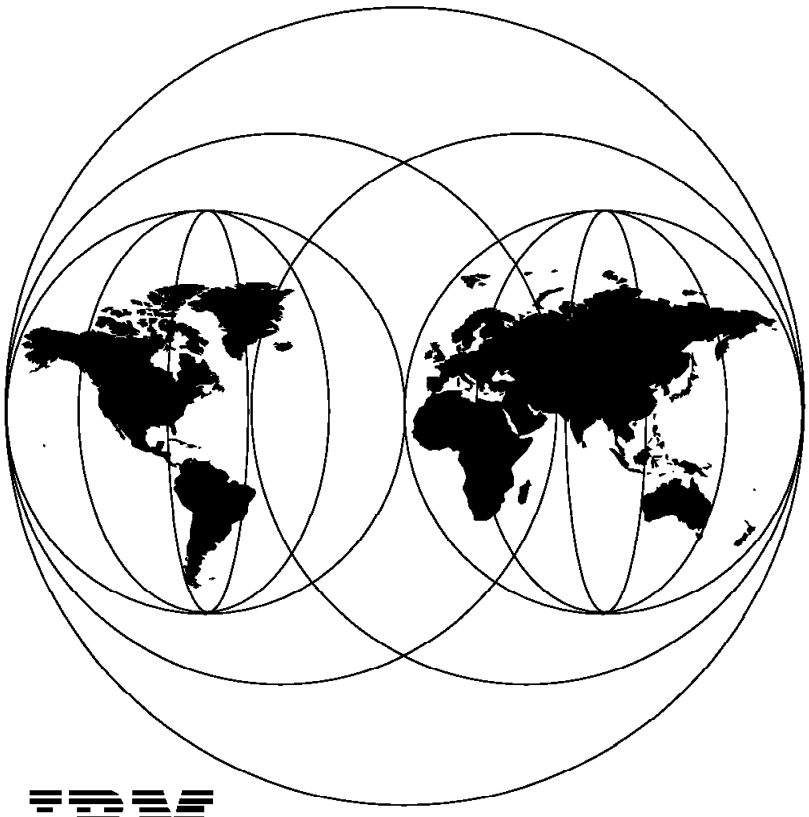
International Technical Support Organization

GG24-4476-00

DFSORT Release 13 Benchmark Guide

May 1995



IBM

International Technical Support Organization

San Jose Center

International Technical Support Organization

GG24-4476-00

DFSORT Release 13 Benchmark Guide

May 1995

Take Note!

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

First Edition (May 1995)

This edition applies to Release 13 of IBM DFSORT, Program Number 5740-SM1, and to any subsequent releases until otherwise indicated in new editions or technical newsletters.

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

An ITSO Technical Bulletin Evaluation Form for reader′s feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. 471 Building 070B
5600 Cottle Road
San Jose, California 95193-0001

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

Abstract

Sorting is one of the most frequently used functions at most data processing sites. Benchmarking is a method of comparing program products with similar function. Each product is run in a controlled environment on a similar collection of jobs to test its function and performance.

This book provides guidance to data processing organizations and IBM support personnel on how to plan, construct, and run a sort benchmark. It is assumed that the reader is familiar with IBM sort products or their equivalent.

(35 pages)

Contents

Abstract . . . iii

Special Notices . . . ix

Preface . . . xi
How This Document Is Organized . . . xi
Related Publications . . . xii
DFSORT Library . . . xii
Other Documentation . . . xii
Acknowledgments . . . xiii

Chapter 1. Introduction . . . 1

Chapter 2. Sort Benchmark: Job Selection . . . 3
2.1 Constructing a Sort Benchmark . . . 3
2.2 Using Vendor-Supplied Benchmarks: Advantages and Disadvantages . . . 4
2.3 Summary and Conclusions . . . 5

Chapter 3. Virtual Storage Considerations . . . 7
3.1 Above and below 16MB Virtual . . . 7
3.2 Using Exits to Monitor Storage for Sorts . . . 8
3.3 Using Hiperspaces and Data Spaces . . . 9
3.4 Using Multiple Output Data Sets . . . 9
3.5 Running Resident or Nonresident . . . 10
3.6 Using EXCPVR . . . 11

Chapter 4. Expanded Storage and Hiperspace Considerations . . . 13
4.1 Hipersorting with MVS/ESA Systems . . . 13
4.2 Allocation of Hiperspace . . . 13

Chapter 5. Data Space Considerations . . . 15
5.1 Sorting with Data Space on MVS/ESA Systems . . . 15
5.2 Sorting with Data Space on MVS/XA Systems . . . 16

Chapter 6. Allocation of Input, Output, and Work Data Sets . . . 17
6.1 How Many Work Data Sets? . . . 17
6.2 How Much Work Space? . . . 17
6.3 Channel and Device Separation . . . 17
6.4 Advantages of Using 3990 Cached Storage Control Units . . . 18
6.5 Work Data Set Considerations with MVS/ESA Systems and Hipersorting 18

Chapter 7. Installation Considerations . . . 21
7.1 Program Residency . . . 21
7.2 SVC . . . 21
7.3 System/370-XA Sorting Instructions . . . 21
7.4 Installation Parameters . . . 21
7.4.1 Equivalent Installation Parameters between Sort Products . . . 25
7.4.2 Optimum Mix of Sort Parameters: Benchmark and Production . . . 25

Chapter 8. How to Run a Sort Benchmark . . . 27
8.1 Obtaining Repeatable Results . . . 27

8.2 Avoiding Statistical Bias 28
8.3 Planning for Data Collection and Measuring 28
8.3.1 Measuring Elapsed Time 28
8.3.2 Measuring CPU Time 29

Appendix A. A Sort Benchmark Checklist 31

List of Abbreviations 35

Tables

1. Significant DFSORT Installation Parameters 22
2. Sort Benchmark Factors Checklist 31

Special Notices

This book is intended to help you to benchmark the performance of DFSORT Release 13 with competing sort products.

This book also documents product-sensitive programming interface and associated guidance information provided by DFSORT.

Product-sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning DFSORT. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Product-sensitive programming interface and associated guidance information is identified where it occurs by an introductory statement to a chapter or section.

