isolation for the Hipersorting applications, since EXPMAX=n or p% still allows access to all of the expanded storage by non-Hipersorting applications.

**EXPOLD=MAX**

This setting is recommended only if mass migration of expanded storage data to auxiliary storage does not cause a problem on your system. During batch processing windows, migrating old data to auxiliary storage improves overall system performance, since it allows more active data, like DFSORT work data, to use expanded storage in place of DASD. But, the one-time migration can have a negative impact on system performance, especially online response time, during other periods.

If the migration of large amounts of old expanded storage data is a concern for your site, set EXPOLD=n or p%. n, a number of megabytes, or p%, a percentage of the configured expanded storage, is an amount of migration that your site can tolerate at a single time. Do not set EXPOLD=0 since old expanded storage data will never migrate due to DFSORT activity. Also, old data will monopolize expanded storage and force DFSORT to use DASD in place of expanded storage.

**EXPRES=0**

This setting allows DFSORT as much access to expanded storage as any other application. If you want non-Hipersorting applications to have preferred access to expanded storage, set EXPRES=n, where n is some small number of megabytes or EXPMAX = p%, where p% is a small percentage of the configured expanded storage. This is the preferred way to control total DFSORT Hipersorting activity. This setting permits DFSORT almost full use of available expanded storage when non-Hipersorting activity is light, but greatly reduces the possibility of an overuse of expanded storage when there is a sudden increase in the use of expanded storage by

non-Hipersorting applications. Generally, you use this setting when the non-Hipersorting applications have small, sudden needs for expanded storage or the Hipersorting activity is unusually large.

**HIPRMAX=OPTIMAL**

This setting allows the DFSORT installation options EXPMAX, EXPOLD, and EXPRES to control the total Hipersorting activity on a system. Before the availability of dynamic Hipersorting (that is, prior to Release 13), DFSORT only partially regulated total Hipersorting activity. Also, HIPRMAX was the only installation option to control Hipersorting activity.

As a result, many sites found it useful to set HIPRMAX=n. With dynamic Hipersorting, DFSORT has full control over total Hipersorting activity, and you can customize your definition of HIPRMAX=OPTIMAL with the installation options EXPMAX, EXPOLD, and EXPRES. HIPRMAX=n or p% should now be reserved for use as a run-time override for applications that have a special reason to limit the amount of Hiperspace available for Hipersorting.

Hipersorting can provide considerable elapsed time and EXCP count improvements for DFSORT sorting applications. However, the CPU time may increase due to increases in the Hiperspace processing time (HPT). As a result, if your tuning efforts emphasize CPU time over elapsed time and EXCP counts, you may wish to turn off Hipersorting by specifying HIPRMAX=0. However, most system programmers feel that the additional HPT overhead is more than compensated for by the significant reductions in elapsed time, channel usage, and EXCPs.

# Cache Fast Write

Cache fast write (CFW) refers to the capability of the cached 3990 storage control units to use cache memory to improve average input and output times. For a more detailed description of this capability, see “Cache Fast Write (CFW)” on page 12. This section provides recommendations for using cache fast write to improve DFSORT performance.

## Recommendations for Cache Fast Write

The recommended installation setting for cache fast write is CFW=YES. As mentioned in “Benefits” on page 12, however, use of cache fast write benefits DFSORT performance only if a significant percentage of work data can fit into the cache, and if the overall cache activity is not too heavy. If the data in the cache is continuously destaged to DASD, there is no performance advantage in using cache fast write. It is most advantageous to set CFW=YES for small and intermediate sized sorts where the work data sets are relatively small.

A setting of CFW=NO is advisable for sites where a large percentage of the DFSORT workload consists of larger sorts or for sites where only very small amounts of storage control cache is installed. This setting should also be used if the size of the data set being sorted greatly exceeds the total amount of cache, if there are more important applications other than sorts that need the cache, or if the total activity on the cache with sorts is unacceptable.

For sites where only a few sorting applications are large, it may be more appropriate to leave the global CFW=YES installation setting and disable cache fast write for these individual large applications by using the DEBUG NOCFW run-time parameter.

## DFSORT Installation Defaults

DFSORT is shipped with a set of installation defaults that are used by all DFSORT applications at a site. These defaults set values for various DFSORT parameters. They can be changed on a site-wide level by environment and time-of-day using ICEMAC. See *Installation and Customization* for complete details of installation defaults.

Most of the installation defaults can be overridden for specific applications by setting the appropriate run-time option(s). In addition, some of the defaults (as well as run-time options) can be overridden for all or a subset of applications at a site through use of an ICEIEXIT routine. See *Application Programming Guide* for complete details of DFSORT's option override scheme.

## ICEMAC

The ICEMAC macro can be used to modify the IBM-supplied installation default values for DFSORT. Many of these installation options have an impact on performance, including:

- Storage limits
- Hiperspace and data space limits
- Use of system-determined block size
- Reallocation of VIO work data sets
- Use of dynamic allocation for work data sets
- Number of work data sets for dynamic allocation
- Device type for dynamically allocated data sets
- Use of control interval access for VSAM data sets
- Number of I/O buffers to use
- IDRC tape compaction ratio
- Use of VERIFY, EQUALS, ALTSEQ, and LOCALE

Modifications to ICEMAC should be carefully chosen to reflect how you want DFSORT to run at your site. You should only specify ICEMAC options if the supplied default values are not acceptable. A USERMOD is required to update the source distribution library to reflect the changed options.

## Environment Installation Modules

ICEMAC allows separate sets of installation defaults for four environment installation modules, based on how DFSORT was invoked, as follows:

**JCL-invoked (ICEAM1)**
DFSORT invoked directly (that is, not through programs) by batch jobs

**Program-invoked (ICEAM2)**
DFSORT invoked through batch programs

**TSO-invoked (ICEAM3)**
DFSORT invoked directly by foreground TSO users

**TSO program-invoked (ICEAM4)**
DFSORT invoked through programs by foreground TSO users

## Time-of-Day Installation Modules

ICEMAC allows each of the environment installation modules to specify time-of-day installation modules (ICETD1–4) with separate sets of installation defaults to be used for runs on specific days and times (for example, from 8:00am to 5:00pm on

Saturday and Sunday). You might want to take advantage of this feature to allow, for example, DFSORT to use larger storage values (DSA, TMAXLIM, and so on) for runs at certain days and times.

## Listing the Installation Defaults with ICETOOL

You can use an ICETOOL job similar to the one in Figure 5 to list the installation defaults actually in use at your site for the eight installation modules and the IBM-supplied defaults they override, where appropriate.
See *Installation and Customization* for complete details of ICEMAC options and

```
//DFRUN   JOB A402,PROGRAMMER
//LISTDEF EXEC PGM=ICETOOL,REGION=1024K
//TOOLMSG DD SYSOUT=A
//DFSMSG  DD SYSOUT=A
//SHOWDEF DD SYSOUT=A
//TOOLIN  DD *
  DEFAULTS LIST(SHOWDEF)
/*
```

*Figure 5. Using ICETOOL to List Installation Defaults*

procedures.

**Note:** The DFSORT installation defaults are based on customer feedback. They should only be changed on an exception basis.

Installation defaults that are inappropriate for an application should be overridden for that application with the corresponding run-time options. See *Application Programming Guide* for complete details of run-time options.

## Installation Options and Performance

Table 1 on page 35 shows site-wide ICEMAC installation options that influence the performance of DFSORT, a description of each option, and additional comments about the option.

*Table 1. ICEMAC Options That Influence DFSORT Performance*

| Type | ICEMAC Option | Description | Comments |
|---|---|---|---|
| Options that tailor main storage | SIZE | Upper limit for total storage above and below 16MB virtual | Limited by TMAXLIM when SIZE=MAX is in effect. The recommended value is MAX. |
| | OVERRGN | Upper limit for storage over and above that specified by REGION. | Limited by the IEFUSI or IEALIMIT installation-wide exits using the “region limit” values. |
| | MAXLIM | Upper limit for storage below 16MB virtual. | Always used. The recommended value is 1MB. |
| | TMAXLIM | Upper limit for total storage above and below 16MB virtual. | Only used when SIZE=MAX or MAINSIZE=MAX is in effect. The recommended value is 4MB. |
| | DSA | Upper limit for dynamic storage adjustment. | Only used when SIZE=MAX or MAINSIZE=MAX is in effect and the use of additional storage should improve performance. The recommended value is 32MB. |
| | MINLIM | Lower limit for SIZE or MAINSIZE. | Only used when SIZE=n or MAINSIZE=n is less than MINLIM. The recommended value is 440KB. |
| | RESALL | Storage below 16MB virtual that is reserved for system use. | Only used when SIZE=MAX or MAINSIZE=MAX is in effect. Can reduce the amount of virtual storage available for use by DFSORT. |
| | RESINV | Storage below 16MB virtual that is reserved for use by an invoking program. | Only used when SIZE=MAX or MAINSIZE=MAX is in effect and DFSORT is program-invoked. |
| | ARESALL | Storage above 16MB virtual that is reserved for system use. | Can reduce the amount of virtual storage available for use by DFSORT. |
| | ARESINV | Storage above 16MB virtual that is reserved for use by an invoking program. | Only used when DFSORT is program-invoked. |

*Table 1. ICEMAC Options That Influence DFSORT Performance (continued)*

| Type | ICEMAC Option | Description | Comments |
|---|---|---|---|
| Options that affect use of Hipersorting | HIPRMAX | Upper limit for Hiperspace for a single application. | Hiperspaces limited by IEFUSI installation exit. Recommended setting is OPTIMAL. Use EXPMAC, EXPOLD, and EXPRES to control Hipersorting on a site-wide basis. |
| | EXPMAX | Upper limit for total available expanded storage used for all Hipersorting activity. | Available expanded storage is subject to non-Hipersorting activity. Should be set to MAX unless you want to limit Hipersorting activity to a fixed portion of expanded storage. |
| | EXPOLD | Upper limit for total old expanded storage used for all Hipersorting activity. | Old expanded storage is subject to non-Hipersorting activity. Should be set to MAX unless a large migration of old expanded storage data causes a problem at your site. |
| | EXPRES | Lower limit for available expanded storage reserved for non-Hipersorting use. | Available expanded storage is subject to non-Hipersorting activity. Should be set to 0 unless you want DFSORT to have lower access priority to expanded storage than other applications. |
| Options that affect use of data space | DSPSIZE | Upper limit for data space size. | Recommended setting is MAX. If central storage is overcommitted, set value to a low value and test it. If resource contention is a problem only at specific times, set the value to MAX in your environment installation modules and use a time-of-day installation module to restrict the DSPSIZE value for all jobs during these specific times. Data spaces are limited by IEFUSI. |

*Table 1. ICEMAC Options That Influence DFSORT Performance (continued)*

| Type | ICEMAC Option | Description | Comments |
|---|---|---|---|
| Options that influence allocation of work data sets | DYNAUTO | Whether work data sets should be dynamically allocated automatically. | Can cause an increase in CPU time for small sort applications but helps minimize the amount of DASD space required for sorting. |
| | DYNALOC | Default device type and number of work data sets when dynamic allocation is requested. | Does not request dynamic allocation; only supplies defaults. |
| | DYNSPC | Default amount of space to allocate for dynamically allocated work data sets. | Only used when DFSORT cannot estimate the input data set size. |
| | IDRCPCT | Compaction ratio to use for IDRC-compacted tape data sets when computing data set size. | Tells DFSORT how much work space to dynamically allocate when input is on an IDRC-compacted tape. |
| | VIO | Whether VIO work data sets should be automatically reallocated to real DASD locations. | ACS routines can override this option. |
| Options that affect VSAM performance | CINV | Whether control interval access is used for VSAM input data sets. | Improves performance for VSAM input data sets. |
| | VSAMBSP | Amount of VSAM buffer space to use. | To improve VSAM performance, set VSAMBSP=OPTIMAL or VSAMBSP=MAX. |
| Options that affect input and output data set performance | IOMAXBF | The maximum buffer space to be used for DASD SORTIN and SORTOUT data sets. | Default value of 32MB is recommended. Should only be lowered when device contention or long channel connect times are a problem. Lowering the default could result in larger EXCP counts for these data sets, but could also decrease the channel connect time per EXCP. |
| | ODMAXBF | The maximum buffer space to be used for each OUTFIL data set. | Default value of 2MB is recommended. Lowering the value can cause performance degradation for the application. When you use more than 2MB, the performance improvements are small except for EXCPs, and, there is an increased need for storage. |

*Table 1. ICEMAC Options That Influence DFSORT Performance (continued)*

| Type | ICEMAC Option | Description | Comments |
|---|---|---|---|
| Other options that affect performance | CFW | Whether cache fast write is used for DFSORT work data sets. | Only applicable for work data sets located on DASDs attached to cached 3990 storage control units. Can be used only if DFSORT SVC is installed. Cache fast write can reduce the elapsed time of sorting applications. CFW is more beneficial to DFSORT performance for small and intermediate sized sorts, where the work data sets are relatively small. |
| | EQUALS | Whether input order is preserved for records with equal keys. | Can cause an increase in CPU time. The default setting of EQUALS=VLBLKSET should be changed to EQUALS=NO if possible. |
| | VERIFY | Whether output records are checked for correct order. | Can cause an increase in CPU time. |
| | ALTSEQ | Whether a collating sequence other than EBCDIC is used. | Can cause an increase in CPU time. |
| | SDB and SDBMSG | Whether system-determined block size is used for output data sets whose block size is zero. | Use of optimum block sizes for output data sets provides more efficient performance than using other block sizes. |
| | IGNCKPT | Specifies whether Checkpoint/Restart requests at run-time should be ignored. | When CKPT is specified, the Blockset technique cannot be selected. Therefore, the recommended setting is IGNCKPT=YES so that the Blockset technique can be used. |
| | CHALT | Specifies whether ALTSEQ is to be applied to character format fields (CH). | As with ALTSEQ, can cause an increase in CPU time. The default setting of CHALT=NO should be used if possible. |
| | COBEXIT | Specifies the library for COBOL E15 and E35 routines. | There are performance advantages to using the newer versions of COBOL rather than OS/VS COBOL. The value specified depends on which version of COBOL you are using. |
| | EFS | Specifies the name of a user-written Extended Function Support program to be called by DFSORT. | Can cause an increase in CPU time and elapsed time. Use only when necessary for your application. |
| | LOCALE | Specifies whether locale processing is to be used and, if so, designates the active locale. | Should only be used when required since it can show degraded performance relative to collation using character encoding values. |

See *Installation and Customization* for a complete list of site-wide installation options and the *Application Programming Guide* for corresponding run-time overrides.

## Installation Exits

DFSORT allows you to use user-written, installation-wide initialization and termination exit routines to perform a variety of functions, such as overriding the options currently in effect and collecting statistical data. For tuning purposes it is often advantageous to install these exits for the following reasons:

- These exits can be used for performance data gathering to help you understand the use of DFSORT at your site and make the appropriate tuning decisions based on this information.
- An initialization routine allows you to override the run-time values set for certain options, which enforces your decisions for those option values for all DFSORT applications at your site.

## ICEIEXIT

A site-supplied ICEIEXIT routine can exercise control over certain DFSORT run-time options.

If present and activated, the ICEIEXIT routine is called and passed installation and run-time information by DFSORT. The ICEIEXIT routine can then use current DFSORT and system information to determine whether to change certain options in effect. This also permits site-wide control of certain options whose installation defaults have been overridden at the application-level.

An ICEIEXIT routine can examine installation and run-time information related to:
- Storage limits
- Hiperspace limits
- Data space limits
- Use of VERIFY
- OUTFIL buffer space limits

and additional run-time information related to:
- DFSORT technique used
- Type of DFSORT application
- Method of DFSORT invocation
- Storage above 16MB virtual
- Configured expanded storage

An ICEIEXIT routine can also change certain run-time options including:
- Storage limits
- Hiperspace limits
- Data space limits
- Use of VERIFY
- OUTFIL buffer space limits

A site could use an ICEIEXIT routine to control applications and enforce site standards. For example:

- Many options (for example, MAXLIM, SIZE, TMAXLIM, DSA) can affect the virtual storage used by DFSORT. An ICEIEXIT routine could specify the amount of virtual storage to be used depending on such factors as performance requirements and jobname.

**Note:** The time-of-day installation modules allow you to specify the virtual storage to be used depending on the day and time when an application runs. See "Time-of-Day Installation Modules" on page 33 for more information.

- Before creating a data space, DFSORT checks to see how much central storage either is not being used or has gone unreferenced for a sufficient period of time. This is to make sure enough real storage is available to back the data space without causing excessive system paging activity. An ICEIEXIT routine can further reduce the risk of overcommitting central storage by limiting the amount of data space that a single DFSORT application can use. This would also override any run-time specifications that try to get around the installation default.

See *Installation and Customization* for information about coding an ICEIEXIT routine and a sample ICEIEXIT routine, which shows how the storage available to DFSORT can be dynamically modified based on the jobname/stepname and type of application.

## ICETEXIT

If present and activated, the ICETEXIT routine is called at the end of DFSORT application processing. It is available for those who wish to make a thorough analysis of DFSORT performance data using a single source of information. See "Using ICETEXIT Data" on page 80 for more information about using this routine; see *Installation and Customization* for complete information on how to write and install an ICETEXIT routine.

# Chapter 5. Run-Time Considerations

This chapter offers advice about improving the performance of individual DFSORT applications by using run-time options related to the following areas:

- Sorting with data space
- Hipersorting
- Cache fast write
- File size
- Storage
- Input and output data sets

The last section includes a table with information on run-time options available with DFSORT that can affect performance.

## Sorting with Data Space

Dataspace sorting is a DFSORT capability that uses data space. See “Sorting with Data Space” on page 10 for a detailed description of dataspace sorting.

## The DSPSIZE Parameter

The DSPSIZE parameter specifies the maximum amount of data space to be used with dataspace sorting. The maximum size of the data space allocated by DFSORT for a job is determined by the minimum of the following values:

- The limit placed on the size of the data space by the system's IEFUSI installation exit. For a description of IEFUSI, refer to *Installation Exits*.
- The DSPSIZE value (either the installation default value, or an overriding value specified at run-time).
- The amount of system paging activity at the start of the run. The size of the data space can be adjusted throughout the run. DFSORT determines the optimal data space size based on system activity, to help avoid a negative impact on system paging when using dataspace sorting. If the system paging levels are high, DFSORT's data space size limit will be low.

## How DFSORT Uses Data Space

The recommended installation setting is the IBM-supplied default of DSPSIZE=MAX. This enables DFSORT to dynamically determine the amount of data space to be used for dataspace sorting, taking into account the size of the file being sorted and the paging activity of the system.

The amount of data space storage used for each sort job is displayed in message ICE188I. DFSORT dynamically determines how best to use data spaces for each particular run:

- If the input data set size is known to be small enough so that the sort can be accomplished in main storage, no data space is created.
- If the size of the input data set in relation to the maximum available data space amount is too large, no data space is created. Dataspace sorting is only used if the size of the data space that could be created is large enough to improve the performance of the sort application.
- If the size of the input data set in relation to the total available main storage is too large, no data space is created. To ensure that you have enough main

storage to use dataspace sorting, follow the recommended virtual storage guidelines for DFSORT (Table 3 on page 50).

In general, DFSORT takes into account the effect on the application's performance and the effect on the system's performance before using data space. If either effect is not desirable, DFSORT chooses not to use dataspace sorting.

See "Sorting with Data Space" on page 10 for information on the benefits and operation of dataspace sorting and "Recommendations for Sorting with Data Space" on page 30 for additional information on using dataspace sorting effectively.

## Hipersorting

Hipersorting is a DFSORT capability that uses Hiperspaces for sorting. For a more detailed description of this capability, see "Hipersorting" on page 8. The sections that follow include information on how to use Hipersorting in the most efficient way at your site.

DFSORT selects the most appropriate mode (Hiperspace-only, Hiperspace-mixed, or DASD-only) for each particular run. Not every sort application can use Hipersorting, and even for those sorts that can use Hipersorting, it may be more advantageous not to use it under certain circumstances. This section examines the most common reasons for not using Hiperspace and explains the possible actions that can be undertaken to allow more jobs to take advantage of Hipersorting.

Some customers have expressed concerns that they would like to see Hipersorting used more often. However, the use of expanded storage must always be weighed against the possibility of degrading performance for a job or for the entire system, by overusing expanded storage. If DFSORT were to use Hipersorting indiscriminately, there could be a significant increase in paging activity and a resulting reduction in total system performance, affecting all jobs, including sorts.