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 IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

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 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (VENDOR) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

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

DFSORT
Enterprise Systems Architecture/390
Enterprise Systems Architecture/370
Enterprise Systems Connection Architecture
ESA/370
ESA/390
ESCON
Hipersorting
Hiperspace
IBM
MVS/ESA
MVS/XA
RMF
S/370
S/390
System/370
System/390

Preface

Sorting is one of the most frequently used functions at most data processing sites. Benchmarking is a method of comparing program products with similar function. Each product is run in a controlled environment on a similar collection of jobs to test its function and performance.

This book provides guidance to data processing organizations and IBM support personnel on how to plan, construct, and run a sort benchmark. It is assumed that the reader is familiar with IBM sort products or their equivalent.

The observations, recommendations, and conclusions in this book are the result of work by IBM in constructing and running sort benchmark job streams. These observations, recommendations, and conclusions may or may not apply to the reader's own environment. The results the reader may achieve by adopting all or some of the recommendations in this book may vary accordingly.

How This Document Is Organized

This book contains the following sections:

- Chapter 1, "Introduction" on page 1 discusses the purpose of this book and why it is important to conduct a sort benchmark.
- Chapter 2, "Sort Benchmark: Job Selection" on page 3 discusses the factors in constructing a sort benchmark as well as the considerations when using a vendor-supplied benchmark.
- Chapter 3, "Virtual Storage Considerations" on page 7 discusses virtual storage and its effects on sort and system performance.
- Chapter 4, "Expanded Storage and Hiperspace Considerations" on page 13 discusses Hiperspace and its effects on expanded storage, sort performance, and system performance.
- Chapter 5, "Data Space Considerations" on page 15 discusses data space and its effects on sort and system performance.
- Chapter 6, "Allocation of Input, Output, and Work Data Sets" on page 17 discusses the considerations when allocating input, output, and work data sets.
- Chapter 7, "Installation Considerations" on page 21 discusses the options and parameter defaults that must be considered at installation time in order to conduct a good and fair benchmark comparison.
- Chapter 8, "How to Run a Sort Benchmark" on page 27 discusses the major factors in running a sort benchmark.
- Appendix A, "A Sort Benchmark Checklist" on page 31 provides a comprehensive list of considerations in setting up and running a sort benchmark.

Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document.

DFSORT Library

- *DFSORT Brochure*, GC33-4033
- *DFSORT Licensed Program Specifications*, GC33-4032
- *Getting Started with DFSORT*, SC26-4109
- *DFSORT Messages, Codes and Diagnosis Guide*, SC26-7057
- *DFSORT Application Programming Guide*, SC33-4035
- *DFSORT Installation and Customization*, SC33-4034
- *DFSORT Panels Guide*, SC26-7037
- *DFSORT Tuning Guide*, SC26-3111
- *DFSORT Reference Summary*, SX33-8001.

The entire DFSORT library can be ordered by using order number SBOF-1243.

All of the unlicensed publications, except the *Reference Summary*, are available on CD-ROM as part of the *MVS Collection Kit*, SK2T-0710.

Other Documentation

- Using SMF records
  - *MVS/ESA SP V4 System Management Facilities (SMF)*, GC28-1628
  - *MVS/ESA SP V5 System Management Facilities (SMF)*, GC28-1457
  - *MVS/XA SMF*, GC28-1153
- Using IEALIMIT and IEFUSI exits
  - *MVS/ESA SP V4 Installation Exits*, GC28-1637
  - *MVS/ESA SP V5 Installation Exits*, SC28-1459
  - *MVS/XA SPL: User Exits, GC28-1147*
- Analyzing RMF reports
  - *MVS/ESA Analyzing RMF Version 4 Monitor I & II Reports*, LY28-1007
  - *MVS/XA RMF Monitor I & II Reference and User's Guide*, LC28-1556

A complete list of International Technical Support Organization publications, with a brief description of each, may be found in:

Bibliography of International Technical Support Organization Technical Bulletins, GG24-3070.

To get a catalog of ITSO technical publications (known as “redbooks”), VNET users may type:

```
TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
```

How to Order ITSO Technical Publications

IBM employees in the USA may order ITSO books and CD-ROMs using PUBORDER. Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-284-4721. Visa and Master Cards are accepted. Outside the USA, customers should contact their local IBM office.

Customers may order hardcopy ITSO books individually or in customized sets, called GBOFs, which relate to specific functions of interest. IBM employees and customers may also order ITSO books in online format on CD-ROM collections, which contain books on a variety of products.

Acknowledgments

The author of this document is:

- Bruce Wagar, IBM Storage Systems Division, San Jose.

Thanks to the following people for their assistance in the production of this document:

- Maggie Cutler, Technical Editor
- Joshua Peleg, International Technical Support Organization, San Jose Center
- Frank Yaeger, IBM Storage Systems Division, San Jose.

Chapter 1. Introduction

Sorting is one of the most important parts of a site's data processing workload. Many sites run thousands of sorts per week. It is difficult to think of a nontrivial data processing application that does not use sorting.

The performance of a sort is critical. The evaluation of the performance of a sort product offering should not rely solely on the results of a sort vendor's benchmark. The vendor's benchmark probably will not represent any individual customer's production environment and may very well contain jobs for which the vendor's product performs unusually well. It may be true that the vendor benchmark results influence the choice of a sort vendor, but this should not be the only information used when making a sort product decision. Sort product performance should be substantiated in the environment in which it will be used. Sort jobs should be used that will most closely parallel how the product will perform in the live production environment.

When one considers how much resource is used for sort jobs and how critical sorting is to the data processing production environment, it becomes clear that the investment in time and resources to build a truly representative sort benchmark is well worth the effort. The consequence of not making that investment may be that the selected sort product does not achieve the anticipated performance levels, and the throughput of the overall data processing environment will be negatively impacted.

Once the sort benchmark is developed, it can be used to evaluate future releases of sort products. If the benchmark is to maintain its value to the installation, it should be kept current to reflect changes in the sort production environment.

To determine which sort product to install in your production environment, you should evaluate the performance and functional characteristics of the various sort product offerings. Then you can decide which product satisfies your particular requirements. The functional evaluation can be made by doing a "paper" analysis of the product description of each sort product. However, to evaluate the relative performance of each sort product, you may want to conduct an evaluation of each sort product's performance in your environment.

The book provides guidance on how to plan for such a sort performance evaluation. Many sort customers do not install a new sort product directly into the installation's production environment. The system programmer responsible for a sort performance evaluation will probably need to evaluate each sort product's performance characteristics by doing a sort benchmark before the new sort is installed.

This book covers the following subjects:

- How to select a sort benchmark job
- Virtual storage considerations
- Expanded storage and hiperspace considerations
- Data space considerations
- Allocation of input, output, and work data sets

- Installation considerations
- How to run a sort benchmark
- A final checklist to help ensure that the sort benchmark being constructed has considered all factors relevant to sort product performance.

The remaining chapters of this book address these subjects. The adoption of some or all of the recommendations in the book should result in a more thorough, comprehensive comparison of the sort products' performance characteristics.

Chapter 2. Sort Benchmark: Job Selection

The most important task in running a sort benchmark is the proper selection of the jobs that make up the benchmark. Two alternatives are available:

- Construct a sort benchmark, using typical production jobs
- Use sort benchmark jobs supplied by a sort product vendor.

2.1 Constructing a Sort Benchmark

Constructing a sort benchmark requires the most effort. The amount of effort required varies according to the degree of analysis that you perform. The analysis can range from selecting jobs that can be easily run to analyzing the sort workload in the production environment. The effort should result in selecting sort jobs that are either frequently run or the biggest consumers of system resources or both.

When constructing a sort benchmark, you should consider selecting only those sort jobs that are representative of sorts run in the production environment. The paragraphs that follow provide guidance on how to select those jobs.

The factors to consider when selecting benchmark jobs can be summarized as follows:

- When selecting benchmark candidates, look for sort jobs that have one or more of the following characteristics:
  - Are long running and must run within a fixed amount of time or “batch window”
  - Are frequently run, one or more times per day
  - Are invariably an integral part of critical or major applications
  - Use a relatively large percentage of central processing unit (CPU) resource.
  - Have a relatively large amount of I/O activity
  - Are invoked from other applications, for example, COBOL, SAS, DB2, and the BLDINDEX function of IDCAMS
  - Use the sort product′s copy or merge functions (do not necessarily limit the benchmark to sort functions).
  - Use sort options and functions found in production jobs, such as exits (E15 and E35), record level editing (INREC, OUTREC, and OUTFIL), record filtering (INCLUDE, OMIT, and OUTFIL), record summarization (SUM), multiple output data sets (OUTFIL), and reports (OUTFIL).
- Once you have identified the benchmark job characteristics, you can pull the jobs that make up a sort benchmark directly from the production environment, or you can develop a prototype of the sort jobs. Whichever you choose, select sort jobs that represent the following characteristics in the majority of your production jobs:
  - Record length
  - Sort key length

- Record format (fixed, variable, spanned)
- Data set size (number of records, number of bytes)
- Degree of randomness of the data (whether the data is presequenced)
- Virtual storage size
- Amount of hiperspace available
- Amount of data space available
- Job control language (JCL) REGION size
- Input and output data sets—direct access storage device (DASD) or tape, striped or compressed
- Number and location of work data sets
- Number of output data sets
- Sort product functions and options (for example, exits, INREC, OUTREC, INCLUDE, OMIT, SUM, and OUTFIL).

These characteristics will most likely have the greatest impact on sort performance. Because the purpose of a benchmark is to evaluate performance that will in fact occur in a production environment, the characteristics of the sorts from the production environment that have the greatest effect on sort performance must be preserved.

Most sort products also offer copy and merge functions. If these functions are used in the production environment, it is important to include copy and merge applications in the benchmark.

2.2 Using Vendor-Supplied Benchmarks: Advantages and Disadvantages

Using a vendor-supplied benchmark has advantages and disadvantages. The obvious advantages of the vendor-supplied benchmark are that:

- You save the time and effort required to construct a benchmark.
- The jobs are already defined and can be very easy to run.
- Very little preparation or analysis is necessary. All you need to do is collect the data from the runs and summarize the results.

The obvious disadvantages of a vendor-supplied benchmark are that:

- The jobs may not represent a given production environment, so the results of the benchmark may not be applicable in your production environment.
- The following sort product parameters that affect sort performance may not be similar to the parameters used in your production environment:
  - Record formats and sizes
  - Data set sizes
  - Degree of randomness in data
  - Virtual storage size
  - Amount of hiperspace available
  - JCL REGION sizes
  - Method of invocation, direct or by another program

  - Inclusion or omission of user exits
  - Sort product installation options
  - Presence or absence of functions such as exits, record level editing, record filtering, record summarization, reports, and multiple output data sets.
  - Number and location of work data sets available
  - Number of output data sets
  - Mixture of sort jobs that make up the vendor's workload.
- You may not be able to re-create the vendor's performance improvements on the site's configuration. The configuration used by the vendor to obtain the quoted benchmark results may be different from the site's configuration in the following ways:
  - CPU and amount of central storage available
  - Amount of expanded storage available
  - Configuration of DASD, storage control units, and channels
  - Operating system release and maintenance levels
  - Device and channel separation of:
    - Input and output data sets from the work data sets
    - Output data sets from each other
    - Work data sets from each other.

2.3 Summary and Conclusions

Vendor-supplied benchmarks can never totally represent the sort workload in every data processing operation. The construction of a sort benchmark should involve the careful, thoughtful selection of sort jobs that accurately reflect jobs from production in terms of:

- How much of the system's resource is used by each sort job
- How important the sorts are to the overall production workload
- How frequently the sorts are run
- Having the same sort job parameters:
  - Record sizes
  - Sort key sizes
  - Record formats
  - Data set sizes
  - Randomness of data
  - Virtual storage allocation
  - Hiperspace allocation
  - Data space creation
  - Input and output data sets—DASD, tape, or pipe; striped or compressed; VSAM or sequential
  - Number and location of output data sets

  - Number and location of work data sets
  - Sort product functions (exits, record level editing, record filtering, record summarization, and reports).
- Use of the sort product's copy and merge functions
- Having the same system configuration parameters:
  - CPU model and amount of central storage
  - Amount of expanded storage
  - DASD, storage control unit, and channel configuration
  - Operating system release and maintenance level
  - Device separation of:
    - Input and output data sets from the work data sets
    - Work data sets from each other
    - Output data sets from each other.