The recommended setting for HIPRMAX is OPTIMAL. This allows the DFSORT installation options EXPMAX, EXPOLD, and EXPRES to control the total Hipersorting activity on a system. Before the availability of dynamic Hipersorting (that is, prior to Release 13), DFSORT only partially regulated total Hipersorting activity. Also, HIPRMAX was the only installation option to control Hipersorting activity. As such, many sites found it useful to set HIPRMAX=n. With dynamic Hipersorting, DFSORT has full control over total Hipersorting activity, and sites can customize their definition of HIPRMAX=OPTIMAL with the installation options EXPMAX, EXPOLD, and EXPRES. HIPRMAX=n and HIPRMAX=p% should now be reserved for use as a run-time override for applications that have a special reason to limit the amount of Hiperspace available for Hipersorting.

The number of kilobytes of Hiperspace storage used during the sort is displayed in message ICE180I. If Hipersorting is not used, the message shows a value of 0K.

## Limitations

The maximum amount of Hiperspace used by DFSORT is limited to the minimum of the following values:

- The IEFUSI exit limit on the total amount of Hiperspace and data space that can be allocated in a single job step. See *MVS/ESA Installation Exits* for a description of IEFUSI.
- The HIPRMAX value in effect, when set to a value other than OPTIMAL. The HIPRMAX value in effect is either the installation default, an overriding value

specified at run-time, or an overriding value specified in the installation ICEIEXIT routine. Note that the value specified in the ICEIEXIT routine overrides any other value.

- Available expanded storage. Throughout the run, DFSORT determines the amount of available expanded storage, subtracts from this the amount of expanded storage needed by other concurrent Hipersorting applications, and factors in the values specified for installation options EXPMAX, EXPOLD, and EXPRES. If as a result of any such check, either an expanded storage shortage is predicted or one of the site limits for total expanded storage usage by all Hipersorting applications is reached, DFSORT switches from using Hiperspace to using DASD work data sets for all currently running Hipersorting applications.

  In addition, all future DFSORT applications are prevented from using Hipersorting until the expanded storage situation is relieved. This prevents Hipersorting applications by themselves from causing a shortage of expanded storage.

  Since this last criteria depends very heavily on system activity, especially other concurrent Hipersorting activity, DFSORT applications can use varying amounts of Hiperspace when run at different times and under different conditions. In fact, it is possible for such applications to not use any Hiperspace.

The following are those cases for which you should not attempt to adjust your application; in these cases the best performance for the individual job and for the system is achieved by not using Hipersorting:

- **Other performance features are in use.** Hipersorting is not used when DFSORT decides to use dataspace sorting. DFSORT dynamically chooses between using dataspace sorting and using Hipersorting and selects the one that provides the best performance for the particular sort. Message ICE188I indicates whether dataspace sorting was used for a particular run.
- **The size of the input data set is very small.** If the amount of data to be sorted is known to be small enough that the sort can be accomplished in main memory, Hipersorting is not used. Since no intermediate data is generated, neither Hiperspace nor DASD work data sets are needed. The presence of message ICE080I indicates that a sort was processed in main memory.

## Application Adjustments

The following are those cases for which you may want to adjust your application in order to take advantage of Hipersorting.

- **The Blockset technique was not selected.** Hipersorting is supported only for the Blockset technique. If Blockset is not selected, message ICE800I indicates why it was not selected.

  Note that the ICE800I message is printed only when a SORTDIAG DD statement was coded in the sort's JCL, or installation option DIAGSIM=YES has been specified for your site. Use the ICE800I reason code to determine the exact condition that is preventing the use of Blockset. If you are interested in using Hipersorting for the job, change your application appropriately to eliminate the particular condition, so that Blockset can be used.
- **Insufficient available virtual storage.** In some cases, the amount of virtual storage available to DFSORT can influence the potential effectiveness of a Hiperspace. A Hiperspace of a certain size could be too small to improve performance when an insufficient amount of virtual storage is available, whereas the same size Hiperspace might be large enough to improve performance when a sufficient amount of storage is available. Since DFSORT does not use Hiperspace when doing so would not result in a performance benefit, insufficient virtual storage can indirectly prevent the use of Hipersorting.

Supply DFSORT with sufficient virtual storage if you would like Hipersorting to be used. The third value in message ICE092I or ICE093I indicates the amount of storage available for a particular sort job. To help reduce the likelihood of not using Hipersorting because of insufficient virtual storage, ensure that this value is at least the maximum recommendation given in Table 2 on page 49. If necessary, increase the amount of virtual storage available to the job by specifying a larger MAINSIZE value on the OPTION control statement and/or raising the REGION value on the sort step EXEC statement.

- **Insufficient available expanded storage.** The size of the input data set in relation to the total available expanded storage has an important effect on the performance of Hipersorting. If the size of the Hiperspace that could be created is too small to hold a significant percentage of the intermediate data, then the performance of the run would be degraded compared to using DASD-only mode. Therefore, DFSORT chooses not to use Hipersorting in this situation.

  If you would like Hipersorting to be used, there are several possible approaches you can take:

  – Make sure that the HIPRMAX value or the installation IEFUSI exit is not limiting the application to a small amount of Hiperspace. Setting HIPRMAX=OPTIMAL (or to a very large value) and having IEFUSI allow at least 2 GB of Hiperspace per application will remove this limitation.

  – Make sure that the EXPMAX, EXPOLD, and EXPRES values allow significant amounts of Hipersorting. This is accomplished by setting EXPMAX and EXPOLD to large values (or MAX) and EXPRES to a small value.

  – Rerun the application when system activity, especially other concurrent Hipersorting activity, is lower so that more expanded storage is available for the sort. The more expanded storage available, the larger the Hiperspace that can be created by DFSORT, and the larger the data set size for which Hipersorting can be allowed.

    Remember that some data sets are so large that Hipersorting can **never** be used to sort them, even if all the expanded storage installed on a system were available for the sort. To allow Hipersorting in such cases, you can either break up the large sort into multiple smaller sorts, or install more expanded storage on the system.

  – Reduce the size of the input data set, so that less expanded storage is required for the sort. For some applications it is not necessary to sort all of the data, since only a subset is needed for processing. For example, use of INCLUDE/OMIT, SKIPREC, or STOPAFT can significantly reduce the amount of intermediate storage required by DFSORT. See *Application Programming Guide* for more details about these features of DFSORT.

  – Ensure that DFSORT knows the size of the input data set. In certain situations, DFSORT cannot accurately determine the amount of data to be sorted, and thus cannot accurately determine if the use of Hipersorting would be advantageous for a particular job. To ensure that DFSORT makes the correct decision about whether or not to use Hipersorting, specify a FILSZ parameter on the OPTION control statement. This is especially important for sorts using tape input data sets, E15 user exits, or INCLUDE/OMIT statements. See *Application Programming Guide* for more details on when DFSORT cannot accurately determine the input data set size.

  – The parameters that control the system resources manager (SRM) can indirectly affect the amount of expanded storage that is available for all the jobs on your system, including sort jobs. For example, PWSS=(0,100) may cause DFSORT to not use Hipersorting. See *Initialization and Tuning* for information about SRM and its parameters.

- **Inefficient work data set usage.** When a Hiperspace-mixed mode run is possible, DFSORT must decide how best to use both Hiperspace and DASD work data sets. In most cases, trade-offs can be made such that both types of intermediate storage can be used efficiently. In some cases, however, it is impossible to use both Hiperspace and DASD efficiently, in which case DFSORT chooses not to use Hipersorting.

  In order to avoid such cases, use only 3380 or later model DASD, and supply sufficient virtual storage to DFSORT, as described in Table 2 on page 49. Sometimes, it is necessary to rerun the jobs when there is less system activity (to allow selection of Hiperspace-only mode) in order to take advantage of Hipersorting.

These are some of the most common reasons why Hipersorting is not used under particular circumstances. In general, DFSORT takes into account the effect on the application's performance and the effect on the system's performance before using Hiperspace. If either effect is not desirable, DFSORT chooses not to use Hipersorting.

See "Hipersorting" on page 8 for information on the benefits and operation of Hipersorting and "Hipersorting" on page 31 for additional information on using Hipersorting effectively.

## Cache Fast Write

With DFSORT, cache fast write (CFW) refers to the capability of DFSORT to take advantage of the storage control unit's cache fast write function when writing to the work data sets. The recommended setting for cache fast write is CFW=YES. If you want to change the CFW setting for a specific application, you can use the CFW or NOCFW options of the DEBUG statement at run-time for that application.

Cache fast write benefits DFSORT performance only if a large percentage of work data can fit into the cache. Since data in cache is destaged to disk when additional cache slots are needed, there is no performance advantage in using cache fast write after the cache has been filled. Every write operation will be delayed by the speed of the DASD, since destaging is required.

It is most advantageous to use CFW for small and intermediate sized sorts, where the work data sets will be relatively small. Large sorts, for example, sorts that require more work space per work data set than is available in the cache, should not be allowed to use CFW, to avoid flooding the cache. If large sorts are allowed to use CFW, the performance of other applications that are trying to use the cache could be adversely affected.

See "Cache Fast Write (CFW)" on page 12 for information on the benefits and operation of cache fast write and "Cache Fast Write" on page 32 for additional information on using cache fast write effectively.

## File Size

File size is set using the FILSZ option which specifies the exact or estimated number of records to be sorted. DFSORT uses this record count to determine the input file size for a sort application. This value is important for sort applications, since it is used for several internal optimizations as well as for dynamic work data set allocation.

### Run-Time Considerations

The type of FILSZ value specified controls the way DFSORT performs the above function, and can have a significant effect on performance and work data set allocation. Available values include:

- x – specifies the exact number of records to be sorted. This value is always used for file size calculations. It has an additional feature that if x is not the exact number of records sorted, DFSORT terminates.
- Ux – specifies an estimated number of records to be sorted. This value is always used for file size calculations.
- Ex – specifies an estimated number of records to be sorted. DFSORT can choose to ignore this value in cases where a better estimate is available.

If you enter an incorrect value for the FILSZ parameter, DFSORT's performance can be degraded. An incorrect value can also cause wasted DASD space or termination when work space is dynamically allocated. Therefore, it is important to update the record count value whenever the number of records to be sorted changes significantly.

For a variable-length record sort application, if your average record length is significantly different from one-half of the maximum record length, use the AVGRLEN option to specify an accurate average record length. This will help you avoid wasted DASD space or termination when work space is dynamically allocated.

## Storage

DFSORT sorts most efficiently when sufficient virtual storage is available to enable an optimal balance between placing data in virtual and auxiliary storage. When virtual storage is limited, DFSORT must expend more resources to transfer data between virtual and auxiliary storage, which causes increased CPU time, elapsed time, and I/O usage.

Sufficient real storage must be available to support DFSORT's virtual storage requirements. Supplying DFSORT with more virtual storage might not improve performance if the available system resources cannot accommodate the corresponding increase in virtual storage activity. If real storage resources become overcommitted, excessive paging can result. This can cause the performance of both DFSORT and the system to degrade. It is important, therefore, to balance virtual storage resources supplied to DFSORT with the overall system resource requirements.

DFSORT's Dynamic Storage Adjustment (DSA) feature can let DFSORT tune the right amount of virtual storage for sort applications relieving system and application programmers of the task. See "Dynamic Storage Adjustment" on page 11 for more information on the benefits and operation of DSA.

See "Virtual Storage" on page 20 for a description of virtual storage.

## Data Set Size and Virtual Storage

The relationship between data set size and amount of virtual storage available is critical to the performance of DFSORT. Basically, there are four separate cases to consider.

- When virtual storage is larger than the data set, DFSORT may be able to perform the sort entirely within virtual storage, without need to store intermediate

data. This is called an *in-main-storage sort*. Indeed, this is the preferred method for sorting small data sets, since it minimizes I/O usage as well as CPU and elapsed time.

- When virtual storage is smaller than the data set, Hiperspace or work data sets are needed to store the intermediate data. Provided virtual storage is sufficient (see Table 2 on page 49 for guidelines), DFSORT is still able to perform an efficient sort, with elapsed and CPU times close to those of an in-main storage sort. I/O or Hiperspace usage is increased, however, reflecting the need to write intermediate data to work data sets or Hiperspace.
- When virtual storage is reduced further or the data set size is increased, DFSORT is forced to make less efficient use of Hiperspace or work data sets. DFSORT does what it can to maintain performance but is forced to use Hiperspace or work data sets less efficiently as the ratio of data set size to available storage increases. The loss of efficiency adversely affects elapsed time and EXCP counts.

  This performance degradation can be especially dramatic when using work data sets allocated on devices attached to non-synchronous storage control units or connected to ESCON channels. In such cases, it is especially important to follow the virtual storage guidelines explained in "Virtual Storage Guidelines" on page 49. Avoid using the 3390 Model 9 for work data sets since this device has slower random access performance than other 3390 devices; it is designed to store large amounts of input and output data which is accessed sequentially. See *Application Programming Guide* for more details about non-synchronous storage subsystem considerations.
- When virtual storage is very small or the data set size is very large, DFSORT may require several additional passes over the data to perform the sort. This phenomenon is known as *intermediate merging*, and results in significant performance degradation. Figure 6 on page 48 shows the benefit of increasing virtual storage to eliminate intermediate merging.

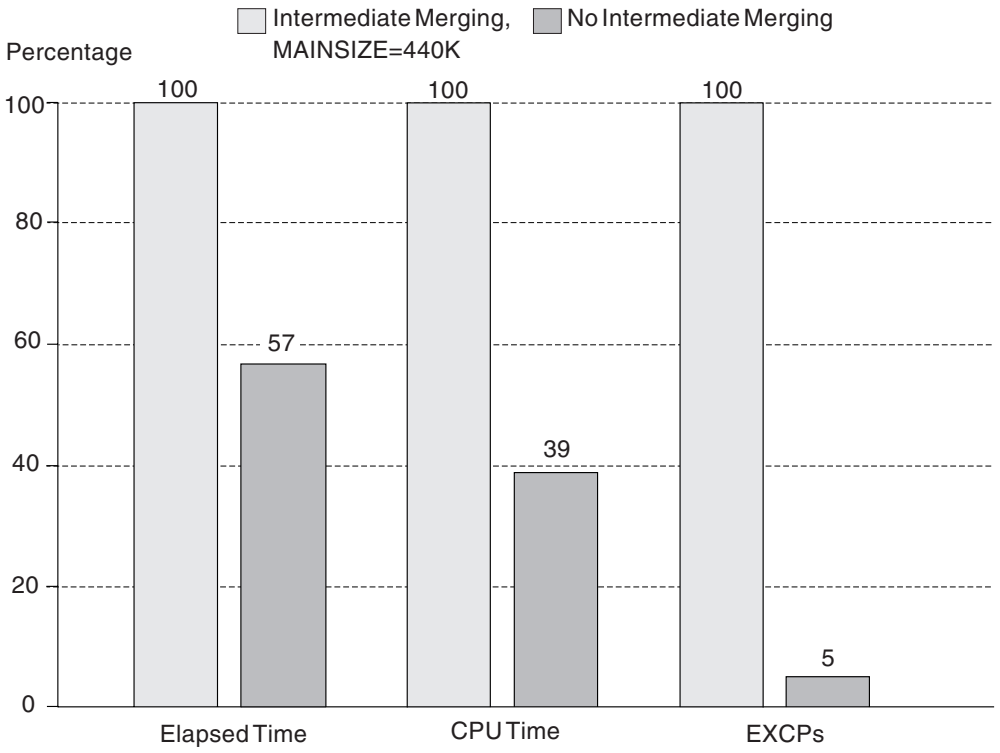



*Figure 6. Benefits of Eliminating Intermediate Merging*

All other factors being equal, the range of data set sizes that DFSORT can sort efficiently (or sort without requiring intermediate merging) grows roughly as the square of the virtual storage size. That is, doubling the virtual storage in an application enables the application to handle data sets four times as large with the same degree of efficiency. Likewise, halving the virtual storage causes the application to handle data sets only one-fourth as large with the same efficiency.

## Virtual Storage Limitations

With the possible exception of in-main storage sorts, providing more storage than needed to do an efficient sort (see Table 2 on page 49 for storage guidelines) will probably not result in any significant performance improvement. In fact, elapsed time (and possibly CPU time) may even increase a little! While this degradation might not be very noticeable, increasing virtual storage increases the overall effect DFSORT has on the system by tying up more central storage than necessary. This can result in fewer jobs being able to run at the same time as well as increased paging activity on the system.

If user exit routines are used, they will affect DFSORT virtual storage requirements. The exit routines will occupy virtual storage, and any storage requests they issue will reduce the amount of storage available to DFSORT. The MODS control statement should be used to reserve storage for exit routines.

If the storage available to DFSORT below 16MB virtual is severely limited (for example, to less than 256KB), the use of any of the following can result in storage failures or terminations:

- Spanned records
- COBOL exit routines
- CHALT, LOCALE, or SMF options

- ALTSEQ, INCLUDE, OMIT, SUM, OUTFIL, OUTREC, or INREC control statements
- Very large blocks or logical records
- VSAM data sets
- An Extended Function Support (EFS) program
- An ICETEXIT routine
- A large ICEIEXIT routine
- A large number of JCL or dynamically allocated work data sets

You can avoid storage problems and achieve better DFSORT performance by making sure MINLIM is always set to a reasonable value (for example, 440KB) and by using SIZE/MAINSIZE=MAX with DSA at 32 or more, or SIZE/MAINSIZE=nM with n set to at least that recommended in Table 2.

## Virtual Storage Guidelines

Table 2 gives guidelines for the recommended minimum virtual storage to use for a sort application based on its data set size. If you do not know the data set size, you can run the application and look at message ICE134I. The table gives a range of virtual storage sizes for each possible data set size. The low end of each range should produce an efficient sort for the given data set size. The high end, in some cases, will enable an even more efficient sort. Using less than the low end will likely produce noticeable degradation while using more than the high end will probably not have a significant impact on performance.

*Table 2. Recommended Minimum Storage Guidelines for Sorting Without Data Space*

| Input Data Set Size | Recommended Minimum Storage |
|---|---|
| Less than 50MB | 4MB |
| 50MB to 100MB | 4-6MB |
| 100MB to 200MB | 4-8MB |
| 200MB to 500MB | 6-12MB |
| 500MB to 1GB | 8-16MB |
| 1GB to 2GB | 12-24MB |
| More than 2GB | 16-32MB |

In order to guarantee the most efficient sorting, use the higher end of the range shown. In order to guarantee efficient, but not necessarily best, sorting, use the lower end. These values are intended for sorting without data space. See Table 3 on page 50 for storage recommendations for sorting with data space.

Although sort applications can usually run with less virtual storage than the recommended minimum, the recommended amount enables DFSORT to sort most efficiently. Using less than the recommended amount can result in the effects described in “Data Set Size and Virtual Storage” on page 46.

## Virtual Storage and Sorting with Data Space

Dataspace sorting has a different set of guidelines regarding virtual storage. For one thing, dataspace sorting creates and uses a data space to hold the records currently being processed. The size of this data space is chosen to be large enough to guarantee an efficient sort. Otherwise, dataspace sorting is not used.

### Run-Time Considerations

As a result of the ability of dataspace sorting to adjust its virtual storage requirements dynamically to the data set being sorted, and the fact that the virtual storage made available through the data space is in addition to the virtual storage available to DFSORT normally (through the SIZE or MAINSIZE parameter), the guidelines in Table 2 on page 49 are not applicable to dataspace sorting. Instead, use the values found in Table 3 for dataspace sorting applications.

*Table 3. Recommended Minimum Storage Guidelines for Sorting with Data Space*

| Input Data Set Size | Recommended Minimum Storage for Dataspace Sorting |
|---|---|
| Less than 200MB | 4MB |
| 200MB to 500MB | 4-6MB |
| 500MB to 1GB | 4-8MB |
| 1GB to 2GB | 4-10MB |
| More than 2GB | 4-12MB |

In order to guarantee the most efficient sorting, use the higher end of the range shown. In order to guarantee efficient, but not necessarily best, sorting, use the lower end. These values are intended for sorting with data space. See Table 2 on page 49 for storage recommendations for sorting without data space.

## Input and Output Data Sets

The performance of DFSORT can be affected by your choice of block sizes, the types of devices for input and output data sets, user exits, VIO, and some enhancements for input and output data sets. The sections that follow describe these items in more detail.

## Block Sizes

Choosing an efficient block size can improve space utilization and I/O performance.

### Space Utilization

The amount of space on a track or cylinder occupied by user data depends on the block size specified for the data set. Grouping logical records into blocks reduces the amount of space needed to store data. Because fewer physical records are needed to store the same number of logical records, the amount of space for count and key areas, and for gaps between records is reduced.

Larger block sizes offer better opportunities for increased DASD space utilization. An appropriately selected block size can result in higher space utilization than a smaller block size. An inappropriately selected block size (large or small) can result in poor space utilization. See *3390 Direct Access Storage Introduction* for detailed information on determining the space utilization for given block sizes on 3390 devices.

Figure 7 on page 51 shows the 3390 space utilization for various block sizes.

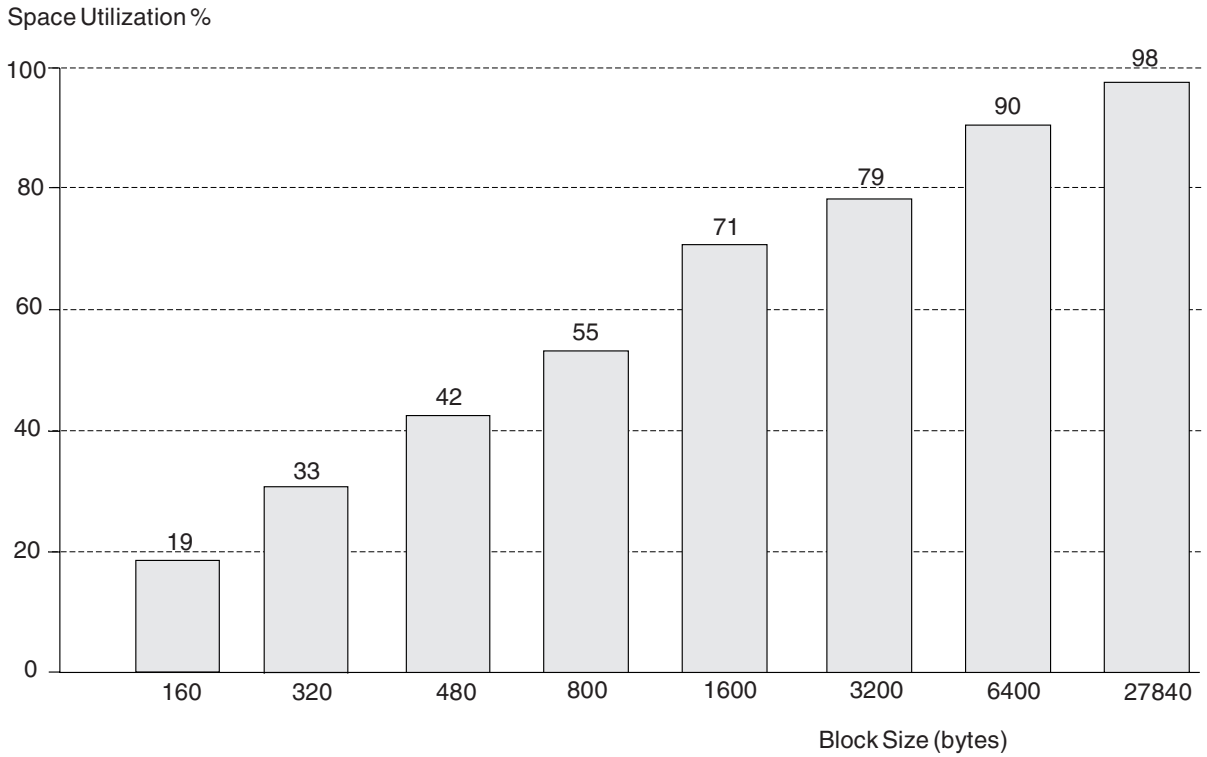



*Figure 7. 3390 Utilization for Various Block Sizes. Assumes data records are stored in equal-length physical records with no keys.*

## I/O Performance

Although small block sizes permit more concurrent channel operations, they reduce the net data transfer rate (the actual amount of data transferred per second). This can impact the elapsed time of a DFSORT application performing a significant amount of I/O. Small block size transfer also requires more CPU involvement and can, therefore, increase CPU time.

Large block sizes enable a higher net data transfer rate for sequential data sets, such as for input and output, and reduce the amount of processor time needed to service a channel program. An example of the benefits of appropriately large input and output data set block sizes is shown in Figure 8 on page 52.

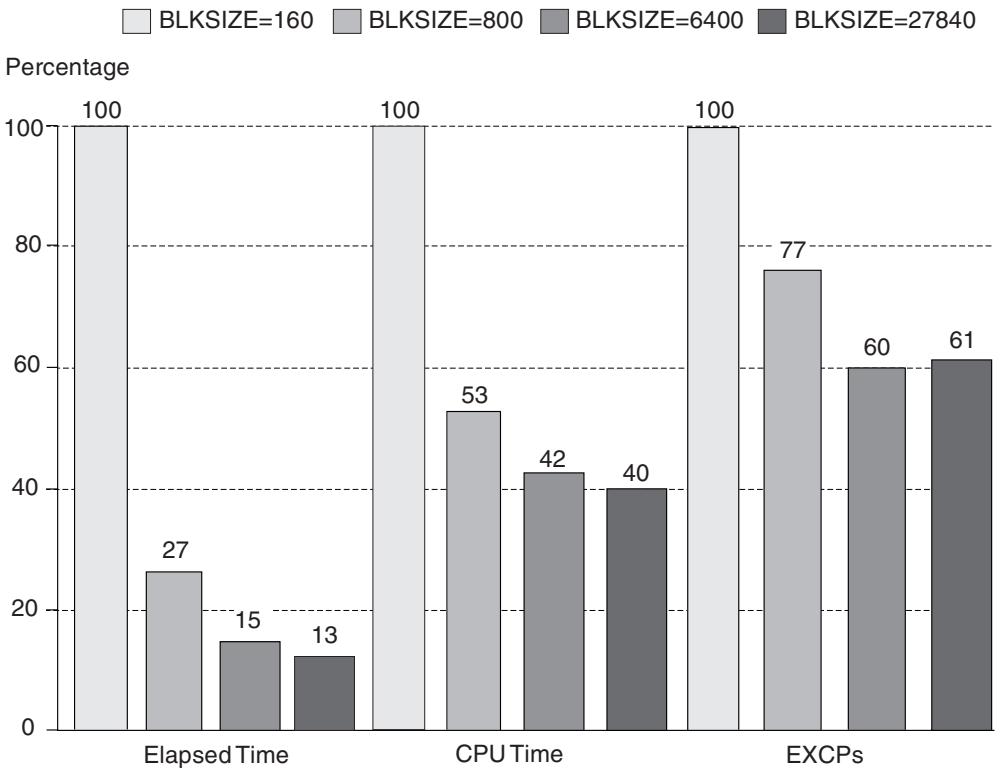



*Figure 8. Benefits of Large Input/Output Data Set Block Sizes*

### Recommendations

When selecting a block size for input or output, consider these factors:

- Smaller data set sizes generally result in less efficient use of DASD space.
- DFSORT applications that process data sets with small block sizes will generate higher EXCP counts and probably increase CPU time.

Select as large a block size as possible that is appropriate for the track capacity of the device you are using, adjusted to accommodate the size of the logical records in the data set. A block size of approximately 6000 bytes is a reasonably good choice, offering at least 85 percent utilization with 3390 devices. Even higher utilization can be obtained by selecting optimum block sizes based on the attributes of your data sets. See *3390 Direct Access Storage Introduction* for detailed information.

If you use DFSORT's system-determined block size feature (installation options SDB=YES and SDBMSG=YES) and do not supply an output block size, DFSORT chooses an optimal block size for the output data set based on its device type.

## Type of Device

For optimal performance, use 3390 devices for input, output, and work data sets to gain the advantages of higher data transfer rates, larger track capacity, and multiple path access. Avoid using the 3390 Model 9 for work data sets since this device has slower random access performance than other 3390 devices; it is designed to store large amounts of input and output data which is accessed sequentially. In addition, because 3390 devices can only be attached to 3990 storage control units, DFSORT might be able to benefit from using the advanced features of the 3990 storage

control units. “Chapter 3. Environment Considerations” on page 17 describes how DFSORT uses 3990 facilities. Other ways of improving DFSORT processing of the input and output data sets are:

- Use multiple channel paths
- Allocate enough primary space for the output data sets to avoid the need for additional extents.
- Use separate devices for the input and work data sets, and for the output and work data sets. (DFSORT application data sets that are accessed concurrently should reside on separate devices.)

## VIO for DFSORT Data Sets

DFSORT temporary data sets allocated to virtual devices (VIO) can provide significant elapsed time improvements. However, the trade-off for improving elapsed time using VIO is a serious CPU time degradation.

VIO should be used for DFSORT input, work, and output data sets only when:

- Elapsed time is of primary concern, or
- The temporary data sets are passed to and from other job steps whose performance can be improved by using VIO.

To realize elapsed time improvements, VIO should be used for:

- Input and work data sets; or
- Work and output data sets; or
- Input, output, and work data sets.

In a DFSMS environment, data sets used by DFSORT might be allocated to virtual devices by the automatic class selection (ACS) routines, overriding the VIO installation option in some cases. “System-Managed Storage” on page 21 explains how the ACS routines can be changed to avoid VIO allocation for DFSORT temporary data sets.

## Input and Output Data Set Enhancements

You can also use certain enhancements for input and output data sets to improve performance. These enhancements include compression, striping, and SmartBatch for OS/390. See “Compression” on page 13, “Striping” on page 13, and “SmartBatch Pipes” on page 14 for a more detailed description of each of these items.

Using compression, striping, and SmartBatch for OS/390 affects performance as follows:

- The time needed to transfer data is decreased, sometimes dramatically.
- The time needed to perform the DFSORT application is decreased, sometimes dramatically.
- Work data set I/O is much more likely to be a bottleneck in sort applications that use these enhancements. To eliminate the need for work data set I/O when using compression, striping, or SmartBatch for OS/390, do one of the following:
  - Use Hipersorting for all sorting, or
  - Sort entirely in main storage or data space for small to medium size sorts.

## Run-time Options and Performance

Table 4 shows run-time options that influence the performance of DFSORT, a description of each option, any restrictions, the IBM-supplied default value and a possible reason for modifying that value at run-time. All IBM-supplied default values can be changed to site default values using the ICEMAC macro.

*Table 4. Run-time Options That Influence DFSORT Performance*

| Run-time Option | Description | Restriction | IBM-supplied Default Value and Reason for Modifying |
|---|---|---|---|
| SIZE and MAINSIZE | Upper limit for total storage above and below 16MB virtual. | Limited by TMAXLIM or MAXLIM when SIZE=MAX or MAINSIZE=MAX is in effect. | The default is MAX. Modify when sorting unusually large data sets. |
| RESALL | Storage below 16MB virtual that is reserved for system use. | Only used when SIZE=MAX or MAINSIZE=MAX is in effect. Can reduce the amount of virtual storage available for use by DFSORT. | The default is 4KB. Modify when sufficient REGION is specified but application terminates for lack of below 16MB virtual storage. |
| RESINV | Storage below 16MB virtual that is reserved for use by an invoking program. | Only used when SIZE=MAX or MAINSIZE=MAX is in effect and DFSORT is program-invoked. | The default is 0. |
| ARESALL | Storage above 16MB virtual that is reserved for system use. | Can reduce the amount of virtual storage available for use by DFSORT. | The default is 0. |
| ARESINV | Storage above 16MB virtual that is reserved for use by an invoking program. | Only used when DFSORT is program-invoked. | The default is 0. |
| HIPRMAX | Upper limit for Hiperspace for a single application. | Hiperspaces limited by IEFUSI installation exit. | The default is OPTIMAL. Set to 0 to disable Hipersorting. |
| DSPSIZE | Upper limit for data space size. | Data spaces limited by IEFUSI. | The default is MAX. Set to 0 to disable dataspace sorting. |
| DYNALLOC | Requests dynamic allocation and specifies device type and number of work data sets. | | The default is (SYSDA, 4). |
| CINV and NOCINV | Whether control interval access is used for VSAM input data sets. | Improves performance for VSAM input data sets. | The default is CINV. |
| CFW and NOCFW | Whether cache fast write is used for DFSORT work data sets. | Only applicable for work data sets located on DASDs attached to cached 3990 storage control units. Can be used only if DFSORT SVC is installed. Cache fast write can reduce the elapsed time of sorting applications. CFW is more beneficial to DFSORT performance for small and intermediate sized sorts. | The default is YES. Use NOCFW for large sorts. |
| EQUALS and NOEQUALS | Whether input order is preserved for records with equal keys. | Can cause an increase in CPU time. | The default is VLBLKSET. Use NOEQUALS whenever possible. |
| VERIFY and NOVERIFY | Whether output records are checked for correct order. | Can cause an increase in CPU time. | The default is NOVERIFY. Use VERIFY only when necessary. |

*Table 4. Run-time Options That Influence DFSORT Performance (continued)*

| Run-time Option | Description | Restriction | IBM-supplied Default Value and Reason for Modifying |
|---|---|---|---|
| ALTSEQ | Whether a collating sequence other than EBCDIC is used. | Can cause an increase in CPU time. | There is no default value. Use ALTSEQ only when necessary. |
| AVGRLEN | Specifies the average input record length in bytes for variable-length sort applications. | This value is used when necessary to determine the input file size. The resulting value is important for sort applications, since it is used for several internal optimizations as well as for dynamic work data set allocation (see OPTION DYNALLOC). | There is no default value. Using a value close to the actual average record length may improve variable-length record sort performance. |
| FILSZ | Specifies either the exact number of records to be sorte or an estimate of the number of records to be sorted. | The type of value specified can have a significant effect on performance and work data set allocation. See “File Size” on page 45 for more detailed information on this option. | There is no default value. Using a value close to the actual file size may improve sort performance. |
| ODMAXBF | The maximum buffer space to be used for each OUTFIL data set. | Lowering the value can cause performance degradation for the application. When you use more than 2MB, the performance improvements are small except for EXCPs, and, there is an increased need for storage. | The default is 2MB. When you are running OUTFIL applications with a large number of output data sets and constrained storage, use a smaller value to reduce total virtual storage usage. |
| NOBLKSET | Controls the use of Blockset techniques. | If necessary, DFSORT can still use non-Blockset techniques for other reasons. Using non-Blockset techniques significantly degrades performance. | There is no default value. Specify NOBLKSET only temporarily to bypass a Blockset problem. |
| CHALT and NOCHALT | Specifies whether ALTSEQ is to be applied to character format fields (CH). | As with ALTSEQ, can cause an increase in CPU time. | The default is NOCHALT. Use CHALT only when necessary. |
| COBEXIT | Specifies the library for COBOL E15 and E35 routines. | There are performance advantages to using the newer versions of COBOL rather than OS/VS COBOL. | The default is COBEXIT=COB1. Use the value that corresponds to the version of COBOL you are using. |
| EFS | Specifies the name of a user-written Extended Function Support program to be called by DFSORT. | Can cause an increase in CPU time and elapsed time. | The default is EFS=NONE. Use EFS only when necessary for your application. |
| LOCALE | Specifies whether locale processing is to be used and, if so, designates the active locale. | Should only be used when required since it can show degraded performance relative to collation using character encoding values. | The default is LOCALE=NONE. Use LOCALE only when necessary for your application. |

See *Application Programming Guide* for a complete list of run-time override options.

# Chapter 6. Application Considerations

Most sites have many applications involving sorting. DFSORT is often used by these applications either directly by invoking DFSORT with JCL, or indirectly by invoking DFSORT from a program. In particular, the COBOL SORT statement, and the PL/I sort routines result in calls to DFSORT (see the appropriate language books for complete details).

The way in which COBOL interfaces with DFSORT depends on the use of COBOL features such as FASTSRT, NOFASTSRT, USING, GIVING and INPUT and OUTPUT PROCEDUREs, and DFSORT features such as COBOL exits and DFSORT control statements. In general, the featuers you use are dictated by the needs of your application. But in many cases, an application can achieve its results using one or another of these features, that is, you have a choice. This chapter describes some of the COBOL and DFSORT features you can choose and the performance and productivity implications of doing so.

See *COBOL Application Programming Guide* or *Application Programming Guide* for details if you are not familiar with any of the COBOL or DFSORT features described in this chapter. See *Getting Started* for excellent tutorials on DFSORT control statements.

**Notes:**

1. Although this chapter uses VS COBOL II for purposes of illustration and performance comparisons, it applies equally to newer versions of COBOL, that is, COBOL for MVS & VM and COBOL for OS/390 & VM.
2. Many of the techniques described in this chapter can also be used with other languages that can call DFSORT such as PL/I and Assembler.

## VS COBOL II Interfaces to DFSORT

**Programming Interface information**

Understanding the interfaces between DFSORT and languages such as VS COBOL II can help you design more efficient applications. Knowing which interfaces are used enables you to:

- Obtain more information about these interfaces in the DFSORT books.
- Determine the best way to take advantage of these interfaces.

## Invoking DFSORT from COBOL

In order to invoke DFSORT from COBOL, you must code a COBOL SORT statement. This statement has the following variations which affect the way in which DFSORT and COBOL pass information back and forth:

- USING
- GIVING
- INPUT PROCEDURE
- OUTPUT PROCEDURE

In addition, the VS COBOL II compile-time options FASTSRT and NOFASTSRT also affect the interfaces between COBOL and DFSORT.

### Application Considerations

The interfaces that result from the COBOL SORT statement and the FASTSRT/NOFASTSRT compile-time options are described in “Processing with FASTSRT” and “Processing with NOFASTSRT”.

When a COBOL calling program is used, DFSORT control statements can be specified using the COBOL data set defined by IGZSRTCD or the DFSORT data set defined by SORTCNTL or DFSPARM. There are some differences in how these data sets can be used. For example:

- SORTCNTL allows you to specify comment statements, blank statements, remarks, and labels. DFSPARM allows comment statements, blank statements, and remarks, but not labels. IGZSRTCD does not allow comment statements, remarks, or labels, and eliminates blank statements.
- DFSPARM and IGZSRTCD enable you to pass certain DFSORT run-time options (such as MSGDDN) that are ignored in SORTCNTL.
- Using the COBOL special register SORT-CONTROL enables you to pass different IGZSRTCD data sets to DFSORT when you have multiple SORT statements. The statements in IGZSRTCD are actually placed in the parameter list in storage that COBOL passes when it calls DFSORT. A separate parameter list with the appropriate control statements is passed for each call.
- SORTCNTL and DFSPARM enable you to pass DFSORT control statements without increasing the storage used for the parameter list.

For our examples, we use SORTCNTL, although IGZSRTCD or DFSPARM would do just as well.

## Processing with FASTSRT

VS COBOL II’s FASTSRT compile-time option is a special feature of the language that can improve performance for qualifying applications which use the COBOL SORT statement. You should always specify the FASTSRT option. VS COBOL II decides automatically at compile-time whether FASTSRT can actually be used. For example, FASTSRT cannot be used for input processing when an INPUT PROCEDURE is specified. See *COBOL Application Programming Guide* for the complete list of FASTSRT requirements and restrictions.

When FASTSRT is in effect for input processing, VS COBOL II passes the ddname of the input data set to DFSORT. DFSORT uses this ddname to read the input data set directly.

When FASTSRT is in effect for output processing, VS COBOL II passes the ddname of the output data set to DFSORT. DFSORT uses this ddname to write the output data set directly.

Input or output processing by DFSORT rather than COBOL can result in reductions in CPU time, EXCPs, and elapsed time.

**Notes:**

1. OS/VS COBOL has no equivalent to FASTSRT.
2. PL/I’s PLISRTA subroutine is equivalent to using FASTSRT for both input and output processing. PLISRTB is equivalent to using FASTSRT for output processing and PLISRTC is equivalent to using FASTSRT for input processing.

## Processing with NOFASTSRT

When NOFASTSRT is in effect for input processing, the USING or INPUT PROCEDURE causes COBOL to generate a DFSORT E15 user exit routine and

pass its address to DFSORT. When an INPUT PROCEDURE is used, COBOL incorporates the INPUT PROCEDURE code into the E15 routine it generates.

DFSORT does not read the input data set directly, but instead obtains all the input records from the E15 routine. The E15 routine generated by COBOL reads the input data set and passes one record to DFSORT each time it is called.