Since sorts are a critical and fundamental part of data processing applications and usually consume a significant amount of system resource, the first step toward evaluating sort products is to build a sort benchmark that represents the sort jobs run in the production environment. This custom-built benchmark can then be used to evaluate future sort product offerings as well as future releases of the currently installed sort product.

Chapter 3. Virtual Storage Considerations

In general, using more virtual storage helps the sort product run more efficiently by letting it process large pieces of data at a time. Having enough virtual storage reduces elapsed time, CPU time, and execute channel program (EXCP) counts.

For small input data sets, a sufficiently large virtual storage area might accommodate an in-memory sort (that is, one done entirely within virtual storage). Generally speaking, Data Facility Sort (DFSORT) requires a storage allocation of 1.4 times the input data set size (in bytes) to accomplish an in-memory sort. Once a sort can be accomplished in memory, you cannot obtain additional improvements by increasing the virtual storage.

For sorts that cannot be accomplished in memory, you must write data to temporary storage. The data can be placed either in expanded storage or on DASD. Increasing the allotted virtual storage can make a significant contribution in reducing the amount of data movement by allowing larger pieces of data to be processed at a time. It is important to use large virtual storage areas when sorting large data sets. Be sure, however, that there is *sufficient central storage* to back up the virtual storage requested. Otherwise, the system may experience excessive paging, and the performance of the sort jobs may be severely degraded.

3.1 Above and below 16MB Virtual

In many customer environments, the virtual storage space below 16MB virtual is constrained. DFSORT helps to relieve this constraint by allocating most of its storage above 16MB virtual. Using storage above 16MB virtual also allows more virtual storage to be allocated to DFSORT, which improves its performance in many cases.

Most of the time the extra storage above 16MB virtual is available to DFSORT without requiring the user to change the JCL for the sort job. For example, assume 512KB virtual storage has been requested in the JCL. DFSORT, with its installation defaults, uses a total of 4MB virtual storage.

DFSORT has dataspace sorting capability (fixed-length record sort only) that uses data space available with the Multiple Virtual Storage/Enterprise Systems Architecture (MVS/ESA) operating system on Enterprise Systems Architecture/System 370 (ESA/370) and Enterprise Systems Architecture/System 390 (ESA/390) systems. Dataspace sorting for fixed length records is also available on Multiple Virtual Storage/Extended Architecture (MVS/XA) systems through use of virtual dataspace, a DFSORT capability that enables dataspace sorting on MVS/XA systems by simulating data space. With dataspace sorting, DFSORT determines at run time how much virtual storage to acquire. This gives DFSORT the flexibility to use more virtual storage if appropriate and without requiring any changes to the JCL. *Most of DFSORT's storage is allocated above 16MB virtual* (or in a separate data space, for MVS/ESA dataspace sorting).

The DFSORT MAXLIM installation option specifies the default maximum amount of virtual storage that DFSORT attempts to allocate below 16MB virtual in the primary address space.

Recommendation:

Ensure that the amount of storage specified below 16MB virtual is equivalent for all sort products in the benchmark.

The DFSORT installation option TMAXLIM specifies the default maximum total amount of virtual storage that DFSORT attempts to allocate above and below 16MB virtual in the primary address space. Data space creations (for dataspace sorting runs) are in addition to this storage and are controlled by the DSPSIZE option. You can override TMAXLIM at run time by using the EXEC PARM option, SIZE, or the DFSORT OPTION control statement option MAINSIZE. Specifying SIZE=MAX or MAINSIZE=MAX (instead of specifying a specific value for SIZE or MAINSIZE) causes DFSORT to use TMAXLIM for the maximum amount of storage.

We recommend that you keep the IBM-supplied installation default of SIZE=MAX and use TMAXLIM to control total available storage.

The DFSORT program product has an installation default for TMAXLIM of 4194304 (4MB). If this number is increased, central storage should be sufficient to support the larger virtual region and avoid excessive paging. For most cases, 4MB is sufficient storage to perform an efficient sort. Very large sorts (say, on data sets of more than 100MB) benefit from better performance, however, if they are provided with more storage. For these large applications, we recommend that you override TMAXLIM by using the SIZE or MAINSIZE option. More information on tuning virtual storage can be found in the *DFSORT Application Programming Guide* or *DFSORT Tuning Guide*.

3.2 Using Exits to Monitor Storage for Sorts

The information in this section is part of the product-sensitive programming interface.

DFSORT supports an installation-wide initialization exit routine (ICEIEXIT), which, among other things, can control the virtual storage use of DFSORT′s applications. When an ICEIEXIT routine is installed with DFSORT, it receives control from DFSORT before running each sort job. Among other things, an ICEIEXIT routine can be used to override the storage options in effect at run time, based on such factors as time of day or system control block information. The exit allows system programmers to prevent sort jobs from degrading overall system performance when the system is heavily loaded by limiting the amount of virtual storage available to DFSORT. Conversely, the exit can be used to improve the performance of sort jobs when the system is lightly loaded by increasing the amount of storage available to DFSORT. An ICEIEXIT routine can eliminate the need for manual intervention for each sort job by allowing the storage available to DFSORT to be dynamically controlled on the basis of overall system and individual job considerations.

Recommendation:

Ensure that all sort products have access to the same amounts of storage during a benchmark. It is important that initialization exit routines not allow different sort products to use different amounts of storage during a benchmark. The use of different amounts of storage would provide an unfair advantage to the sort product that receives more virtual storage, thus biasing the results.

3.3 Using Hiperspaces and Data Spaces

For MVS/ESA systems, sort products can acquire virtual storage outside their primary address space by using hiperspaces and data spaces. DFSORT, for instance, uses hiperspaces as a supplement or replacement for work data sets with its Hipersorting feature. In addition, DFSORT offers dataspace sorting for fixed-length record sorts, which uses a data space as an extension to the region in which the sort would ordinarily run. Other sort products may offer the same or similar features.

Hiperspace pages are never backed by central storage; they are backed primarily by expanded storage, with auxiliary storage used when available expanded storage is insufficient. In order to access data in a hiperspace, the data must be moved into central storage in page-size increments. This process requires extra processor time. It also makes it necessary to access hiperspace data in relatively large pieces. On the plus side, hiperspaces have little impact on central storage paging activity.

Unlike hiperspace pages, data space pages are backed primarily by central storage. Assuming enough available central storage to back the data space pages, access to data in the data space is just as fast as access to any other data in central storage, and the data does not have to be accessed in large pieces. Unfortunately, data spaces can have a significantly overload central storage paging activity.

Recommendation:

As with primary virtual storage, try to ensure that each sort product in a benchmark has access to the same amounts of hiperspace and data space. This is especially important for a data space, because it can be used in almost the same way as primary virtual storage.

3.4 Using Multiple Output Data Sets

DFSORT controls the amount of virtual storage per each output data set. This storage is in addition to the storage normally needed by DFSORT.

The value of the ODMAXBF parameter specifies the maximum buffer space to be used for each OUTFIL data set. The ODMAXBF value can be specified as an installation or run time parameter, or in an ICEIEXIT routine. The default value of 2M is recommended for the ODMAXBF option in effect. Lowering ODMAXBF can cause performance degradation for the application but might be necessary if you consider the amount of storage used for OUTFIL processing to be a problem. Raising ODMAXBF can improve EXCPs for the application but can also increase the amount of storage needed.

The storage used for OUTFIL processing will be adjusted automatically according to the total storage available, the storage needed for non-OUTFIL processing, and the number of OUTFIL data sets and their attributes (for example, block size). OUTFIL processing will be subject to the ODMAXBF limit in effect and the system storage limits (for example, IEFUSI), not to the DFSORT storage limits (that is, SIZE, MAXLIM, and TMAXLIM). DFSORT attempts to use storage above the 16MB virtual for OUTFIL processing whenever possible.

Recommendation:

Ensure that each sort product in a benchmark has access to the same amount of additional virtual storage when processing multiple output data sets.

Use the ODMAXBUF default value, which is up to 2M for each output data set.

3.5 Running Resident or Nonresident

Running resident means running the sort product from the pageable link pack area (LPA) or the extended link pack area (ELPA). These areas are common to all MVS primary address spaces, so the code in them does not need to be fetched from auxiliary storage. To be in the LPA or ELPA, the sort product modules must be located in SYS1.LPALIB.

Running nonresident means that the modules are linked from a program library, such as SYS1.LINKLIB, or a private library. The modules are run from the private portion of the primary address space, so they need to be fetched from auxiliary storage. Running with resident modules is more efficient because the modules do not have to be fetched from auxiliary storage and they do not count against the virtual storage limits of the region in which the application is running.

With DFSORT, two options for product installation allow a site to balance space constraints on LPALIB with performance requirements:

- Only the modules containing Blockset (the primary sorting technique) are installed resident in LPALIB. This is the recommended method.
- DFSORT is installed nonresident, with all modules in SYS1.LINKLIB or a private library.

Experience has shown that while the issue of program residency is significant for small sorts, the distinction becomes minimal for large sorts. DFSORT typically uses approximately 10 EXCPs for program load when its modules reside in a private library. Because sorting of large data sets uses thousands of EXCPs, loading sort product modules from program libraries should not greatly affect the results.

Moreover, for benchmark purposes, it might not be possible to place modules from different sort products or releases in LPALIB. There may be conflicts in the module names among sort products. These modules would have to be assigned different module names if they were to be resident simultaneously in LPALIB. Changing the modules names may result in the incorrect running of the sort products being evaluated. If each set of sort modules is placed in PLPA one at a time, an initial program load (IPL) with CLPA option would be required each time.

Recommendation:

For the above reasons, to achieve an objective evaluation, we strongly recommend that you place in private libraries all of the sort products you compare.

3.6 Using EXCPVR

The use of the execute channel program virtual real (EXCPVR) supervisor call (SVC) for I/O reduces the CPU overhead required to perform I/O. DFSORT primarily uses the EXCP and EXCPVR SVC for its I/O. When using EXCP, the operating system performs page fixing and freeing each time an I/O occurs. When using EXCPVR, DFSORT performs page fixing and page freeing only once for the entire run. Using EXCPVR significantly reduces the number of page fix and page free operations, which in turn reduces the CPU time.

The use of EXCPVR may result in increased paging rates and cause other programs to be swapped out, especially when using larger storage areas or on heavily loaded systems. This phenomenon is less likely to occur on systems with large amounts of central storage.

Recommendation:

Use EXCPVR to reduce CPU time if your system has sufficient central storage and low paging activity. On DFSORT, EXCPVR=ALL fully enables EXCPVR and is the IBM-supplied default. If a high paging rate is a concern, try using EXCPVR=NOWRK. This causes DFSORT to use EXCPVR for input and output data sets, but not for work data sets. It reduces the number of pages that are fixed but still provides improved performance. If you do not want DFSORT to use EXCPVR, specify EXCPVR=NONE.

Notes:

1. You must first install the DFSORT SVC if you want DFSORT to use EXCPVR.
2. DFSORT does not use EXCPVR for OUTFIL data sets.

Chapter 4. Expanded Storage and Hiperspace Considerations

A number of sort products offer improved performance through the use of hiperspaces. Although these products may differ in how they use hiperspaces, they improve elapsed time and I/O performance by using expanded storage as a fast alternative to DASD.

4.1 Hipersorting with MVS/ESA Systems

Hipersorting is a DFSORT capability that uses the hiperspace feature of the MVS/ESA operating system along with expanded storage to optimize sort performance. With Hipersorting, DFSORT uses hiperspace instead of, or in addition to, DASD for temporary storage of records. Hiperspace is backed first by expanded storage, and then by auxiliary storage. Because hiperspace data generally resides in expanded storage, the benefits associated with expanded storage also apply to Hipersorting.

Hipersorting significantly reduces I/O processing because some or all of the records are no longer written to temporary DASD work space. The reduced I/O processing in turn reduces elapsed time, channel usage, and EXCPs.

4.2 Allocation of Hiperspace

DFSORT dynamically determines and allocates the amount of hiperspace it needs. A rule of thumb is that, in most cases, DFSORT requires at least half the input data set size for the hiperspace work space. Otherwise, Hipersorting is not used. Note also that if DFSORT has more intermediate data than the hiperspace holds, it writes the excess data to DASD work data sets.