When NOFASTSRT is in effect for output processing, the GIVING or OUTPUT PROCEDURE causes COBOL to generate a DFSORT E35 user exit routine and pass its address to DFSORT. When an OUTPUT PROCEDURE is used, COBOL incorporates the OUTPUT PROCEDURE code into the E35 routine it generates.

DFSORT does not write the output data set directly, but instead passes all the output records to the E35 routine. DFSORT calls the E35 routine generated by COBOL once for each record so the E35 routine can write the record to the output data set.

Input or output processing by COBOL rather than DFSORT can result in degraded performance.

**Notes:**

1. OS/VS COBOL's input and output processing is equivalent to using NOFASTSRT.
2. PL/I's PLISRTD subroutine is equivalent to using NOFASTSRT for both input and output processing. PLISRTC is equivalent to using NOFASTSRT for output processing and PLISRTB is equivalent to using NOFASTSRT for input processing.

## Performance

When FASTSRT is in effect for a COBOL sort, DFSORT reads the input data set and writes the output data set rather than COBOL. This results in reductions in elapsed time, CPU time and EXCPs. An example of the benefits of FASTSRT is shown in Figure 9 on page 60. For this comparison, the same COBOL sort with USING and GIVING was run with FASTSRT and NOFASTSRT.

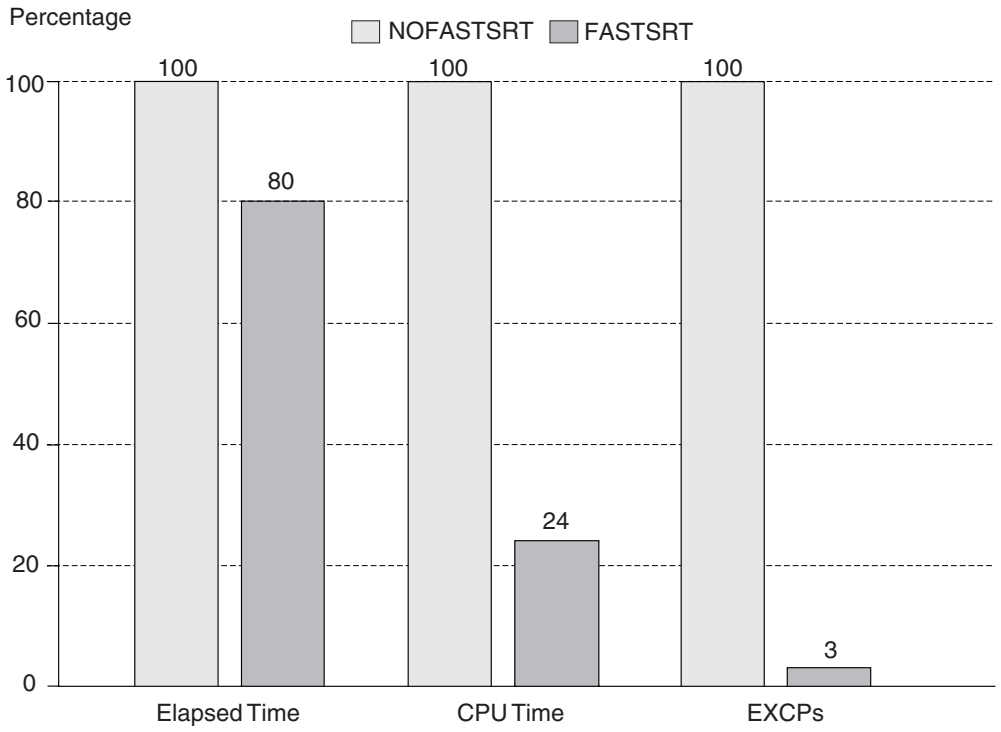



*Figure 9. Benefits of FASTSRT*

**End of Programming Interface information**

## Sample Sorting Application

The remainder of this chapter describes and compares three different methods for doing the following application:

1. Pre-processing: Delete all input records which have zero in a particular field (call it the OMIT field). The OMIT field is columns 30-34 of the records.
2. Sorting: Sort the remaining records in descending order by a particular character format field (call it the KEY field). The KEY field is in columns 5-24 of the records.
3. Post-processing: Write one output record with each key.

The three methods are as follows:

- Method 1: COBOL program with INPUT/OUTPUT PROCEDUREs
- Method 2: COBOL program with DFSORT control statements
- Method 3: DFSORT with control statements

## Method 1: COBOL Program with INPUT/OUTPUT PROCEDUREs

**Programming Interface information**

Method 1 uses a SORT statement INPUT PROCEDURE for pre-processing and a SORT statement OUTPUT PROCEDURE for post-processing.

NOFASTSRT is in effect for input and output processing due to the use of INPUT and OUTPUT PROCEDUREs.

An INPUT PROCEDURE or OUTPUT PROCEDURE can add, delete, alter, edit, or otherwise modify the records.

The INPUT PROCEDURE and OUTPUT PROCEDURE are actually special forms of E15 and E35 exits which are called by DFSORT during its processing, but controlled by the COBOL calling program.

The INPUT PROCEDURE is responsible for supplying all of the input records to be sorted to DFSORT, while the OUTPUT PROCEDURE is responsible for disposing of all the records after they are sorted.

## COBOL Calling Program

```
*-----------------------------------------------------------------
*  METHOD 1: COBOL INPUT AND OUTPUT PROCEDURES.
*-----------------------------------------------------------------
*  1. PRE-PROCESS:
*     THE PROGRAM USES A SORT INPUT PROCEDURE TO DELETE RECORDS
*     WITH A ZZZZZ OMIT FIELD BEFORE SORTING. THE OMIT FIELD IS
*     IN COLUMNS 30-34.
*  2. SORT
*     THE PROGRAM CALLS DFSORT TO SORT THE RECORDS IN DESCENDING
*     ORDER. THE KEY IS IN COLUMNS 5-24.
*  3. POST-PROCESS:
*     THE PROGRAM USES A SORT OUTPUT PROCEDURE TO WRITE ONE
*     RECORD WITH EACH KEY AFTER SORTING.
*
*  INPUT/OUTPUT: READS INDS AND WRITES OUTDS.
*                DFSORT PASSES RECORDS TO THE PROCEDURES.
*-----------------------------------------------------------------
 ID DIVISION.
   PROGRAM-ID. CASE1.
 ENVIRONMENT DIVISION.
   INPUT-OUTPUT SECTION.
   FILE-CONTROL.
        SELECT INDS    ASSIGN TO INDS.
        SELECT OUTDS   ASSIGN TO OUTDS.
        SELECT SORT-FILE  ASSIGN TO SORTFILE.
```

*Figure 10. COBOL Calling Program for Method 1 (Part 1 of 3)*

```
 DATA DIVISION.
   FILE SECTION.
   FD INDS RECORD CONTAINS 160 CHARACTERS
      LABEL RECORD STANDARD BLOCK 27840
      DATA RECORDS ARE INDS-RECORD.
   01 INDS-RECORD.
    05 FILLER        PIC X(4).
    05 INDS-KEY      PIC X(20).
    05 FILLER        PIC X(5).
    05 INDS-OMIT     PIC X(5).
    05 FILLER        PIC X(126).
   FD OUTDS RECORD CONTAINS 160 CHARACTERS
     LABEL RECORD STANDARD BLOCK 27840
     DATA RECORDS ARE OUTDS-RECORD.
  01 OUTDS-RECORD.
   05 FILLER          PIC X(160).
  SD SORT-FILE RECORD CONTAINS 160 CHARACTERS
     DATA RECORD SORT-RECORD.
  01 SORT-RECORD.
   05 FILLER        PIC X(4).
   05 SORT-KEY      PIC X(20).
   05 FILLER        PIC X(136).
  WORKING-STORAGE SECTION.
  01 FLAGS.
   05 INDS-EOF     PIC X VALUE SPACE.
      88 SFLAG     VALUE "Y".
   05 TEMP-EOF     PIC X VALUE SPACE.
      88 TFLAG     VALUE "Y".
  01 PSTART     PIC 9(1) VALUE 0.
  01 SAVE-KEY   PIC X(20).
   01 TEMP-RECORD.
    05 FILLER        PIC X(4).
    05 TEMP-KEY      PIC X(20).
    05 FILLER        PIC X(136).
   PROCEDURE DIVISION.
   MASTER SECTION.
*-------------------------------------------------------------
*     CALL DFSORT TO SORT THE RECORDS IN DESCENDING ORDER.
*-------------------------------------------------------------
       SORT SORT-FILE
        ON DESCENDING KEY SORT-KEY
        INPUT PROCEDURE INPUT-PROC
        OUTPUT PROCEDURE OUTPUT-PROC.
       IF SORT-RETURN > 0
          DISPLAY "SORT FAILED".
       STOP RUN.
```

*Figure 10. COBOL Calling Program for Method 1 (Part 2 of 3)*

```
*-------------------------------------------------------------
*     SORT INPUT PROCEDURE:
*       READ INDS.
*       DELETE ALL RECORDS WITH A 'ZZZZZ' OMIT FIELD.
*       SEND ALL OTHER RECORDS TO DFSORT FOR SORTING.
*-------------------------------------------------------------
   INPUT-PROC SECTION.
      OPEN INPUT INDS
      READ INDS AT END SET SFLAG TO TRUE
      END-READ
      PERFORM UNTIL SFLAG
        IF INDS-OMIT NOT = "ZZZZZ"
          RELEASE SORT-RECORD FROM INDS-RECORD
        END-IF
        READ INDS AT END SET SFLAG TO TRUE
        END-READ
      END-PERFORM.
      CLOSE INDS.
*-------------------------------------------------------------
*     SORT OUTPUT PROCEDURE:
*       RECEIVE RECORDS FROM DFSORT INTO TEMP.
*       WRITE ONE RECORD WITH EACH KEY TO OUTDS.
*-------------------------------------------------------------
    OUTPUT-PROC SECTION.
       OPEN OUTPUT OUTDS
       RETURN SORT-FILE INTO TEMP-RECORD AT END SET TFLAG TO TRUE
       END-RETURN
       PERFORM UNTIL TFLAG
         IF PSTART = 0
 *-------------------------------------------------------------
 *     FIRST RECORD - SAVE KEY AND WRITE RECORD TO OUTDS.
 *-------------------------------------------------------------
           MOVE TEMP-KEY TO SAVE-KEY
           WRITE OUTDS-RECORD FROM TEMP-RECORD
           MOVE 1 TO PSTART
         ELSE
           IF TEMP-KEY NOT = SAVE-KEY
*-------------------------------------------------------------
*     RECORD HAS NEW KEY - SAVE KEY AND WRITE RECORD TO OUTDS.
*-------------------------------------------------------------
            MOVE TEMP-KEY TO SAVE-KEY
            WRITE OUTDS-RECORD FROM TEMP-RECORD
          END-IF
        END-IF
        RETURN SORT-FILE INTO TEMP-RECORD
          AT END SET TFLAG TO TRUE
        END-RETURN
      END-PERFORM.
      CLOSE OUTDS.
```

*Figure 10. COBOL Calling Program for Method 1 (Part 3 of 3)*

## Operation (NOFASTSRT in Effect)

The SORT statement results in a call to DFSORT with a parameter list that contains a SORT control statement and other information.

DFSORT treats the INPUT PROCEDURE as an E15 user exit which must supply all the input records. DFSORT calls the INPUT PROCEDURE once for each input record. The INPUT PROCEDURE reads the input data set and uses RELEASE to pass each record with a non-zero OMIT field to DFSORT.

DFSORT sorts the records passed to it by the INPUT PROCEDURE as requested by the SORT statement passed to it by the calling program.

### Application Considerations

DFSORT treats the OUTPUT PROCEDURE as an E35 user exit which must dispose of all the output records. DFSORT calls the OUTPUT PROCEDURE once for each sorted record. The OUTPUT PROCEDURE uses RETURN to obtain the records passed from DFSORT and writes one record with each key to the output data set.

## Performance

Since the COBOL program's INPUT and OUTPUT PROCEDUREs must, by definition, read and write the data sets, NOFASTSRT is in effect for Method 1. FASTSRT is in effect for the other methods we will describe, providing significant performance improvements.

**End of Programming Interface information**

# Method 2: COBOL Program with DFSORT Control Statements

**Programming Interface information**

Method 2 eliminates the COBOL pre-processing and post-processing code by using DFSORT control statements to provide the equivalent functions. An OMIT statement is used instead of an E15, and a SUM statement is used instead of an E35.

FASTSRT is used for input and output processing.

DFSORT provides a powerful set of collating and editing functions available though the use of control statements and options. Collating and editing functions can be used to replace program code. They are designed to adapt to the run-time characteristics of an application in order to provide significant performance benefits.

DFSORT's most significant collating and editing functions are:

- SORT or MERGE: enable you to override the SORT or MERGE control statement generated by the compiler. The override statement can contain the DFSORT year 2000 field format (Y2x).
- INCLUDE or OMIT: enable you to include or delete records whose fields meet certain criteria.
- INREC and OUTREC: enable you to delete and rearrange fields in your records, and to insert blanks, zeros, and constants in records.
- SUM: enables you to sum fields in records with equal keys and to keep only one record with each key.
- OUTFIL: enables you to perform a wide variety of tasks (for example, subsets, editing, reports, and conversion) for multiple output data sets.
- SKIPREC and STOPAFT: enable you to delete records at the beginning or end of your data set.

When a COBOL calling program is used, INCLUDE, OMIT, INREC, OUTREC, SUM, and OUTFIL can be specified in the data set defined by IGZSRTCD, SORTCNTL, or DFSPARM. SKIPREC and STOPAFT can be specified on an OPTION statement in the data set defined by IGZSRTCD, SORTCNTL, or DFSPARM.

The figures that follow show the COBOL calling program and DFSORT control statements for Method 2.

## Operation (FASTSRT in Effect)

The SORT statement results in a call to DFSORT with a parameter list that contains a SORT control statement and other information.

DFSORT reads the input data set and deletes each record with a zero OMIT field as requested by the OMIT statement.

DFSORT sorts the remaining records as requested by the SORT statement passed from the calling program.

DFSORT writes one record with each key to the output data set as requested by the SUM statement.

## Productivity

The use of DFSORT editing functions rather than COBOL code reduces considerably the effort required to perform the application. Source code for pre-processing and post-processing is eliminated along with the time to compile and debug the code.

In addition, future changes to the editing functions performed by the application can be made by simply changing the control statements. Access to source code or recompiles are not necessary.

## Control Statements

```
//SORTCNTL DD *
   OMIT COND=(30,5,CH,EQ,C'ZZZZZ')
   SUM FIELDS=NONE
```

*Figure 11. DFSORT Control Statements for Method 2*

## COBOL Calling Program

```
*-------------------------------------------------------------
*  METHOD 2: COBOL MAIN PROGRAM.
*-------------------------------------------------------------
*  1. PRE-PROCESS:
*     A DFSORT OMIT CONTROL STATEMENT DELETES RECORDS WITH A
*     'ZZZZZ' OMIT FIELD BEFORE SORTING. THE OMIT FIELD IS IN
*     IN COLUMNS 30-34.
*  2. SORT
*     THE PROGRAM CALLS DFSORT TO SORT THE RECORDS IN DESCENDING
*     ORDER. THE KEY IS IN COLUMNS 5-24.
*  3. POST-PROCESS:
*     A DFSORT SUM CONTROL STATEMENT WRITES ONE RECORD WITH
*     EACH KEY AFTER SORTING.
*
*  INPUT/OUTPUT: DFSORT READS SORTIN AND WRITES SORTOUT.
*-------------------------------------------------------------
 ID DIVISION.
  PROGRAM-ID. CASE2.
 ENVIRONMENT DIVISION.
  INPUT-OUTPUT SECTION.
    FILE-CONTROL.
       SELECT SORTIN  ASSIGN TO SORTIN.
       SELECT SORTOUT ASSIGN TO SORTOUT.
       SELECT SORT-FILE  ASSIGN TO SORTFILE.
 DATA DIVISION.
   FILE SECTION.
    FD SORTIN RECORD CONTAINS 160 CHARACTERS
       LABEL RECORD STANDARD BLOCK 27840
       DATA RECORDS ARE SORTIN-RECORD.
    01 SORTIN-RECORD.
      05 FILLER          PIC X(160).
    FD SORTOUT RECORD CONTAINS 160 CHARACTERS
       LABEL RECORD STANDARD BLOCK 27840
       DATA RECORDS ARE SORTOUT-RECORD.
    01 SORTOUT-RECORD.
      05 FILLER          PIC X(160).
    SD SORT-FILE RECORD CONTAINS 160 CHARACTERS
       DATA RECORD SORT-RECORD.
     01 SORT-RECORD.
      05 FILLER        PIC X(4).
      05 SORT-KEY      PIC X(20).
      05 FILLER        PIC X(136).
    WORKING-STORAGE SECTION.
  PROCEDURE DIVISION.
    MASTER SECTION.
```

*Figure 12. COBOL Calling Program for Method 2 (Part 1 of 2)*

```
*-------------------------------------------------------------
*     CALL DFSORT TO SORT THE RECORDS IN DESCENDING ORDER.
*-------------------------------------------------------------
      SORT SORT-FILE
        ON DESCENDING KEY SORT-KEY
        USING SORTIN
        GIVING SORTOUT.
      IF SORT-RETURN > 0
        DISPLAY "SORT FAILED".
      STOP RUN.
```

*Figure 12. COBOL Calling Program for Method 2 (Part 2 of 2)*

## Performance

Since Method 2 uses USING and GIVING rather than INPUT and OUTPUT PROCEDUREs, FASTSRT is in effect. In addition, by eliminating the overhead related to passing each record between DFSORT and the user exits, and enabling DFSORT to use its highly optimized editing code, Method 2 achieves significant performance gains in CPU time, elapsed time, and EXCPs compared to Method 1. Figure 13 shows a performance comparison between Method 1 and Method 2.

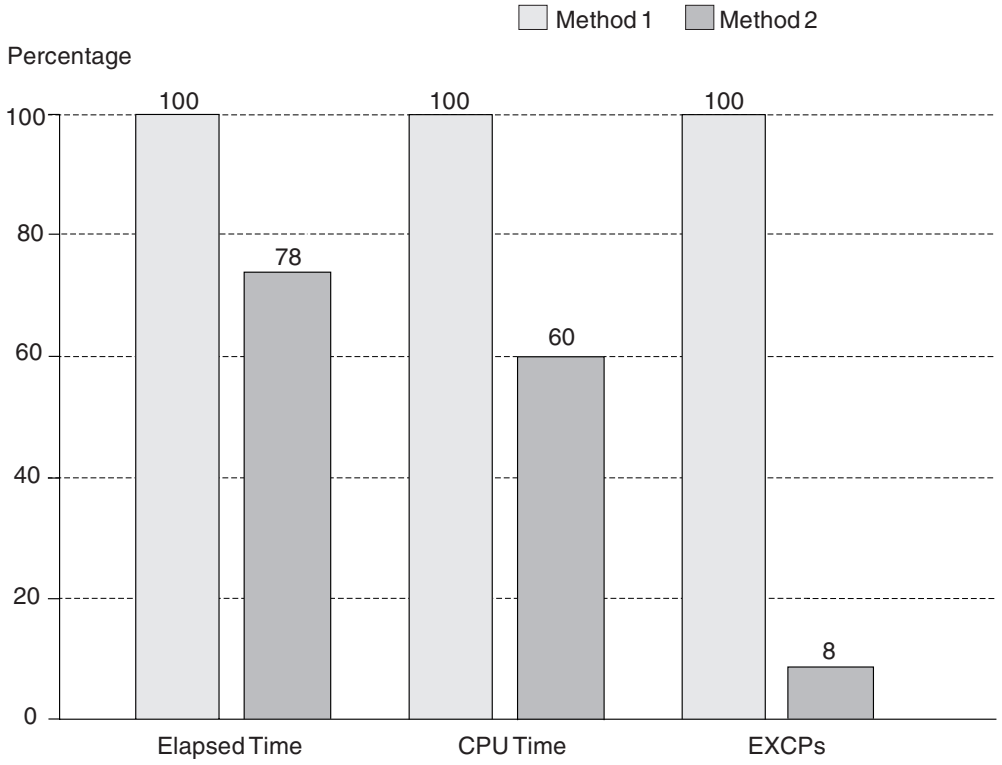



*Figure 13. Method 1 vs Method 2 Performance Comparison*

**End of Programming Interface information**

## Method 3: DFSORT with Control Statements

Method 3 takes the final step of eliminating the COBOL calling program by invoking DFSORT directly (using PGM=ICEMAN or PGM=SORT on the JCL EXEC statement). Calling DFSORT directly is only feasible if all of the functions of the COBOL program can be duplicated using DFSORT control statements, options, or user exits.

The SORT statement of the COBOL program is replaced by an equivalent DFSORT control statement, SORT.

When DFSORT is invoked directly, control statements can be specified using the data set defined by SYSIN or DFSPARM. DFSPARM can be used to specify certain options (for example, MSGDDN) that are ignored in SYSIN. For this example, we use SYSIN, although DFSPARM would do just as well.

## Control Statements

```
//SYSIN DD *
   OMIT COND=(30,5,CH,EQ,C'ZZZZZ')
   SORT FIELDS=(5,20,CH,D)
   SUM FIELDS=NONE
```

*Figure 14. DFSORT Control Statements for Method 4*

## Operation

The system calls DFSORT directly.

DFSORT reads the input data set and deletes each record with a zero OMIT field as requested by the OMIT statement.

DFSORT sorts the remaining records as requested by the SORT control statement.

DFSORT writes one record with each key to the output data set as requested by the SUM statement.

## Productivity

Elimination of all COBOL code reduces significantly the effort required to perform an application. You can use control statements to duplicate complex code logic quickly and effectively. Source code for pre-processing, sorting, and post-processing is eliminated along with the time to compile and debug the code.

In addition, future changes to the editing and sorting functions performed by the application can be made by simply changing the control statements. Access to source code, and recompiles, are not necessary.

## Performance

Performance for Method 3 is comparable to that for Method 2.

# Chapter 7. DFSORT Performance Data

Tuning activity is often started when a particular job is taking too long to complete or is using system resources excessively. But if such a “problem job” does not exist at your site, where do you start to look for ways to improve DFSORT’s performance or balance its use of resources with other requirements?

The purpose of this chapter is to outline the actions available for those who want to tune the use of DFSORT at their site. It also shows what information you need about DFSORT, where to find it, and what methods are available for collecting the data, including use of the DFSORT installation exits ICEIEXIT and ICETEXIT. The specific topics include:

- Where to find performance indicators
- An overview of DFSORT performance information
- Sources of DFSORT performance information
- Techniques for analyzing DFSORT performance data
- Balancing DFSORT requirements with system resources
- Understanding trade-offs in improving performance

## DFSORT Performance Indicators

Performance indicators can be found in a number of different places. Where you choose to find them depends on your own knowledge of OS/390 or MVS/ESA, available tools, and how much effort you want to spend.

Sources you can use include:

**JES Log**

For batch jobs, JES writes a job log including messages issued and system accounting information. Figure 15 on page 70 shows a sample JES2 log for a DFSORT job. The JES2 log produced at your site will, of course, be different.

```
                    J E S 2  J O B  L O G  --  S Y S T E M  E D S 3  --  N O D E  S N J M A S 3

16.47.30 JOB12904 ---- WEDNESDAY, 22 APR 1998 ----
16.47.30 JOB12904  ICH70001I Y897797  LAST ACCESS AT 16:47:17 ON WEDNESDAY, APRIL 22, 1998
16.47.30 JOB12904  $HASP373 TGSORT   STARTED - INIT    1 - CLASS A - SYS EDS3
16.47.36 JOB12904  -                                          --TIMINGS (MINUTES)--
16.47.36 JOB12904  -JOBNAME  STEPNAME PROGRAM     RC   EXCP    CPU    SRB  CLOCK   SERV  PG
16.47.36 JOB12904  -TGSORT   SORT     ICEMAN    0000    154    .00    .00     .0  62008  31
16.47.36 JOB12904  -TGSORT   ENDED. TOTAL CPU TIME=   .00  TOTAL ELAPSED TIME=    .0 MINUTE
16.47.36 JOB12904  $HASP395 TGSORT   ENDED
------ JES2 JOB STATISTICS ------
  22 APR 1998 JOB EXECUTION DATE
           14 CARDS READ
          132 SYSOUT PRINT RECORDS
            0 SYSOUT PUNCH RECORDS
            7 SYSOUT SPOOL KBYTES
         0.09 MINUTES EXECUTION TIME
        1 //TGSORT JOB (YAEGER,M72,050,005),CLASS=A,                              JOB12904
          // MSGCLASS=H,USER=Y897797,MSGLEVEL=(1,1),TIME=(,15),NOTIFY=Y897797,
          // PASSWORD=
        2 //JOBLIB   DD DSN=SORTDEV.R14DEV.SICELNKN,DISP=SHR
        3 //         DD DSN=SORTDEV.R14DEV.SICELINK,DISP=SHR
        4 //SORT    EXEC  PGM=ICEMAN
        5 //SORTDIAG  DD  DUMMY
        6 //SYSOUT    DD  SYSOUT=*
        7 //SYSUDUMP  DD  SYSOUT=*
        8 //SORTIN DD DSN=Y897797.TG.INPUT,DISP=SHR
        9 //SORTOUT DD DSN=&&OUTPUT,UNIT=SYSDA,SPACE=(CYL,(30,10),RLSE),
          // DISP=(,PASS)
       10 //SYSIN    DD    *
ICH70001I Y897797  LAST ACCESS AT 16:47:17 ON WEDNESDAY, APRIL 22, 1998
IEF236I ALLOC. FOR TGSORT SORT
IEF237I 4241 ALLOCATED TO JOBLIB
IEF237I 4241 ALLOCATED TO
IEF237I DMY  ALLOCATED TO SORTDIAG
IEF237I JES2 ALLOCATED TO SYSOUT
IEF237I JES2 ALLOCATED TO SYSUDUMP
IGD103I SMS ALLOCATED TO DDNAME SORTIN
IGD100I 1497 ALLOCATED TO DDNAME SORTOUT  DATACLAS (         )
IEF237I JES2 ALLOCATED TO SYSIN
IGD100I 1485 ALLOCATED TO DDNAME SORTWK01 DATACLAS (         )
IGD100I 1F03 ALLOCATED TO DDNAME SORTWK02 DATACLAS (         )
IGD100I 1487 ALLOCATED TO DDNAME SORTWK03 DATACLAS (         )
IGD100I 14F3 ALLOCATED TO DDNAME SORTWK04 DATACLAS (         )
IEF285I   SYS98112.T164731.RA000.TGSORT.R0344103       DELETED
IEF285I   VOL SER NOS= SCR301.
IEF285I   SYS98112.T164731.RA000.TGSORT.R0344104       DELETED
IEF285I   VOL SER NOS= SCR302.
IEF285I   SYS98112.T164731.RA000.TGSORT.R0344105       DELETED
IEF285I   VOL SER NOS= SCR303.
IEF285I   SYS98112.T164731.RA000.TGSORT.R0344106       DELETED
IEF285I   VOL SER NOS= SCR306.
IEF142I TGSORT SORT - STEP WAS EXECUTED - COND CODE 0000
IEF285I   SORTDEV.R14DEV.SICELNKN                      PASSED
IEF285I   VOL SER NOS= SRTLIB.
IEF285I   SORTDEV.R14DEV.SICELINK                      PASSED
IEF285I   VOL SER NOS= SRTLIB.
IEF285I   Y897797.TGSORT.JOB12904.D0000102.?           SYSOUT
IEF285I   Y897797.TGSORT.JOB12904.D0000103.?           SYSOUT
IGD104I Y897797.TG.INPUT                              RETAINED,  DDNAME=SORTIN
IEF285I   SYS98112.T164730.RA000.TGSORT.OUTPUT.H03     PASSED
IEF285I   VOL SER NOS= SCR305.
IEF285I   Y897797.TGSORT.JOB12904.D0000101.?           SYSIN
IEF373I STEP/SORT    /START 1998112.1647
IEF374I STEP/SORT    /STOP  1998112.1647 CPU    0MIN 00.43SEC SRB    0MIN 00.02SEC VIRT  1120K SYS   400K EXT
IEF285I   SORTDEV.R14DEV.SICELNKN                      KEPT
IEF285I   VOL SER NOS= SRTLIB.
IEF285I   SORTDEV.R14DEV.SICELINK                      KEPT
IEF285I   VOL SER NOS= SRTLIB.
IEF237I 1497 ALLOCATED TO SYS00001
IEF285I   SYS98112.T164736.RA000.TGSORT.R0344107       KEPT
IEF285I   VOL SER NOS= SCR305.
IEF285I   SYS98112.T164730.RA000.TGSORT.OUTPUT.H03     DELETED
IEF285I   VOL SER NOS= SCR305.
IEF375I  JOB/TGSORT  /START 1998112.1647
IEF376I  JOB/TGSORT  /STOP  1998112.1647 CPU    0MIN 00.43SEC SRB    0MIN 00.02SEC
```

*Figure 15. A Sample JES2 Log*

### Messages

Some programs (like DFSORT) issue messages quantifying their use of certain system resources. This is the simplest way of accessing such information. In the case of DFSORT, when a SORTDIAG DD statement is present or the ICEMAC option DIAGSIM=YES has been specified for your site, additional messages useful for tuning are issued. An example of the

DFSORT messages is shown in Figure 16. See *Messages, Codes, and Diagnosis Guide* for explanations of these messages.

```
ICE805I 1 JOBNAME: TGSORT   , STEPNAME: SORT
ICE802I 0 BLOCKSET     TECHNIQUE IN CONTROL
ICE143I 0 BLOCKSET     SORT  TECHNIQUE SELECTED
ICE000I 1 - CONTROL STATEMENTS FOR 5740-SM1, DFSORT REL 14.0 - 16:47 ON WED APR 22, 1998 -
            SORT FIELDS=(1,4,BI,A)
ICE193I 0 ICEAM1 ENVIRONMENT IN EFFECT - ICEAM1 INSTALLATION MODULE SELECTED
ICE088I 2 TGSORT  .SORT     .          , INPUT LRECL = 100, BLKSIZE = 27900, TYPE = FB
ICE093I 0 MAIN STORAGE = (MAX,4194304,4194304)
ICE156I 0 MAIN STORAGE ABOVE 16MB = (4107248,4107248)
ICE127I 0 OPTIONS: OVFLO=RC0 ,PAD=RC0 ,TRUNC=RC0 ,SPANINC=RC16
ICE128I 0 OPTIONS: SIZE=4194304,MAXLIM=1048576,MINLIM=450560,EQUALS=N,LIST=Y,ERET=RC16 ,MSGDDN=SYSOUT
ICE129I 0 OPTIONS: VIO=N,RESONT=NONE,SMF=NO    ,WRKSEC=Y,OUTSEC=Y,VERIFY=N,CHALT=N,DYNALOC=(SYSDA    ,004),ABCODE=MSG
ICE130I 0 OPTIONS: RESALL=4096,RESINV=0,SVC=109  ,CHECK=Y,WRKREL=Y,OUTREL=Y,CKPT=N,STIMER=Y,COBEXIT=COB1
ICE131I 0 OPTIONS: TMAXLIM=4194304,ARESALL=0,ARESINV=0,OVERRGN=65536,CINV=Y,CFW=Y,DSA=0
ICE132I 0 OPTIONS: VLSHRT=N,ZDPRINT=N,IEXIT=N,TEXIT=N,LISTX=N,EFS=NONE     ,EXITCK=S,PARMDDN=DFSPARM ,FSZEST=N
ICE133I 0 OPTIONS: HIPRMAX=OPTIMAL,DSPSIZE=MAX ,ODMAXBF=0
ICE084I 0 EXCP ACCESS METHOD USED FOR SORTOUT
ICE084I 0 EXCP ACCESS METHOD USED FOR SORTIN
ICE890I 0 DC 15010200 TC 0 CS DSVVV KSZ 4 VSZ 4
ICE887I 0 CSES 613276,760271,2023227 ES 353831,379040,1641996
ICE886I 0 SYS 284899 TSTG 0 FS 3959 INIT 40 MAX 3959 LEN 0
ICE998I 0 FSZ=150102 RC  IGN=0 E  AVG=100 0  WSP=19496 C  DYN=0 0
ICE996I 0 ESM=2147483392,ESO=2147483392,ESR=0,ESP=4096,ESS=16384,CES=499712
ICE997I 0 HWSP=5130,HMAX=379040,HES=376832,ASV=379040,EQ=11
ICE898I 0 OMAX=353831,NMAX=379040,TMAX=1641996,CMAX=379040,HU=100,BUN=61408,MD=H  ,DU=0,DR=10000
ICE880I 0 QP=2 QA=3 HI=92 LI=91 MI=93 TZ=8192 N1=5130 N2=5130 SZ=94
ICE889I 0 CT=MAX
ICE902I 0 0 AT00  I AT11
ICE900I 0 CON=1,MUV=0,VOL=1,ENU=0,SBK=1,SRC=177
ICE999I 0 PWK=0 PSP=0 SWK=0 SSP=0 TWK=0 TSP=0 RWK=0 RSP=0 AWK=0
ICE090I 0 OUTPUT LRECL = 100, BLKSIZE = 27900, TYPE = FB   (SDB)
ICE055I 0 INSERT 0, DELETE 0
ICE054I 0 RECORDS - IN: 150000, OUT: 150000
ICE134I 0 NUMBER OF BYTES SORTED: 15000000
ICE165I 0 TOTAL WORK DATA SET TRACKS ALLOCATED: 0 , TRACKS USED: 0
ICE180I 0 HIPERSPACE STORAGE USED = 15300K BYTES
ICE188I 0 DATA SPACE STORAGE USED = 0K BYTES
ICE891I 1 4153944 WMAIN, 10664 CMAIN, MAX CALLOC, N SCN, B BA, 0 AZ, 0 BZ
ICE892I 1 100 RIN  27900 BLI  27900 BLO  100 RUN  61408 BUN  9672 CPU  00 CVC
ICE893I 1 72 XIN 35 WIN 0 GIN NDEN PFP00 B00 CM000 CIX UPTH LMK
ICE894I 0 8 STR  8 MOR  33 IPB  73 OPB  0 CYL  0   MN
ICE881I 0 EQ=11 DX=0 D2=1 D3=0 D4=1 AS=0 SA=0 SB=0 SC=0
ICE885I 0 DAT 00 DSR 0402 BINS 0 BSZ 0 RCP 0 AJC 0 RLC 0 DUNIT 0
ICE895I 0 384 MUNIT  0 SUNIT  48 OUNIT
ICE896I 0 8 SET  0 DEXTOT  0 BLK   150102 CSZ  0 WE
ICE910I 0 C5C6C7C8E4C9E5E6E7CFCED3EDE8
ICE804I 1 SORTWK04 EXCP COUNT: 0
ICE804I 1 SORTWK03 EXCP COUNT: 0
ICE804I 1 SORTWK02 EXCP COUNT: 0
ICE804I 1 SORTWK01 EXCP COUNT: 0
ICE804I 1 SORTOUT  EXCP COUNT: 8
ICE804I 1 SORTIN   EXCP COUNT: 8
ICE899I 0 HSR=11,HSW=9,HRE=256,HWE=9,HRP=3840,HWP=3840,HWM=3825
ICE052I 0 END OF DFSORT
```

*Figure 16. DFSORT Messages. SORTDIAG DD was present.*

**ICEIEXIT**

The installation initialization exit (ICEIEXIT) allows you to examine certain installation and run-time information for each DFSORT application. The ICEIEXIT routine could be used to collect the information relevant to performance, and to write this information to a data set which can be analyzed. Refer to “Using ICEIEXIT Data” on page 79 for information on using ICEIEXIT data to do a moderate analysis of your DFSORT applications. Refer to “Installation Exits” on page 39 for a brief overview of ICEIEXIT.

**SMF** System management facilities (SMF) collects a variety of system and application-related information into a number of different SMF records written by the system and various programs.

SMF type-30 (subtype 4) records, for instance, are written for each job step, and summarize the step’s consumption of major system resources, such as CPU time (broken down into these fields: TCB, SRB, RCT, HPT, and IIP), elapsed time, EXCP counts, and device connect times. In addition, DFSORT can write an SMF type-16 record, which summarizes the key

statistics from a particular DFSORT run. By running reports against SMF data, sites can use this information for workload analysis, planning, and accounting purposes.

To extract performance information from SMF data, you can run an SMF report, use a data reduction program (refer to “Service Level Reporter” on page 75 for a description of one such program), use the analysis and reporting features of DFSORT or its ICETOOL utility, or write a program of your own. Since SMF data often contains accounting and other sensitive data, access might be limited.

**ICETEXIT**
The installation termination exit (ICETEXIT) allows you to collect extensive performance-related information about all DFSORT applications at a site. ICETEXIT provides comprehensive data for each DFSORT application including the information contained in DFSORT's type-16 SMF record. Refer to “Using ICETEXIT Data” on page 80 for information on using ICETEXIT data to do a moderate analysis of your DFSORT applications. Refer to “Installation Exits” on page 39 for a brief overview of ICETEXIT.

**RMF** Resource Measurement Facility (RMF) measures system, address space, and workload activity. You can use RMF to generate reports online or off-line, or as a real-time monitor. Most of these reports concern performance-related statistics for processor, storage, I/O devices, and system data sets.

RMF's primary use is for system-level tuning, but it can also detail the performance characteristics of subsystems and workloads. Many system programmers rely heavily on RMF reports for these analysis activities. RMF writes a series of records into the SMF system data sets which can later be processed by RMF, SLR, or other SMF analysis programs.

As for DFSORT performance, RMF can be used to understand DFSORT's use of resources, to discern the effects of running DFSORT applications, and to highlight contention for resources. For example, it can help you identify situations where multiple work data sets are being placed on the same DASD, by showing a high device utilization for that particular DASD. As with SMF, access to RMF and its data might be restricted.

For a detailed cross reference of sources of performance indicators, refer to Table 5 on page 73.

## Overview of DFSORT Performance Information

Table 5 on page 73 lists the main DFSORT performance areas, sources of information about each area, and suggested analysis activities for each area.

Each level of analysis discussed in “Analysis Techniques for DFSORT Performance Data” on page 77 produces a set of data to interpret. This figure is intended to help you associate that data with certain performance areas in DFSORT. For example, if you have decided to do a simple analysis and are concerned with your use of Blockset, you would check DFSORT messages ICE143I and ICE800I to determine whether or not the Blockset technique is being used, and if it is not, why it is not.

You can also use this figure to help you determine which level of analysis you want to do, given the performance areas you are interested in. For example, if you were

concerned about intermediate merging, you would be able to tell from this figure that you would need to do moderate analysis to obtain data about this performance area.

Table 5 uses the following abbreviations:

**ICExxxx**
: DFSORT messages (The ICE8xxI messages only appear if you have coded the SORTDIAG DD statement or if the ICEMAC option DIAGSIM=YES has been specified for your site.). See *Messages, Codes, and Diagnosis Guide* for explanations of the DFSORT messages.

**JLOG** Job/JES log.

**SMF16**
: SMF type-16 record (issued by DFSORT).

**SMF30**
: SMF type-30 (subtype 4) record.

**IEXIT** Information passed to the DFSORT ICEIEXIT routine.

**TEXIT** Information passed to the DFSORT ICETEXIT routine (includes contents of the SMF type-16 record).

An individual source might not contain all of the information listed under Analysis.