In addition to the amount of data being sorted, several other factors can limit the amount of hiperspace used by an application:

- The IEFUSI exit can limit the total amount of hiperspace and data space available to an application.
- HIPRMAX can limit the amount of hiperspace available to an application.
- Sufficient available expanded storage must be present to back DFSORT′s hiperspaces. “Available” expanded storage is defined as the expanded storage that MVS/ESA uses to back new hiperspace data and consists of the following two types:
  - Free expanded storage. This is expanded storage that is not being used by any application.
  - Old expanded storage. This is expanded storage that is being used by another application, but whose data has not been referenced for a sufficiently long period such that MVS/ESA migrates it to auxiliary storage to make room for new hiperspace data.

  The amount of available expanded storage changes constantly according to current system activity. Consequently, DFSORT checks the available expanded storage level throughout a Hipersorting application and switches from hiperspace to work data sets if the available expanded storage level gets too low.

- Other concurrent Hipersorting applications further limit the amount of available expanded storage. A Hipersorting application is aware of the expanded storage needs of every other Hipersorting application on the system and does not attempt to back its hiperspace data with expanded storage needed by another Hipersorting application. This prevents overcommitment of expanded storage resources in the event of multiple large concurrent Hipersorting applications starting at similar times on the same system.
- The EXPMAX, EXPOLD, and EXPRES installation options can also be used to further limit the amount of expanded storage available to Hipersorting applications. EXPMAX limits the total amount of available expanded storage that can be used at any one time to back DFSORT hiperspaces. EXPOLD limits the total amount of old expanded storage that can be used at any one time to back DFSORT hiperspaces. EXPRES sets aside a specified amount of available expanded storage that is reserved for use by non-Hipersorting applications.

Some of these limits depend on system and other Hipersorting activity throughout the time a Hipersorting application runs. Consequently, the amount of hiperspace a Hipersorting application uses can vary from run to run.

Recommendation:

Use Hipersorting to optimize elapsed time, channel busy time, device connect time, and EXCPs when running on MVS/ESA systems. HIPRMAX=OPTIMAL is the IBM-supplied default. It is more efficient to limit total expanded storage usage by all Hipersorting applications in a system (through EXPMAX, EXPOLD, and EXPRES) than it is to impose a limit on each individual Hipersorting application.

Each of the installation options (EXPMAX, EXPOLD, EXPRES, and HIPRMAX) fulfills a different role, and each works somewhat independently of the others. Setting EXPMAX=MAX, EXPOLD=MAX, EXPRES=0, and HIPRMAX=OPTIMAL (the defaults) permits the maximum amount of Hipersorting, subject to available expanded storage and other concurrent Hipersorting activity. Setting either EXPMAX=0, EXPRES=MAX, or HIPRMAX=0 disables Hipersorting (although HIPRMAX can be overridden at run time). In general, setting EXPMAX, EXPOLD, and HIPRMAX to large values (and EXPRES to a small value) allows a generous amount of Hipersorting, whereas setting either EXPMAX, EXPOLD, or HIPRMAX to a small value (or EXPRES to a large value) allows only a limited amount of Hipersorting.

Chapter 5. Data Space Considerations

A number of sort products offer improved performance through the use of data spaces. Although these products may differ in how they use data spaces, they improve performance by using more central storage and less auxiliary storage.

5.1 Sorting with Data Space on MVS/ESA Systems

Dataspace sorting is a DFSORT capability for fixed-length record sorts that uses the data space feature of MVS/ESA systems to optimize DFSORT performance. With dataspace sorting, DFSORT uses data space as an extension to the virtual storage in which it would ordinarily run. Like primary virtual storage, data space is backed first by central storage, then by expanded storage, and then by auxiliary storage. Because dataspace sorting generally uses more virtual storage, the benefits associated with increased virtual storage also apply to dataspace sorting.

Dataspace sorting improves performance significantly by allowing:

- Use of better internal methods
- More in-memory sorts
- Larger I/O buffers (which in turn reduces I/O processing)
- DFSORT to select the optimal amount of virtual storage with which to perform an efficient sort.

DFSORT automatically determines and creates the amount of data space it needs. A rule of thumb is that, for in-memory sorts, DFSORT requires a data space about 10% larger than the input data set size. For sorts that are too large to be in-memory sorts, DFSORT requires a data space large enough to hold a small portion of the input data set. Typically, this data space size can be 5% or 6% of the input data set size, sometimes less. Note that DFSORT does not use dataspace sorting and Hipersorting at the same time.

In addition to the size of the input data set, four other factors affect how much data space DFSORT creates:

- Before sorting, DFSORT queries the system about central storage availability. on the basis of the information returned, DFSORT creates data space, as needed, up to the point that DFSORT would affect the performance of other programs currently using central storage. As a result, it is possible for successive runs of the same sort job to use varying amounts of data space and DASD work space.
- If, when running a dataspace sorting application, DFSORT queries the system and discovers that paging activity is too high, it reduces the size of its data space and frees up the excess pages. DFSORT tries to keep enough data space, however, to perform an efficient (though not necessarily optimal) sort.
- You can control the *maximum* amount of data space for dataspace sorting with the DSPSIZE installation or run time option. With DSPSIZE, you can either set a limit on the amount of data space available to DFSORT or choose to allow DFSORT to use as much data space as it needs. In either case, DFSORT does not create any more data space than is available in central storage.

- The maximum amount of data space that can be created is further restricted by the IEFUSI limit, if any.

Recommendation:

Use dataspace sorting as an alternative to Hipersorting to optimize performance when running on MVS/ESA systems. DSPSIZE=MAX is the IBM-supplied default. It allows DFSORT to use as much data space as it needs. To limit the amount of data space available to DFSORT, specify DSPSIZE=n, where n is the number of megabytes of data space that DFSORT can use. To suppress dataspace sorting entirely, specify DSPSIZE=0.

5.2 Sorting with Data Space on MVS/XA Systems

DFSORT can simulate dataspace sorting on MVS/XA systems through use of its virtual dataspace capability. With virtual dataspace, DFSORT performs dataspace sorting using a separate storage acquisition within the primary addressspace instead of an actual data space creation. Thus, DFSORT is further limited by the IEFUSI ceiling on the total amount of virtual storage available in the primary address space. Furthermore, with virtual dataspace, DFSORT does not check the system to see what effect its storage acquisition is having or will have on system paging. Except as noted above, DFSORT performs dataspace sorting in the same way on MVS/XA systems as it does on MVS/ESA systems, and with the same benefits.