*Table 5. Sources of Performance Indicators*

| Area | Source | Analysis |
|---|---|---|
| Blockset | ICE143I, ICE189A, ICE800I, SMF16, IEXIT, TEXIT | Ensure the Blockset technique is being used or determine why it is not being used. |
| Intermediate merge | TEXIT | Ensure there are no intermediate merges for a Blockset sort (if appropriate). An intermediate merge is a condition caused by a very low virtual storage to data set size ratio, and usually results in significant performance degradation. Providing sufficient virtual storage to DFSORT eliminates intermediate merges. |
| Virtual storage | ICE039A, ICE080I, ICE092I, ICE093I, ICE115A, ICE156I, ICE231I, JLOG, SMF30, TEXIT | Ensure adequate storage is available both above and below 16MB virtual or determine how much more storage is needed. **Note:** JES log messages and SMF type-30 fields can be misleading as to how much storage DFSORT actually uses. The DFSORT messages and ICETEXIT values contain the actual amount of storage used by DFSORT. |
| DFSORT storage options | ICE128I, ICE130I, ICE131I, IEXIT, TEXIT | Ensure SIZE=MAX or MAINSIZE=MAX is used and RESALL, TMAXLIM, MAXLIM, OVERRGN, DSA and MINLIM values are appropriate. |
| Work data sets | ICE129I, ICE165I, JLOG, SMF16, SMF30, TEXIT, | Determine whether dynamic allocation or JCL allocation of work data sets is used, how much work space is allocated and used, and to what types of devices the data sets are allocated. In addition, ensure that the work space is minimized. |
| Hipersorting | ICE133I, ICE180I, IEXIT, SMF16, TEXIT | Ensure Hipersorting is used, if appropriate, and determine how much Hiperspace is used. |
| Dataspace sorting | ICE133I, ICE188I, IEXIT, SMF16, TEXIT | Ensure dataspace sorting is used, if appropriate, and determine how much data space is used. |
| CPU time | JLOG, SMF16, SMF30, TEXIT | Determine the effects of tuning on CPU time. |
| Elapsed time | JLOG, SMF16, SMF30, TEXIT | Determine the effects of tuning on elapsed time. |

*Table 5. Sources of Performance Indicators (continued)*

| Area | Source | Analysis |
|---|---|---|
| EXCPs | ICE804I, JLOG, SMF16, SMF30, TEXIT | Determine the effects of tuning on EXCPs. |
| Device connect time | SMF30 | Determine the effects of tuning on device connect time. |
| Storage control cache | TEXIT | Ensure storage control units with cache are used, and that cache fast write is used. |
| System determined block size | ICE090I, ICE210I | Ensure system-determined block size is used for output data sets, when appropriate. |
| Block sizes | ICE088I, ICE090I, ICE2101, SMF16, TEXIT | Ensure adequately large block sizes are used for input and output data sets. |
| DASD extents | SMF16, TEXIT | Ensure the number of extents required for input, output, and work data sets is minimized. |
| DFSORT SVC | ICE145A, ICE187I, ICE191I, ICE194I, ICE816I | Ensure that the DFSORT SVC is available. |
| Residency | ICE129I | Ensure that DFSORT modules are resident, if appropriate. |
| DFSORT level | ICE000I, SMF16, TEXIT | Ensure that the latest DFSORT level is installed. |
| Control field length | SMF16, TEXIT | Ensure that the control fields are only as long as necessary to distinguish the records. |
| EQUALS | ICE128I, TEXIT, SMF16 | Ensure that the EQUALS option is used only when necessary. |
| VERIFY | ICE129I, IEXIT, TEXIT | Ensure that the VERIFY option is used only when necessary. |
| Number of DASD work data sets and devices | ICE804I, JLOG, SMF16, SMF30, TEXIT | For DASD-only work data set applications, ensure that at least three data sets are used and that the data sets are on separate devices. |
| Type of application (sort, merge, or copy) | ICE143I, SMF16, IEXIT, TEXIT | Understand the type of application and tune accordingly. |
| Record length and format | ICE088I, ICE089I, ICE090I, ICE091I, ICE210I, SMF16, TEXIT | Understand what type of data is being processed, and tune accordingly. |
| Input, work, and output data set sizes | ICE054I, ICE055I, ICE098I, ICE134I, ICE227I, ICE228I, SMF16, TEXIT | Understand the characteristics of the data being processed, and tune accordingly. |
| VIO | JLOG, SMF30 | Ensure that VIO is only used when appropriate. |

# Sources of DFSORT Performance Information

OS/390 and MVS/ESA systems provide a wealth of performance data which is often difficult to analyze in its raw form. You need to convert this raw data into processed data that can be used for trend analysis and management reports. You can use PMIO to help you get this kind of information. Service Level Reporter (SLR), Enterprise Performance Data Manager (EPDM), and DFSORT/ICETOOL can also provide useful performance information.

## Service Level Reporter

Service Level Reporter (SLR) is a post-processing program for system log data sets. One of its capabilities is to process SMF records written by the system and by products such as DFSORT, NetView, RMF, RACF, and DB2. Additionally, SLR can process any sequential data set, provided you predefine the format.

Data is collected from log data sets into a VSAM relational database, from which reports can be produced, either interactively or with batch jobs. With its Graphical Data Display Manager (GDDM*) interface, SLR enables you to create additional graphical representations of your reports.

**Online**
This consists of an ISPF dialog for both predefined reports and a powerful ad-hoc report generator. The emphasis is on historical, trend-type reporting, but very detailed data (down to the record level) can also be displayed.

**Batch** This is useful for submitting reports that occur on a regular basis. You can use the online facility for report definition, but use batch mode for running the reports.

The ability of SLR to provide both summary and detailed reports makes it a useful tool for initial analysis of DFSORT performance information. Once a bottleneck has been identified, it might be necessary to use other tools, although SLR data is often sufficient.

## Enterprise Performance Data Manager

Enterprise Performance Data Manager (EPDM) is also a post-processing program for system log data sets. One of its capabilities is to process SMF records written by the system and by products such as DFSORT, NetView, RMF, RACF, and DB2. Additionally, EPDM can process any sequential data set, provided you predefine the format.

Data is collected from log data sets into a DB2 relational database, from which reports can be produced, either interactively or with batch jobs. With its Graphical Data Display Manager (GDDM) interface, EPDM enables you to create additional graphical representations of your reports.

**Online**
This consists of an ISPF dialog for both predefined reports and a powerful ad-hoc report generator. The emphasis is on historical, trend-type reporting, but very detailed data (down to the record level) can also be displayed.

**Batch** This is useful for submitting reports that occur on a regular basis. You can use the online facility for report definition, but use batch mode for running the reports.

The ability of EPDM to provide both summary and detailed reports makes it a useful tool for initial analysis of DFSORT performance information. Once a bottleneck has been identified, it might be necessary to use other tools, although SLR data is often sufficient.

## Performance Management for I/O

IBM can provide further tuning of DFSORT through the Performance Management for I/O (PMIO) offering. If you need to reduce your batch application elapsed time or the resources required to run batch, PMIO can provide an IBM Performance

Specialist who conducts a study using your installation's SMF data. This study results in specific tuning recommendations and usually consists of three phases:

**Phase 1**

This phase is focused on developing an overall approach to tuning the batch applications. The available system resources and type of workloads being tuned are examined. Then, an overall approach is devised for tuning the batch applications that are being studied.

For example, a system with plenty of processor storage and CPU available would be a good candidate for Hipersorting and OUTFIL, if the workload could take advantage of them.

**Phase 2**

This phase is focused on application level analysis including:

- Which jobs run at what times?
- How many significant processing steps does each job have such as DFSORT or DB2?
- What dependencies exist between various jobs?
- Which jobs fail and how?

Using this information, a decision is made about which jobs are most worthwhile to tune.

**Phase 3**

During this phase, specific recommendations are provided to speed up or overlap certain applications. For example, a job might be split into two. The first step might include a sort using Hipersorting. The output of the first step might be routed to the second step using SmartBatch Pipes.

In addition to the three phases, PMIO produces system-wide reports on DFSORT. PMIO also provides information about the sort invocation being tuned by:

- Analyzing SMF Type-16 records produced by DFSORT. These give information on the sort, merge or copy operation, including such items as elapsed time, CPU time, bytes sorted, input and output data set information (such as data set block size), and the amount of sort work I/O performed.
- Providing a more complete picture of the job and step in which the DFSORT invocation is being performed. For example, many sorts are program-invoked. Such a program might perform other processing which could be optimized. PMIO "job dossiers" report on step characteristics (such as CPU time and storage usage), job characteristics (such as the number of steps and lines printed), and data set usage (using the "Life Of A Data Set" (LOADS) technique).
- Analyzing SMF Type-42 (subtype 6) records to provide detailed information on data set performance.
- Analyzing SMF Type-91 records to provide information on balancing pipes. DFSORT applications often run faster than their partners, so pipe balancing can be important. For example, if a SORTOUT data set is piped, it may be necessary to clone the readers of that pipe.

Though PMIO reports at the job level, it also provides system-wide analysis of long-running DFSORT invocations. This analysis can be used to prioritize DFSORT applications to be tuned.

## DFSORT/ICETOOL

If you do not have access to SLR, PMIO, or similar information, you can use the DFSORT editing functions in combination with the reporting capabilities of ICETOOL

or OUTFIL to generate performance data reports. The INCLUDE/OMIT, OUTFIL, and INREC/OUTREC control statements are helpful in selecting the particular fields of the particular SMF, RMF, or other data records to be analyzed. ICETOOL or OUTFIL can then be used to generate printable reports from the resulting raw data. See *Application Programming Guide* for more information about ICETOOL, OUTFIL, and editing functions.

## Analysis Techniques for DFSORT Performance Data

After identifying where and how you can obtain the performance data, you need to analyze the performance data. You need to understand your current use of DFSORT in order to assess whether any changes are needed. Tuning might not be necessary.

Investigating DFSORT use does not need to take a lot of effort or a long time. You can investigate DFSORT a number of times, and with each iteration, look at a different aspect or go into a greater level of detail. Where you begin depends on how much time you have and the effort you are prepared to invest.

You can also investigate DFSORT use to ensure that any tuning you have done has had the desired effect. Alternative techniques for investigating current DFSORT use are described in this section and ordered according to how much effort is involved.

Regardless of which technique you choose, you might want to use the ICETOOL DEFAULTS operator to list the ICEMAC installation defaults currently in effect at your site. You can use this operator at any time to produce a list of the values for each ICEMAC parameter for each of the four environment installation modules and each of the four time-of-day installation modules. Refer to “ICEMAC” on page 33 for information on ICEMAC and an example of an ICETOOL defaults job. The output from DEFAULTS can help you analyze how you are using DFSORT and is beneficial to have regardless of the level of analysis you are performing. See *Application Programming Guide* and *Installation and Customization* for more information about the ICETOOL DEFAULTS operator.

## Simple Analysis

Generally, a few DFSORT applications at a site can be identified as being the largest and longest running, or on a critical path. An application is said to be on a critical path if any delay in its completion would delay the finish of the workload. Tuning activities on the critical path can have a beneficial effect in reducing the overall elapsed time of the batch workload. The performance of these applications can determine whether you finish inside or overrun the batch window, or whether you are using too many system resources. The “80/20 rule of thumb” applies to DFSORT: often 80% of the system’s resources used by DFSORT are used by 20% of DFSORT applications.

Start your investigation by concentrating on these applications. Use a SORTDIAG DD statement (or make sure the ICEMAC option DIAGSIM=YES has been specified for your site) to obtain all available messages.

DFSORT messages are a simple and effective source of information for individual applications. They give you a picture of the processing and help you develop an action plan.

Performance-related information in the DFSORT messages includes:

- Amount of virtual storage available

### Performance Data

- DFSORT technique used
- Total bytes and records sorted
- Number and distribution of EXCPs by data set
- Work data set space allocated and used
- Hiperspace or data space used
- Availability of the DFSORT SVC
- Access methods and block sizes used
- Settings of various options (for example, EQUALS, VERIFY) that can affect performance

## Moderate Analysis

If you can take the time to do some analysis of all your DFSORT applications, you can pinpoint more precisely where to tune your use of DFSORT to perform more efficiently. The easiest way to start is to use information optionally provided by DFSORT. To do this, you can use data from SMF records or an ICEIEXIT routine.

### Using SMF Data

Four SMF record types are particularly useful for analyzing the performance of DFSORT: type-30, type-16, type-42 and type-91. To collect an SMF record type, make sure that the active SMFPRMxx member is set up to collect that particular type. For type-30 records, make sure subtype 4 (step total) records are collected. If device connect time and EXCP statistics are desired in the type-30 (subtype 4) records, the DETAIL option must be in effect. For type-42 records, make sure subtype 6 (data set statistics) records are collected. See *MVS Installation and Tuning Reference* for more information on setting up an SMFPRMxx member.

***Type-30 Records:*** SMF type-30 (subtype 4) records contain useful information on the resource consumption of a particular job step. These include:

- CPU time (total, and broken down by field)
- Elapsed time
- EXCPs (total, and broken down by device)
- Device connect time, broken down by device
- Paging statistics

See *MVS System Management Facilities* for more detailed information on the type-30 fields.

***Type-16 Records:*** DFSORT produces SMF type-16 records. They contain data on a particular DFSORT step. If you want to produce DFSORT SMF records, DFSORT's supplied ICEMAC default of SMF=NO must be changed to SMF=FULL or SMF=SHORT, or SMF=FULL or SMF=SHORT must be specified on an OPTION statement in DFSPARM or in the extended parameter list at run-time. This must be done in addition to setting up the appropriate SMFPRMxx member. The DFSORT SVC must be available for DFSORT to be able to write the SMF records to an SMF data set. Refer to "DFSORT SVC" on page 26 for additional information.

Some of the fields in the type-16 record containing performance-related data are:

- Type of application (sort, merge, or copy)
- DFSORT technique used
- Elapsed, TCB, and SRB time used
- Type and length of records being sorted
- Total bytes and records sorted
- Work data set allocation data

- Access methods and block sizes used
- Control field length
- Amount of Hiperspace and data space used
- Number of input and output data set I/O calls
- Number of work data set EXCPs

See *Installation and Customization* for a complete list of the type-16 fields.

There are several reasons why it might be convenient to use the DFSORT type-16 records:

- You already have procedures to analyze this information.
- The scope of the information you are interested in is limited to the information available in the SMF records.
- You do not wish to write and maintain an analysis routine.

***Type-42 Records:*** DFSMS/MVS produces SMF type-42 (subtype 6) records. They contain information on a particular DASD data set OPEN for a particular job, such as:

- Start Subchannels (SCCHs) (physical I/Os)
- Response time components (Connect, Disconnect, IOSQ and Pending times)
- Cache performance information

Both interval and CLOSE-time records are produced (unless interval recording is disabled).

See *MVS System Management Facilities* for more detailed information on the type-42 subtype 6 fields.

***Type-91 Records:*** SmartBatch for OS/390 produces SMF type-91 records. They contain information on the use of a particular SmartBatch pipe, such as:

- The jobs that use the pipe
- The number of blocks of data transferred through the pipe
- The number of waits on the pipe for empty or full

This information can help you balance pipes between DFSORT applications and other applications.

See *SmartBatch/MVS* for more information on the fields in the SMF type-91 record.

## Using ICEIEXIT Data

DFSORT enables you to provide an installation initialization exit (ICEIEXIT) routine which can be used to examine some performance-related information about all DFSORT applications at a site.

If present and activated, the ICEIEXIT routine is passed installation and run-time information. As this book illustrates in later sections, the values of installation and run-time parameters affect DFSORT performance.

The ICEIEXIT routine can examine installation and run-time information related to:

- Virtual storage limits
- Hiperspace limits
- Data space limits
- Use of VERIFY
- OUTFIL buffer space limits

as well as additional run-time information on:
- Type of DFSORT application (sort, merge, or copy)
- Method of invocation
- Use of the Blockset technique
- Use of storage above 16MB virtual

See *Installation and Customization* for complete information on how to write and install an ICEIEXIT routine.

# Thorough Analysis

To tune DFSORT thoroughly, you need extensive data about all DFSORT applications run by you or your site, including elapsed time, CPU time, number of EXCPs, and types and amounts of virtual storage used. JES logs and DFSORT messages can be used to analyze an isolated number of applications, but this is not practical to apply to a large number of DFSORT jobs. In addition, these sources do not supply all of the data needed to perform such a thorough analysis of DFSORT resource consumption.

## Using ICETEXIT Data

DFSORT enables you to provide an installation termination exit (ICETEXIT) routine which can be used to collect extensive performance-related information about all DFSORT applications at a site.

If present and activated, the ICETEXIT routine is called at the end of DFSORT application processing. It is available for those who wish to make a thorough analysis of DFSORT performance data using a single source of information. See *Installation and Customization* for complete information on how to write and install an ICETEXIT routine.

ICETEXIT provides comprehensive information on each DFSORT application, including the information contained in DFSORT's type-16 SMF record.

**Note:** The SMF record is available to the ICETEXIT routine even if the SMF facility is not being used (in this case, the SMF record is constructed and passed to the ICETEXIT routine, but not written to the SMF data set).

Additional information available to the ICETEXIT routine includes:
- Control statements specified
- User exits used
- Options (for example, VERIFY) in effect
- Control field formats used
- Work data set EXCPs broken down by data set and DFSORT phase
- Use of cache fast write
- Virtual storage statistics
- DFSORT phase timing statistics (elapsed, TCB, SRB)
- Input and output data set statistics
- Hipersorting statistics
- Sorting with data space statistics

In some cases, you only need a portion of this information for performance and tuning reasons.

The advantage of using an ICETEXIT routine is that all the information about each DFSORT application is available to the routine. You do not need to use information from a number of other sources. How you process the data depends on your requirements.

Examples of how you can use an ICETEXIT routine include:

- Collating information from SMF type-16 records with run-time and installation option values to identify applications with options which degrade performance.
- Write an SMF record using your own format.

You can write the record to a private data set for subsequent processing.

You can process SMF records or private data set records using a data reduction program such as SLR. Refer to “Service Level Reporter” on page 75 for a description.

The example in “Appendix A. Sample ICETEXIT” on page 85 provides a sample ICETEXIT assembler routine. This installation-wide exit creates a user SMF record from the data passed by DFSORT.

The assembler source for an SVC which writes the user SMF record to SMF is also included. This enables the ICETEXIT routine to run unauthorized but write SMF records using the SVC.

See *Installation and Customization* for more information on how to write and install an ICETEXIT routine,

## Using RMF Data

Using ICETEXIT data enables you to understand the details of how DFSORT is being used at your site. But in addition to optimizing the performance of individual or groups of sort applications, it is also very important to pay some attention to the system’s overall performance when making tuning decisions for DFSORT.

Using SMF type-30 records and certain RMF reports, such as the I/O device activity, the paging activity, and the address space resource data reports, enables you to measure the amount of impact (if any) on system resources that is caused by DFSORT applications. With this information it is possible to balance DFSORT’s resource requirements with those of the other applications on the system.

See *Analyzing RMF Reports* for more information about RMF reports.

# DFSORT Requirements and System Resources

The previous sections described the significance of understanding your site’s current use of DFSORT, in making the proper DFSORT tuning decisions. But in addition to optimizing the performance of individual or groups of sort applications, it is also very important to pay some attention to the system’s overall performance when making tuning decisions for DFSORT (or tuning decisions for any application, for that matter). System resources are always limited, and a heavy overcommitment of resources usually means a severe performance degradation for the entire system. So it is advisable to balance DFSORT’s resource requirements with those of the other applications on the system.

Analyzing system performance is usually accomplished with the Resource Measurement Facility (RMF) or an equivalent product. Many different types of reports can be generated with RMF, such as CPU Activity Reports, I/O Activity Reports, and Paging Activity Reports. A lot of very useful information is contained in these reports, but for the purpose of this discussion on DFSORT tuning, only the

### Performance Data

I/O and Paging Activity Reports will be addressed. For detailed information on system-wide tuning in general, see the *SPL: Initialization and Tuning* and *Resource Measurement Facility User's Guide*

## Placement of Data Sets

The RMF I/O Device Activity Report provides device related information such as average response times, average connect and disconnect times, and average activity rates. This data is furnished for each online I/O device in one or more device classes. The information in the I/O Activity Report (in conjunction with the Channel Activity and I/O Queueing Activity reports) can help you to identify possible areas of device contention. For instance, if the report shows a high device activity rate and an unusually large pending time for a particular device (or group of devices) containing DFSORT work data sets, the performance of the sort application was most certainly adversely affected. If it turns out that the volumes selected for the work data sets happened to, for example, also contain the tables for a very active data base, it might be advantageous to make changes to the applications or to the I/O configuration that would prevent future allocation of DFSORT work data sets to those particular devices. The same idea holds true for DFSORT input and output data sets, and in general for any application on the system; if there is a lot of contention for a few I/O devices, the data on those devices should be distributed over more (less active) volumes to achieve an overall balance in I/O activity rates. Keep in mind that for best DFSORT performance, the input and work data sets, as well as the work and output data sets should be located on different volumes, or even different storage subsystems if possible.

## Use of Virtual Storage

The RMF Paging Activity Report contains two fields that can be especially useful in determining contention for central storage resources. The high unreferenced interval count (HUIC) is shown in the expanded storage movement rates section, and the minimum total available central storage frames is shown in the frame and slot counts section. If the Paging Activity Report indicates a very low HUIC and an unusually low available frame count for extended periods of time, then the system is suffering from contention for central storage.

You can use SMF type-30 records to pinpoint which jobs or set of jobs are using the most virtual storage. If these are not DFSORT applications, you may be able to reduce the REGION parameters in the JCL for those jobs, or restrict the available region sizes and limits for those jobs in the IEFUSI system exit. If they are DFSORT jobs, there are two possible actions:

1. Restrict the virtual storage available to DFSORT. This can be done by setting lower TMAXLIM/MAXLIM/DSA installation defaults, or by dynamically restricting the storage size in an ICEIEXIT, or by a combination of the two.
2. Restrict DFSORT's use of dataspace sorting. This can be accomplished by setting a DSPSIZE=n value as installation default where n is the dataspace size limitation for every sort job. The appropriate value to use for n is best determined by experimentation on the system in question. A good technique to use would be to start out with a very low value and then keep raising it by 10MB or 20MB for as long as system performance is not affected significantly. In cases of severe overcommitment of central storage, a DSPSIZE=0 default may be advisable.

However, the possible performance consequences for the DFSORT applications should be taken into account. Refer to "Virtual Storage and Sorting with Data

Space" on page 49 for more information. Performance critical DFSORT jobs may have to be excluded from these storage restrictions.

## Use of Expanded Storage

Similar to the central storage frame counts, the minimum total available expanded storage frame counts can be used to gauge the amount of contention for expanded storage resources. The expanded storage frame counts are also contained in the frame and slot count section of the RMF Paging Activity Report. If the report indicates an unusually low available frame count for extended periods of time, then the system is suffering from a contention for expanded storage.

## Use of VIO Data Sets

Another useful measurement contained in the RMF Paging Activity Report is the total system VIO page-in and page-out rates, which are displayed in the central storage paging rates section. Unusually high VIO paging rates can indicate an overuse of VIO data sets. With the exception of sorting applications that process only very small files, it is generally undesirable to allow the use of VIO devices for DFSORT output and especially DFSORT work data sets. Refer to "System-Managed Storage" on page 21 for details on how to prevent VIO allocations for DFSORT data sets when using DFSMS.

# Performance Trade-Offs

Table 6 is a summary of the techniques described in this book for improving DFSORT performance. In many cases (use of Blockset is a notable exception), an improvement in performance is not free. The figure summarizes these potential trade-offs: increased paging, increased swapping, increased CPU time, and changes needed to applications.

For example, to use this figure, assume you want to improve the elapsed time for a DFSORT application. A number of techniques exist to choose from, ranging from ensuring that DFSORT uses Blockset to modifying the way the application uses DFSORT. You can decide which technique, or combination of techniques, is appropriate based on the effort required and the trade-off, if any, you are prepared to accept.

*Table 6. Summary of Potential Performance Trade-Offs*

| | | Potential Trade-Offs | | | |
|---|---|---|---|---|---|
| **Improvement Area** | **Technique** | **Increased Paging** | **Increased Swapping** | **Increased CPU Time** | **Change Application** |
| CPU Time | Use Blockset | | | | |
| | Use dataspace sorting | | | | |
| | Use 3990 storage control for input data sets [1] | | | | |
| | Increase Virtual Storage | Y | Y | | |
| | Use appropriate large block sizes for input and output data sets | | | | |
| | Use DFSORT functions and exits | | | | Y |

*Table 6. Summary of Potential Performance Trade-Offs (continued)*

| Improvement Area | Technique | Potential Trade-Offs | | | |
|---|---|---|---|---|---|
| | | Increased Paging | Increased Swapping | Increased CPU Time | Change Application |
| Elapsed Time | Use Blockset | | | | |
| | Use dataspace sorting | | | | |
| | Use multiple work data set devices | | | | |
| | Use cached 3990 storage control for work data sets [1] | | | | |
| | Increase Virtual Storage | Y | Y | | |
| | Use Hipersorting | | | Y | |
| | Use VIO data sets | Y | | Y | |
| | Use appropriate large block sizes for input and output data sets | | | | |
| | Use DFSORT functions and exits | | | | Y |
| | Use SmartBatch Pipes | | | Y | |
| | Use striping | | | | |
| I/O Activity | Use Blockset | | | | |
| | Increase Virtual Storage | Y | Y | | |
| | Use Hipersorting | | | Y | |
| | Use dataspace sorting | | | | |
| | Use appropriate large block sizes for input and output data sets | | | | |
| | Use DFSORT functions and exits | | | | Y |
| | Use SmartBatch Pipes | | | Y | |
| | Use striping | | | | |
| Work Data Set Space | Use Hipersorting | | | Y | |
| | Use dataspace sorting | | | | |
| | Increase Virtual Storage | Y | Y | | |
| | Specify number of records when appropriate | | | | Y |

1. Requires the DFSORT SVC.

# Appendix A. Sample ICETEXIT

This appendix contains Programming Interface information.

This appendix contains a sample ICETEXIT routine which creates a composite record containing all of the data areas passed to ICETEXIT and subsequently writes it out to SMF as user record 129. When ICEMAC option TEXIT=YES is specified, DFSORT calls the ICETEXIT routine at the end of each DFSORT application run at the site.

Because this routine does not run as an authorized program, an SVC is required to write the record to SMF. (This user SVC is unrelated to the SVC used by DFSORT itself.) DFSORT SVC shows a sample SVC that can be used for this purpose. However, before installing it as a user SVC at your site, you should consider the security implications of using the SVC as is.

**Note:** The use of this ICETEXIT routine and associated SVC does not replace the DFSORT capability to issue its own type-16 SMF record. If you want DFSORT to issue the type-16 SMF record, use ICEMAC or run-time option SMF=SHORT or SMF=FULL. Depending upon how you install the DFSORT SVC, you may have to change ICEMAC option SVC=n. See *Installation and Customization* for details.

| You can obtain the sample ICETEXIT and SVC shown in this appendix from the
| DFSORT FTP site. The file is TGSAMP.ZIP.

```
ICETEXIT TITLE 'SAMPLE ICETEXIT ROUTINE'
*-------------------------------------------------------------------*
*       SAMPLE ICETEXIT ROUTINE.                                    *
*       PURPOSE:                                                    *
*           THIS ROUTINE WILL COMBINE ALL OF THE DATA AREAS PASSED  *
*           TO ICETEXIT IN ONE USER SMF RECORD ID 129.  THESE       *
*           RECORDS CAN THEN BE ANALYZED AS A GROUP.                *
*                                                                   *
* NOTES:                                                            *
*   1) SEE "DFSORT INSTALLATION AND CUSTOMIZATION" FOR FULL DETAILS *
*      ON ICETEXIT                                                  *
*   2) THIS EXAMPLE ASSUMES THE USE OF SVC 249 - CHANGE IT AS       *
*      APPROPRIATE IF YOU USE A DIFFERENT SVC                       *
*                                                                   *
* REQUIRED DFSORT MACROS: ICESMF, ICEDTEX                           *
* REQUIRED SYSTEM MACROS: SAVE, GETMAIN, TIME, FREEMAIN, RETURN,    *
*                         CVT, IEESMCA, YREGS                       *
*-------------------------------------------------------------------*
         SPACE 1
ICETEXIT CSECT
ICETEXIT AMODE 24
ICETEXIT RMODE 24
         SPACE 1
```

*Figure 17. Sample ICETEXIT (Part 1 of 9)*

## Sample ICETEXIT

```
*--------------------------------------------------------------------*
*        INITIALIZATION.                                             *
*--------------------------------------------------------------------*
         SPACE 1
         SAVE  (14,12),,ICETEXIT-&SYSDATE;-&SYSTIME;
         LR    R12,R15             EPA FOR BASE
         USING ICETEXIT,R12
         LR    R11,R1              SAVE ICETPAR ADDR
         USING ICETPAR,R11
         LA    R0,DSALTH           LENGTH OF DSA TO GETMAIN
       GETMAIN RC,LV=(0)
         LTR   R15,R15             DID WE GET STORAGE?
         BNZ   S01L990                NO..THEN GET OUT
         ST    R1,8(,R13)          FORWARD CHAIN
         ST    R13,4(,R1)          BACKWARD CHAIN
         LR    R13,R1              ->NEW SAVE AREA
         USING DSA,R13
         SPACE 1
*--------------------------------------------------------------------*
*        COMPUTE SIZE OF USER SMF RECORD TO BE BUILT.                *
*--------------------------------------------------------------------*
         SPACE 1
         L     R0,SDASMFL          LENGTH OF CONSTANT PORTIONS
         ICM   R1,B'1111',ICETSMFA  ->SORT SMF RECORD
         BZ    S01L100                BRANCH IF NOT ONE..
         USING ICESMFH,R1
         AH    R0,ICERDW           + LENGTH
         DROP  R1
         SPACE 1
S01L100  DS    0H
         ST    R0,DSASMFL          SAVE FOR LATER FREEMAIN
       GETMAIN RC,LV=(0)
         LTR   R15,R15             DID WE GET THE STORAGE?
         BNZ   S01L950               NO..THEN EXIT
         LR    R8,R1               LOAD FOR BASE
         USING ICE81RCD,R8
         SPACE 1
         LR    R0,R1               TO ADDR
         L     R1,DSASMFL          TO LENGTH
         SR    R14,R14             NO FROM ADDR
         LR    R15,R14             NO FROM LENGTH
         MVCL  R0,R14              ZERO THE AREA
         SPACE 1
```

*Figure 17. Sample ICETEXIT (Part 2 of 9)*

```
*--------------------------------------------------------------------*
*        NOW BUILD THE SMF HEADER.                                   *
*--------------------------------------------------------------------*
         SPACE 1
         MVI   ICE81RTY,129        SET RECORD TYPE
         TIME  BIN
         STM   R0,R1,ICE81TME      SAVE TIME/DATE
         L     R1,16               ->CVT
         USING CVT,R1
         L     R1,CVTSMCA          ->SMCA
         USING SMCABASE,R1
         MVC   ICE81SID,SMCASID    GET SYSID
         DROP  R1
         LA    R9,ICE81LTH         LENGTH OF HEADER
         LA    R10,0(R9,R8)        ->FIRST AVAILABLE POSITION
         SPACE 1
*--------------------------------------------------------------------*
*        APPEND THE DFSORT SMF RECORD.                               *
*--------------------------------------------------------------------*
         SPACE 1
         ICM   R2,B'1111',ICETSMFA    -> DFSORT SMF RECORD
         BZ    S01L110                BRANCH IF NONE
         SPACE 1
         STH   R9,ICE81ROF         SAVE OFFSET
         USING ICESMFH,R2
         LH    R1,ICERDW           LENGTH OF SMF RECORD
         LR    R0,R10              TO ADDRESS
         AR    R9,R1               INCREMENT LENGTH/OFFSET
         AR    R10,R1              ->AFTER SMF RECORD
         LR    R3,R1               FROM LENGTH
         MVCL  R0,R2               APPEND DFSORT SMF RECORD
         DROP  R2
         SPACE 1
*--------------------------------------------------------------------*
*        APPEND THE SMF STATISTICS.                                  *
*--------------------------------------------------------------------*
         SPACE 1
S01L110  DS    0H
         ICM   R2,B'1111',ICETSST    -> SMF STATISTICS
         BZ    S01L120               BRANCH IF NONE
         STH   R9,ICE81SOF         SAVE OFFSET
         USING ICESST,R2
         L     R1,ICESMFST         LENGTH OF SMF STATISTICS
         LR    R0,R10              TO ADDRESS
         AR    R9,R1               INCREMENT LENGTH/OFFSET
         AR    R10,R1              ->AFTER SMF RECORD
         LR    R3,R1               FROM LENGTH
         MVCL  R0,R2               APPEND DFSORT SMF RECORD
         DROP  R2
         SPACE 1
```

*Figure 17. Sample ICETEXIT (Part 3 of 9)*

## Sample ICETEXIT

```
*-------------------------------------------------------------------*
*        APPEND THE GENERAL STATISTICS.                             *
*-------------------------------------------------------------------*
         SPACE 1
S01L120  DS    0H
         ICM   R2,B'1111',ICETGEN    -> GEN STATISTICS
         BZ    S01L130               BRANCH IF NONE
         STH   R9,ICE81GOF         SAVE OFFSET
         USING ICEGEN,R2
         L     R1,ICEGSTAT         LENGTH OF GEN STATISTICS
         LR    R0,R10              TO ADDRESS
         AR    R9,R1               INCREMENT LENGTH/OFFSET
         AR    R10,R1              ->AFTER SMF RECORD
         LR    R3,R1               FROM LENGTH
         MVCL  R0,R2               APPEND DFSORT SMF RECORD
         DROP  R2
         SPACE 1
*-------------------------------------------------------------------*
*        APPEND THE OPTIONS STATISTICS.                             *
*-------------------------------------------------------------------*
         SPACE 1
S01L130  DS    0H
         ICM   R2,B'1111',ICETOPTS   -> OPTIONS STATISTICS
         BZ    S01L140               BRANCH IF NONE
         STH   R9,ICE81OOF         SAVE OFFSET
         USING ICEOPTS,R2
         L     R1,ICEOSTAT         LENGTH OF OPTIONS STATISTICS
         LR    R0,R10              TO ADDRESS
         AR    R9,R1               INCREMENT LENGTH/OFFSET
         AR    R10,R1              ->AFTER SMF RECORD
         LR    R3,R1               FROM LENGTH
         MVCL  R0,R2               APPEND DFSORT SMF RECORD
         DROP  R2
         SPACE 1
```

*Figure 17. Sample ICETEXIT (Part 4 of 9)*

```
*--------------------------------------------------------------------*
*        APPEND THE SORT/MERGE STATISTICS.                           *
*--------------------------------------------------------------------*
         SPACE 1
S01L140  DS    0H
         ICM   R2,B'1111',ICETSMS    -> SORT/MERGE STATISTICS
         BZ    S01L150               BRANCH IF NONE
         STH   R9,ICE81MOF         SAVE OFFSET
         USING ICESMS,R2
         L     R1,ICEFSTAT         LENGTH OF SORT/MERGE STATISTICS
         LR    R0,R10              TO ADDRESS
         AR    R9,R1               INCREMENT LENGTH/OFFSET
         AR    R10,R1              ->AFTER SMF RECORD
         LR    R3,R1               FROM LENGTH
         MVCL  R0,R2               APPEND DFSORT SMF RECORD
         DROP  R2
         SPACE 1
*--------------------------------------------------------------------*
*        APPEND THE VIRTUAL STORAGE STATISTICS.                      *
*--------------------------------------------------------------------*
         SPACE 1
S01L150  DS    0H
         ICM   R2,B'1111',ICETVIRT   -> VIRTUAL STORAGE STATISTICS
         BZ    S01L160               BRANCH IF NONE
         STH   R9,ICE81VOF         SAVE OFFSET
         USING ICEVSTOR,R2
         L     R1,ICEVSTAT         LENGTH OF VIRTUAL STORAGE STATISTICS
         LR    R0,R10              TO ADDRESS
         AR    R9,R1               INCREMENT LENGTH/OFFSET
         AR    R10,R1              ->AFTER SMF RECORD
         LR    R3,R1               FROM LENGTH
         MVCL  R0,R2               APPEND DFSORT SMF RECORD
         DROP  R2
         SPACE 1
*--------------------------------------------------------------------*
*        APPEND THE PHASE TIMING STATISTICS.                         *
*--------------------------------------------------------------------*
         SPACE 1
S01L160  DS    0H
         ICM   R2,B'1111',ICETPTIM   -> PHASE TIMING STATISTICS
         BZ    S01L170               BRANCH IF NONE
         STH   R9,ICE81TOF         SAVE OFFSET
         USING ICEPHAST,R2
         L     R1,ICEPTIME         LENGTH OF PHASE TIMING STATISTICS
         LR    R0,R10              TO ADDRESS
         AR    R9,R1               INCREMENT LENGTH/OFFSET
         AR    R10,R1              ->AFTER SMF RECORD
         LR    R3,R1               FROM LENGTH
         MVCL  R0,R2               APPEND DFSORT SMF RECORD
         DROP  R2
         SPACE 1
```

*Figure 17. Sample ICETEXIT (Part 5 of 9)*

## Sample ICETEXIT

```
*-------------------------------------------------------------------*
*        APPEND THE SORTIN STATISTICS.                              *
*-------------------------------------------------------------------*
         SPACE 1
S01L170  DS    0H
         ICM   R2,B'1111',ICETSIN    -> SORTIN STATISTICS
         BZ    S01L180               BRANCH IF NONE
         STH   R9,ICE81XOF         SAVE OFFSET
         USING ICESRTIN,R2
         L     R1,ICESORTI         LENGTH OF SORTIN STATISTICS
         LR    R0,R10              TO ADDRESS
         AR    R9,R1               INCREMENT LENGTH/OFFSET
         AR    R10,R1              ->AFTER SMF RECORD
         LR    R3,R1               FROM LENGTH
         MVCL  R0,R2               APPEND DFSORT SMF RECORD
         DROP  R2
         SPACE 1
*-------------------------------------------------------------------*
*        APPEND THE SORTOUT STATISTICS.                             *
*-------------------------------------------------------------------*
         SPACE 1
S01L180  DS    0H
         ICM   R2,B'1111',ICETSOUT   -> SORTOUT STATISTICS
         BZ    S01L190               BRANCH IF NONE
         STH   R9,ICE81ZOF         SAVE OFFSET
         USING ICESRTOT,R2
         L     R1,ICESORTO         LENGTH OF SORTOUT STATISTICS
         LR    R0,R10              TO ADDRESS
         AR    R9,R1               INCREMENT LENGTH/OFFSET
         AR    R10,R1              ->AFTER SMF RECORD
         LR    R3,R1               FROM LENGTH
         MVCL  R0,R2               APPEND DFSORT SMF RECORD
         DROP  R2
         SPACE 1
*-------------------------------------------------------------------*
*        APPEND THE SORTWK STATISTICS.                              *
*-------------------------------------------------------------------*
         SPACE 1
S01L190  DS    0H
         ICM   R15,B'1111',ICETSWK    -> SORTWK STATISTICS
         BZ    S01L200               BRANCH IF NONE
         STH   R9,ICE81WOF         SAVE OFFSET
         USING ICEWRKDS,R15
         L     R1,ICESORTW         LENGTH OF SORTWK STATISTICS
         LR    R0,R10              TO ADDRESS
         AR    R9,R1               INCREMENT LENGTH/OFFSET
         AR    R10,R1              ->AFTER SMF RECORD
         LR    R2,R15              FROM ADDRESS
         LR    R3,R1               FROM LENGTH
         MVCL  R0,R2               APPEND DFSORT SMF RECORD
         SPACE 1
```

*Figure 17. Sample ICETEXIT (Part 6 of 9)*

```
*--------------------------------------------------------------------*
*        APPEND THE SORTWK DATA SET ENTRIES.                         *
*--------------------------------------------------------------------*
         SPACE 1
         LH    R7,ICESWKEN         # SORTWK DATA SET ENTRIES
         STH   R7,ICE81EEN         SAVE IN SMF RECORD
         STH   R9,ICE81EOF         SAVE OFFSET
         LA    R15,ICESWK01        ->FIRST IN LIST
         DROP  R15
S01L195  DS    0H
         L     R2,0(,R15)          ->ICEWORK
         USING ICEWORK,R2
         L     R1,ICESWORK         LENGTH OF SORTWK DATA SET ENTRY
         LR    R0,R10              TO ADDRESS
         AR    R9,R1               INCREMENT LENGTH/OFFSET
         AR    R10,R1              ->AFTER SMF RECORD
         LR    R3,R1               FROM LENGTH
         MVCL  R0,R2               APPEND DFSORT SMF RECORD
         SPACE 1
         LA    R15,4(,R15)         ->NEXT IN LIST
         BCT   R7,S01L195          DO ALL IN LIST
         SPACE 1
*--------------------------------------------------------------------*
*        APPEND THE HIPERSORTING STATISTICS.                         *
*--------------------------------------------------------------------*
         SPACE 1
S01L200  DS    0H
         ICM   R2,B'1111',ICETHIPR   -> HIPERSORTING STATISTICS
         BZ    S01L210               BRANCH IF NONE
         STH   R9,ICE81HOF         SAVE OFFSET
         USING ICEHIPER,R2
         L     R1,ICEHSORT         LENGTH OF HIPERSORTING STATISTICS
         LR    R0,R10              TO ADDRESS
         AR    R9,R1               INCREMENT LENGTH/OFFSET
         AR    R10,R1              ->AFTER SMF RECORD
         LR    R3,R1               FROM LENGTH
         MVCL  R0,R2               APPEND DFSORT SMF RECORD
         DROP  R2
         SPACE 1
```