The DFSORT installation option, VIRTDSP, controls the use of virtual dataspace. It either enables or disables the capability on MVS/XA systems. If VIRTDSP is enabled, the DSPSIZE parameter controls dataspace sorting as on MVS/ESA systems.

Recommendation:

Use virtual dataspace as a way to optimize sort performance when running on MVS/XA systems. VIRTDSP=NO is the IBM-supplied default. It disables virtual dataspace.

We recommend changing the IBM-supplied default to VIRTDSP=YES, unless paging is a serious problem on your system. You can use the DSPSIZE option to further limit the amount of dataspace sorting.

Chapter 6. Allocation of Input, Output, and Work Data Sets

In this chapter we discuss the issues to consider when allocating data sets on DASD to be used by DFSORT.

6.1 How Many Work Data Sets?

The characteristics of the production work (SORTWKnn) data sets should be reflected in the benchmark. The critical concern is that there be enough work space available to use. Using more than one work data set may improve performance, but only if the data sets are located on separate devices. The assignment of too many small work data sets may degrade performance because each work data set must be opened and closed. Many applications average around three work data sets, but you should use what is common for your installation.

DFSORT uses two or three work data sets more efficiently than it uses a single work data set, provided that each work data set is on a separate device.

6.2 How Much Work Space?

The amount of space allocated to the work data sets is also a concern. Generally speaking, one and one-half to two times the input data set size should suffice as the necessary work space. Certain conditions (such as record level editing or running in a small virtual storage) might require more work space. Allocating two or three work data sets, each at least as large as the input data set, works well. For multivolume (or very large) input data sets, allocate additional work data sets.

For benchmark jobs, enough work space should be allocated in the primary extents because secondary extent processing can cause a large amount of allocation overhead in the middle of the sort. Also, separate extents need to be accessed with separate channel programs, thus increasing EXCPs. For production jobs, be generous with secondary extent sizes to cover unanticipated work space needs.

6.3 Channel and Device Separation

The proper arrangement of channels and devices to achieve optimal elapsed time results is very important. Configurations that prevent DFSORT from using its I/O capabilities efficiently are a prime source of poor performance. Careful attention to the following items should result in improved sort elapsed time performance:

- Place each output data set on a separate device.
- Place each work data set on a separate device, not on the same device as an input or output data set.
- Avoid using storage devices that are shared with other systems, or that share control units with other systems, or that share a channel with system activity (for example, system residence volumes, paging volumes).

- Use devices accessible from two paths (that is, two channels and two storage control units).
- Use the highest speed devices (especially for work data set storage), such as the 3390-3 or RAMAC Array DASD with a 3990 cached storage control unit, or a RAMAC Array Subsystem.

6.4 Advantages of Using 3990 Cached Storage Control Units

DFSORT takes advantage of the cache of the 3990 cached storage control unit by setting the appropriate caching mode for accessing each of its data sets located on DASDs controlled by 3990 cached storage control units.

DFSORT can also use the cache fast write feature of the 3990 cached storage control unit to reduce elapsed time, provided the cache fast write feature is enabled on the 3990 cached storage control unit. Cache fast write is used to speed up access times of work data sets located on DASDs controlled by 3990 cached storage control units. This DFSORT function is controlled by the cache fast write (CFW) installation or run time option.

Recommendation:

We recommend using the IBM-supplied value, CFW=YES. This enables DFSORT to use cache fast write on 3990 cached storage control units for its work data sets.

Note: The DFSORT SVC must also be installed for DFSORT to use CFW.

6.5 Work Data Set Considerations with MVS/ESA Systems and Hipersorting

In addition to improving sort performance, Hipersorting can also reduce DASD work space. When using Hipersorting, one of two situations takes place:

- Sufficient hiperspace available—no DASD work space required
- Insufficient hiperspace available—some DASD work space required

In both instances, the DASD requirement for the sort can be reduced. However, if work data sets are specified in the JCL, the savings may not be realized. Whether or not DFSORT uses the DASD work space, if work data set DD statements are specified in the JCL, the DASD space in the primary extent will not be freed up until the sort has completed.

Recommendation:

To realize the DASD work space savings provided by Hipersorting, **avoid specifying SORTWKnn data sets in the JCL**. Instead, allow DFSORT to dynamically allocate DASD work space on an “as needed” basis.

When the installation option DYNAUTO=IGNWKDD is specified, DFSORT deallocates SORTWKnn data sets specified in the JCL and uses dynamic allocation. The USEWKDD run time option in an OPTION control statement in DFSPARM may be specified to disable dynamic allocation and cause DFSORT to use the SORTWKnn data sets specified in the JCL.

When DYNAUTO=YES is specified at installation time, DFSORT dynamically allocates DASD work data sets only when they are not specified in the JCL. If

DYNAUTO=NO is specified at installation time, DFSORT does not use dynamic allocation unless it is requested with the DYNALLOC run time option. DYNAUTO=YES is the IBM-supplied default with DFSORT Release 13.

For a complete explanation of DFSORT dynamic allocation parameters, see the *DFSORT Application Programming Guide* and the *DFSORT Installation and Customization* manual.

Chapter 7. Installation Considerations

In this chapter we discuss the installation options that you have to consider when you install different sort products for a benchmark.

7.1 Program Residency

For production purposes, the sort product should be resident. For sort product comparisons, however, all of the sort products should reside in DASD private libraries (nonresident). If one product is resident and another is nonresident, an inequitable comparison will result. Usually two or more products cannot be made resident simultaneously because of conflicts between the sort products′ module names.

For more information on sort product residency, see 3.5, “Running Resident or Nonresident” on page 10.

7.2 SVC

Many sort products, such as DFSORT, have their own SVC for use in making authorized calls when running in unauthorized mode. These SVCs must be installed properly, so that each sort product has access to its special system routines. If this is not possible or practical, each sort product should run without its SVC, so that the benchmark is not biased toward a particular product.

7.3 System/370-XA Sorting Instructions

The System/370-XA Sorting Instructions are a set of IBM processor instructions designed for some of the fixed-length record sorting algorithms that DFSORT uses. These instructions can reduce processor time when sorting fixed-length records. The sorting instructions are available on all IBM processors, as well as on most competing processors that emulate the IBM architecture.