*Figure 17. Sample ICETEXIT (Part 7 of 9)*

## Sample ICETEXIT

```
*-------------------------------------------------------------------*
*        APPEND THE SORTING WITH DATA SPACE STATISTICS               *
*-------------------------------------------------------------------*
         SPACE 1
S01L210  DS    0H
         ICM   R2,B'1111',ICETDATA   -> SORTING WITH DATA SPACE STATS
         BZ    S01L220               BRANCH IF NONE
         STH   R9,ICE81DOF         SAVE OFFSET
         USING ICEDATAS,R2
         L     R1,ICEDSORT         LENGTH OF DATA SPACE STATISTICS
         LR    R0,R10              TO ADDRESS
         AR    R9,R1               INCREMENT LENGTH/OFFSET
         AR    R10,R1              ->AFTER SMF RECORD
         LR    R3,R1               FROM LENGTH
         MVCL  R0,R2               APPEND DFSORT DATA SPACE STATS
         DROP  R2
         SPACE 1
*-------------------------------------------------------------------*
*        SMF RECORD IS BUILT.  SEND IT OFF TO SMF.                  *
*-------------------------------------------------------------------*
         SPACE 1
S01L220  DS    0H
         STH   R9,ICE81LEN         SAVE IN RDW
         BAL   R15,S01L700         BRANCH AROUND LABEL
         DC    C'USMF'             SVC PASSWORD
S01L700  DS    0H
         L     R0,0(,R15)          LOAD INTO PARM REGISTER
         LR    R1,R8               -> USER SMF RECORD
*******************************************************************
* SVC 249 IS USED IN THIS EXAMPLE - CHANGE THIS CALL AS APPROPRIATE
*******************************************************************
         SVC   249                 CALL SVC TO WRITE RECORD
         SPACE 1
*-------------------------------------------------------------------*
*        ALL DONE.  FREE STORAGE AND RETURN.                        *
*-------------------------------------------------------------------*
         SPACE 1
         LR    R1,R8               -> SMF RECORD
         L     R0,DSASMFL          LENGTH OF AREA
      FREEMAIN R,LV=(0),A=(1)
         SPACE 1
S01L950  DS    0H
         LA    R0,DSALTH           LENGTH OF AREA
         LR    R1,R13              ->AREA
         L     R13,4(,R1)          ->CALLER'S SAVEAREA
      FREEMAIN R,LV=(0),A=(1)
         SPACE 1
S01L990  DS    0H
        RETURN (14,12),,RC=0
         TITLE 'S D A - STATIC STORAGE AREA (CONSTANTS)'
         SPACE 1
```

*Figure 17. Sample ICETEXIT (Part 8 of 9)*

```
*--------CONSTANT DEFINING THE COMBINED LENGTHS OF ALL DATA AREAS
*        EXCEPT FOR THE DFSORT SMF RECORD (ICESMF)
         SPACE 1
SDASMFL  DC    A(ICE81LTH+L'ICESMEND+L'ICEGNEND+L'ICEOEND+L'ICESSEND+L'X
               ICEVEND+L'ICETMEND+L'ICESNEND+L'ICESOEND+L'ICESWEND+(32*X
               L'ICESEEND)+L'ICEHREND+L'ICEDSEND)
         TITLE 'D S A - DYNAMIC STORAGE AREA'
DSA      DSECT
DSASAVE  DS    18A                 SAVE AREA
DSASMFL  DS    F                   LENGTH OF USER SMF RECORD AREA
DSALTH   EQU   (*-DSA)             LENGTH OF DSA
         TITLE 'DFSORT USER SMF RECORD GENERATED FROM ICETEXIT ROUTINE'
         SPACE 1
ICE81RCD DSECT
ICE81LEN DS    H                   RECORD LENGTH
ICE81SEG DS    H                   SEGMENT DESCRIPTOR
ICE81FLG DS    XL1                 SYSTEM INDICATOR
ICE81RTY DS    AL1                 RECORD TYPE = AL1(129)
ICE81TME DS    XL4                 TIME, IN HUNDREDTHS OF A SECOND
ICE81DTE DS    PL4                 DATE, IN FORM 00YYDDDF
ICE81SID DS    CL4                 SYSTEM IDENTIFICATION
         SPACE 1
ICE81ROF DS    H                   OFFSET TO SORT SMF RECORD (ICESMF  )
ICE81SOF DS    H                   OFFSET TO SMF STATS       (ICESST  )
ICE81GOF DS    H                   OFFSET TO GENERAL STATS   (ICEGEN  )
ICE81OOF DS    H                   OFFSET TO OPTION STATS    (ICEOPTS )
ICE81MOF DS    H                   OFFSET TO SORT/MERGE STAT (ICESMS  )
ICE81VOF DS    H                   OFFSET TO VIRT STOR. STAT (ICEVSTOR)
ICE81TOF DS    H                   OFFSET TO TIMING STATS    (ICEPHAST)
ICE81XOF DS    H                   OFFSET TO SORTIN STATS    (ICESRTIN)
ICE81ZOF DS    H                   OFFSET TO SORTOUT STATS   (ICESRTOT)
ICE81WOF DS    H                   OFFSET TO SORTWK STATS    (ICEWRKDS)
ICE81EOF DS    H                   OFFSET TO SORTWK ENTRY    (ICEWORK )
ICE81EEN DS    H                   # OF ICEWORK ENTRIES
ICE81HOF DS    H                   OFFSET TO HIPER STATS     (ICEHIPER)
ICE81DOF DS    H                   OFFSET TO DATASPACE STATS (ICEDATAS)
ICE81LTH EQU   *-ICE81RCD          LENGTH OF COMMON PORTION
        ICESMF
        ICEDTEX
         PRINT NOGEN
         CVT   LIST=NO,DSECT=YES
       IEESMCA
         YREGS
         END
```

*Figure 17. Sample ICETEXIT (Part 9 of 9)*

```
IGC0024I TITLE 'SAMPLE SVC 249 - WRITE A USER SMF RECORD'
*--------------------------------------------------------------------*
*        SAMPLE SVC 249 - WRITE A USER SMF RECORD                    *
*        PURPOSE:                                                    *
*                 THIS SVC IS INVOKED BY THE SAMPLE ICETEXIT         *
*                 ROUTINE TO WRITE THE USER SMF RECORD WITHOUT       *
*                 BEING APF AUTHORIZED.                              *
*                                                                    *
* NOTES:                                                             *
*   1) THIS EXAMPLE ASSUMES THE USE OF SVC 249 - CHANGE IT AS        *
*      APPROPRIATE IF YOU USE A DIFFERENT SVC.                       *
*   2) THIS EXAMPLE USES A SIMPLE PASSWORD CHECK TO VERIFY THAT IT   *
*      IS BEING CALLED BY ICETEXIT.  SINCE THIS PASSWORD SCHEME COULD *
*      EASILY BE COMPROMISED, YOU SHOULD CAREFULLY EVALUATE THE      *
*      RISKS IN USING IT AS IS IF YOU INSTALL IT IN YOUR SYSTEM.     *
*                                                                    *
* REQUIRED SYSTEM MACROS: SMFWTM, YREGS                              *
*--------------------------------------------------------------------*
         SPACE 1
IGC0024I CSECT
         USING IGC0024I,R6
         C     R0,PASSWORD          IS THE PASSWORD CORRECT?
         BNER  R14                     NO..RETURN TO CALLER
         SMFWTM (1)                 WRITE THE SMF RECORD
         BR    R14                  RETURN TO CALLER
         SPACE 1
         DS    0F
PASSWORD DC    CL4'USMF'
         YREGS
         END
```

*Figure 18. Sample SVC 249 to Write a SMF User Record*

# Appendix B. Estimating Elapsed Time

Before you try to improve the elapsed time for a particular sort, it is useful to calculate an estimate of the best possible time you could achieve as a basis for comparison.

The following is an example of how to calculate the minimum elapsed time needed for a fixed-length record sort based on the following assumptions:

- Work data sets are not used.
- There are 2868536 records and each record is 568 bytes in length.
- The input block size is 27832 bytes resulting in 2 blocks per track.
- 3390 devices are used for input and output.
- A 3390 device has 15 tracks per cylinder.
- Reading a 3390 device at a sustained rate takes 16 disk revolutions per cylinder and 14.1 msec per revolution.

To calculate the elapsed time, complete the following steps:

1. Multiply the number of records by the record length to get the number of bytes in the input data set:

   ```
   2868536 x 568 = 1629328448 bytes in the input data set
   ```

2. Divide the bytes in the input data set by the input block size to get the number of blocks in the input data set (round up to the next whole block):

   ```
   1629328448
   ----------   =   58542 blocks in the input data set
      27832
   ```

3. Divide the number of blocks in the input data set by the number of blocks per track to get the number of tracks (round up to the next whole track):

   ```
   58542
   ------  =   29271 tracks
     2
   ```

4. Divide the number of tracks by the number of tracks per cylinder to get the number of cylinders required to store the input (round up to the next whole cylinder):

   ```
   29271
   ------  =   1951 cylinders required to store input data
     15
   ```

5. Multiply the number of cylinders by the disk revolutions per cylinder to get the total number of disk revolutions. Note that the number of disk revolutions per cylinder (16) is one more than the number of tracks per cylinder (15) because it takes an additional revolution to switch to the next cylinder:

   ```
   1951 x 16 = 31216 disk revolutions
   ```

6. Multiply the number of disk revolutions by the number of seconds per revolution to get the number of seconds to read theinput:

   ```
   31216 x 0.0141 = 440 seconds to read the input
   ```

   **Note:** The number of seconds per disk revolution for the 3380 is 0.0166 and for the 9345 is 0.0112.

7. Multiply the number of seconds it takes to read the input by 2 to get the number of seconds of elapsed time required to read and write the data:

```
440 x 2 = 880 seconds to read and write the data
```

Thus, for this example, the estimated elapsed time to read and write the data is 14 minutes and 40 seconds.

If you calculate this estimate for a particular sort and use it as a basis for comparison, you will have a more realistic idea of what to expect when you try to improve the elapsed time. Note that this estimate is somewhat lower than the true minimum for elapsed time because it does not account for the additional seconds DFSORT takes to initialize the sort, prime the buffers, flush the I/O buffers, terminate the sort, and perform other administrative activities.

# Summary of Changes

## Release 13

## New Programming Support for Release 13

### DFSORT's Performance Booster for The SAS** System

DFSORT Release 13 provides significant CPU time improvements for SAS applications. To take advantage of this new feature, contact SAS Institute Inc. for details of the support they provide to enable this enhancement.

### Dynamic Hipersorting

Dynamic Hipersorting is a new, automatic feature that eliminates the unintended system paging activity and expanded storage and paging data set space shortages that sometimes resulted from a large amount of Hipersorting activity, especially from multiple concurrent Hipersorting applications.

Dynamic Hipersorting allows for more optimal DFSORT and system performance and provides installation options that allow you to customize HIPRMAX=OPTIMAL to your own criteria. With the advent of this feature, we recommend that you use HIPRMAX=OPTIMAL as your site default.

### Performance

Performance enhancements for DFSORT applications that use the Blockset technique include the following:

- Dataspace sorting, introduced in R12 for fixed-length record sort applications, now available for variable-length record sort applications (MVS/ESA only)
- Improved data processing methods for fixed-length record sort applications
- OUTFIL processing for producing multiple output data sets using a single pass over one or more input data sets.

### OUTFIL Processing

OUTFIL is a new DFSORT control statement that allows you to create one or more output data sets for a sort, copy, or merge application from a single pass over one or more input data sets. You can use multiple OUTFIL statements, with each statement specifying the OUTFIL processing to be performed for one or more output data sets. OUTFIL processing begins after all other processing ends (that is, after processing for exits, options, and other control statements). OUTFIL statements support a wide variety of output data set tasks, including:

- Creation of multiple output data sets containing unedited or edited records from a single pass over one or more input data sets.
- Creation of multiple output data sets containing different ranges or subsets of records from a single pass over one or more input data sets. In addition, records that are not selected for any subset can be saved in another output data set.
- Conversion of variable-length record data sets to fixed-length record data sets.
- Sophisticated editing capabilities such as hexadecimal display and control of the way numeric fields are presented with respect to length, leading or suppressed zeros, symbols (for example, the thousands separator and decimal point), leading and trailing positive and negative signs, and so on. Twenty-six pre-defined editing masks are available for commonly used numeric editing patterns, encompassing many of the numeric notations used throughout the world. In addition, a virtually unlimited number of numeric editing patterns are available via user-defined editing masks.

- Selection of a character or hexadecimal string for output from a lookup table, based on a character, hexadecimal, or bit string as input (that is, lookup and change).
- Highly detailed three-level (report, page, and section) reports containing a variety of report elements you can specify (for example, current date, current time, page number, character strings, and blank lines) or derive from the input records (for example, character fields, edited numeric input fields, record counts, and edited totals, maximums, minimums, and averages for numeric input fields).

## National Language Support

***Cultural Sort and Merge:*** DFSORT will allow the selection of an active locale at installation or run time and will produce sorted or merged records for output according to the collating rules defined in the active locale. This provides sorting and merging for single- or multi-byte character data based on defined collating rules which retain the cultural and local characteristics of a language.

***Cultural Include and Omit:*** DFSORT will allow the selection of an active locale at installation or run time and will include or omit records for output according to the collating rules defined in the active locale. This provides inclusion or omission for single- or multi-byte character data based on defined collating rules which retain the cultural and local characteristics of a language.

***OUTFIL Reports:*** OUTFIL allows date, time, and numeric values in reports to be formatted in many of the notations used throughout the world.

***ICETOOL Reports:*** ICETOOL's DISPLAY operator allows date, time, and numeric values in reports to be formatted in many of the notations used throughout the world.

## ICETOOL Enhancements

ICETOOL is now even more versatile as a result of enhancements to the existing operators. The improvements to ICETOOL include:

- Allowing more data to be displayed with the DISPLAY and OCCUR operators. DISPLAY now allows up to 20 fields (increased from 10) and a line length of up to 2048 characters (increased from 121). OCCUR now allows a line length of up to 2048 characters (increased from 121).
- More extensive formatting capabilities for numeric fields with the DISPLAY operator. Formatting items can be used to change the appearance of individual numeric fields in reports with respect to separators, decimal point, decimal places, signs, division, leading strings, floating strings and trailing strings. Thirty-three pre-defined editing masks are available for commonly used numeric editing patterns, encompassing many of the numeric notations used throughout the world. Leading and trailing strings can also be used with character fields.
- Display of the four-digit or two-digit year with the DISPLAY and OCCUR operators.
- Division of reports into sections with the DISPLAY operator, based on the values in a character or numeric break field. Statistics (total, maximum, minimum and/or average) can be displayed for each section as well as for the entire report.
- Automatic use of OUTFIL processing for a list of TO ddnames with the COPY and SORT operators, resulting in creation of multiple TO (output) data sets from a single pass over the FROM (input) data set.
- Allowing OUTFIL statements to be specified in the USING data set in addition to or instead of the TO operand with the COPY and SORT operators.

- Allowing the active locale to be specified for the COPY, COUNT and SORT operators, in order to override the installation default for the active locale. Thus, multiple active locales can be used in the same ICETOOL job step for these operators.
- Allowing the last record for each unique field value to be kept with the SELECT operator.

## INCLUDE/OMIT Substring Search

INCLUDE and OMIT function enhancements provide powerful substring search capability to allow inclusion or omission of records when:

- A specified character or hexadecimal constant is found anywhere within a specified input field (that is, a constant is a substring within a field) or
- A specified input value is found anywhere within a specified character or hexadecimal constant (that is, a field is a substring within a constant).

## SMF Type-16 Record Enhancements

New fields, such as information pertaining to each DFSORT run about SORTIN, SORTINnn, SORTOUT and OUTFIL data sets, control statements, record counts, specified values for E15, E35, HIPRMAX, DSPSIZE, FILSZ, LOCALE and AVGRLEN, have been added to DFSORT's SMF type-16 record.

SMF=FULL, SMF=SHORT, and SMF=NO can now be specified in an OPTION statement in DFSPARM or the extended parameter list, to produce or suppress the SMF type-16 record for an individual application.

**Note:** The offsets of fields ICESPGN, ICEUSER, and ICEGROUP have changed in the Release 13 SMF record. If you have programs that reference those fields, recompile them using the Release 13 version of the ICESMF macro, before attempting to run them against Release 13 SMF records.

## Other Enhancements

Several ICEMAC installation options have been added or changed:

- The IBM-supplied default for EXCPVR has been changed from ALL to NONE.
- The IBM-supplied default for DYNAUTO has been changed from NO to YES.
- SDBMSG enables you to specify whether DFSORT should use the system-determined optimum block size for DFSORT message data sets and ICETOOL message and list data sets.
- LOCALE enables you to select an active locale.
- ODMAXBF enables you to specify the maximum buffer space DFSORT can use for each OUTFIL data set.
- EXPMAX enables you to specify the maximum total amount of available storage to be used for all Hipersorting applications.
- EXPOLD enables you to specify the maximum total amount of old expanded storage to be used at any one time by all Hipersorting applications.
- EXPRES enables you to specify the minimum amount of available expanded storage to be reserved by DFSORT for use by non-Hipersorting applications.

Several run-time options have been added or changed:

- LOCALE enables you to select an active locale.
- SMF enables you to specify whether DFSORT is to produce SMF type-16 records.
- ODMAXBF enables you to specify the maximum buffer space DFSORT can use for each OUTFIL data set.

- NZDPRINT enables you to indicate that positive ZD summation results are not to be converted to printable numbers (overrides ZDPRINT).
- HILEVEL=YES on the MODS statement enables you to indicate that the E15 and E35 routines are to be treated as COBOL exits.
- DEBUG options BUFFERS=ANY and BUFFERS=BELOW will now be recognized but not used.

DFSORT will now ignore any DD statements not needed for the application (for example, a SORTIN DD statement will be ignored for a merge application).

For unsuccessful completion due to an unsupported operating system, DFSORT, ICEGENER, and ICETOOL will now pass back a return code of 24 to the operating system or invoking program.

The installation initialization exit, ICEIEXIT, enables you to specify the maximum buffer space DFSORT can use for each OUTFIL data set.

The installation termination exit, ICETEXIT, contains additional fields such as a flag to indicate that OUTFIL processing was used.

For INREC and OUTREC:

- The upper limit for columns and the end of fields has been raised from 32000 to 32752.
- 1: before the RDW field of variable-length records will be accepted and ignored.

For INCLUDE and OMIT, COND=ALL, COND=(ALL), COND=NONE, and COND=(NONE) enable you to include or omit all records.

The L2 value from the RECORD statement will be used if the L1 value is not specified when an E15 or E32 user exit passes all of the input records.

When input is a VSAM data set and output is a non-VSAM data set with RECFM not specified, DFSORT will now set the output RECFM as blocked rather than unblocked, when doing so will allow the use of the system-determined optimum block size for output.

## New Programming Support for Release 12 (PTFs)

ICEGENER, copy, and Blockset sort and merge can now be used when a tape output data set is specified with DISP=MOD or DISP=OLD, without specifying the RECFM, LRECL, or BLKSIZE in the DD statement.

Sequential striping is supported for input and output data sets.

Compression is supported for input and output data sets.

BatchPipes/MVS input and output pipes are supported.

## New Device Support for Release 12 (PTFs)

Four-digit device numbers are supported.

The IBM 3390-9 DASD is supported for input, output, and work data sets, although it is not recommended for work data sets for performance reasons.

The IBM RAMAC Array DASD and RAMAC Array Subsystem are supported for input, output, and work data sets.

The IBM 3990 Model 6 control unit is supported.

The IBM cached 9343 control unit models are supported.

# Index

## Numerics



## A



## B



## C

# D

## E



## F



## G



## H



## I



## J

## S



## T



## U



## V

# W

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

**DFSORT**
**Tuning Guide**

**Publication No. SC26-3111-02**

**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 _____

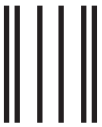
Company or Organization _____

Phone No. _____

**Readers' Comments — We'd Like to Hear from You**
SC26-3111-02

Cut or Fold
Along Line

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 Business Machines Corporation
RCF Processing Department
G26/050
5600 Cottle Road
SAN JOSE, CA 95193-0001

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



Cut or Fold
Along Line

Program Number: 5740-SM1

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC26-3111-02