On some processors, notably older IBM processors and some non-IBM processors, these instructions require a special engineering change (EC) or similar maintenance to enable the sorting instructions. We recommend that you enable the sorting instructions on your system so that DFSORT (and any other sort product that uses them) can take advantage of them.

7.4 Installation Parameters

It is important to ensure that the installation parameters used for each sort product are functionally equivalent. Although each sort product′s installation parameters may have different names, there is much functional similarity among them. Below we discuss how to set the installation parameters for DFSORT.

The defaults supplied with sort products can differ from product to product, and their use could produce a biased performance test. When comparing two sort products, ensure that both have the same characteristics as nearly as possible. Table 1 on page 22 describes some of the significant DFSORT parameters to examine.

Table 1 (Page 1 of 3). Significant DFSORT Installation Parameters. Other sort products may have the same or similar parameters.

Parameter	Description	Possible Settings	IBM-Supplied Value
SIZE	The upper limit for storage. The IEFUSI system installation exit can further limit the actual amount of storage available. Generally speaking, more storage improves performance. Note that too much storage can cause excess paging activity on loaded systems.	A value (in bytes) or MAX (use TMAXLIM)	MAX
TMAXLIM	The upper limit for storage when SIZE=MAX is in effect. The IEFUSI system installation exit can further limit the actual amount of storage available.	A value (in bytes)	4194304 (4MB)
MAXLIM	The upper limit for storage below 16MB virtual when SIZE=MAX is in effect. The JCL REGION value, the OVERRGN value, and the IEALIMIT and IEFUSI system installation exits can further limit the actual amount of storage below 16MB virtual. DFSORT tries to use as much storage above 16MB virtual as possible.	A value (in bytes)	1048576 (1MB)
MINLIM	The smallest value that SIZE can be assigned.	A value (in bytes)	450560 (440KB)
OVERRGN	The amount of storage below 16MB virtual that can be acquired in addition to that specified by the JCL REGION. It is further limited by the region limit of the IEALIMIT and IEFUSI system installation exits.	A value (in bytes)	65536 (64KB)
ODMAXBF	The maximum buffer space DFSORT can use for each OUTFIL data set	A value (in bytes)	2097152 (2MB)
HIPRMAX	The maximum amount of hiperspace that can be used for Hipersorting (MVS/ESA systems only), subject to expanded storage paging activity at the start of the sort. Hipersorting greatly improves elapsed time and I/O performance.	A maximum value (in megabytes) or OPTIMAL (DFSORT determines dynamically the maximum amount of hiperspace to be used for Hipersorting.)	OPTIMAL
EXPMAX	The maximum total amount of available expanded storage to be used at any one time by all Hipersorting applications on an MVS/ESA system	A value between 0 and 8388606 or MAX (DFSORT determines dynamically the total amount of expanded storage available for Hipersorting.)	MAX

Table 1 (Page 2 of 3). Significant DFSORT Installation Parameters. Other sort products may have the same or similar parameters.

Parameter	Description	Possible Settings	IBM-Supplied Value
EXPOLD	The maximum total amount of old expanded storage to be used at any one time by all Hipersorting applications on an MVS/ESA system	A value between 0 and 8388606 or MAX (DFSORT can use up all of the old expanded storage for Hipersorting.)	MAX
EXPRES	The minimum amount of available expanded storage to be reserved for use by non-Hipersorting applications on an MVS/ESA system	A value between 0 and 8388606 or MAX (DFSORT reserves all available expanded storage for use by non-Hipersorting applications, thus disabling Hipersorting.)	0
DSPSIZE	The maximum amount of data space that can be used in dataspace sorting, subject to central storage paging activity at the start and throughout the sort (MVS/ESA systems only) as well as total central storage. Dataspace sorting allows use of more efficient techniques for fixed-length record sorts.	A value (in megabytes) or MAX (as much as necessary)	MAX
VIRTDSP	Allows use of virtual dataspace feature for MVS/XA systems. Virtual dataspace lets MVS/XA systems simulate a data space for use in dataspace sorting.	YES (allow virtual dataspace) or NO (do not allow)	NO
VIO	Allows use of VIO for work data sets. VIO usually degrades or produces unpredictable performance.	YES (allow VIO) or NO (do not allow).	NO
EQUALS	Controls whether the input order of records with equal keys is preserved. EQUALS is always in effect for DFSORT variable-length record sorts. EQUALS degrades performance for fixed-length record sorts.	YES (preserve order) or NO (do not preserve order)	NO
VERIFY	Controls whether the output is checked to verify that it is correctly sorted. VERIFY degrades performance.	YES (check output) or NO (do not check)	NO
DYNAUTO	Controls automatic dynamic allocation of work data sets. Dynamic allocation usually improves performance and eliminates failures associated with insufficient DASD space.	YES (use automatic dynamic allocation, if needed, when no work data sets are specified), IGNWKDD (use automatic dynamic allocation, if needed, even if work data sets are specified), or NO (do not use automatic dynamic allocation)	YES
DYNALOC	Controls the device type and number of dynamic work data sets	A device type (for example, 3390 or SYSDA) and the number of data sets	(SYSDA,4)

Table 1 (Page 3 of 3). Significant DFSORT Installation Parameters. Other sort products may have the same or similar parameters.

Parameter	Description	Possible Settings	IBM-Supplied Value
CFW	Allows use of the cache fast write facility of 3990 cached storage control units to improve elapsed time performance associated with work data set access	YES (allow CFW) or NO (do not allow)	YES
EXCPVR	Allows use of EXCPVR to improve CPU performance associated with page-fixing and page-freeing I/O buffers. EXCPVR can cause excessive page fixing on loaded systems.	ALL (use for all buffers), NOWRK (use for non-work-data-set buffers), or NONE (do not use)	NONE
SDB	Allows use of the system-determined block size facility for choosing the optimum block size for an output data set. SDB improves elapsed time performance.	YES (allow SDB) or NO (do not allow)	YES
WRKREL	Controls whether unused space for work data sets is released. WRKREL degrades performance but improves DASD space management efficiency.	YES (release unused space) or NO (do not release)	YES
OUTREL	Similar to WRKREL, but for the output data sets	See WRKREL settings	YES
WRKSEC	Controls automatic secondary extent allocation of temporary work data sets with no specified secondary extent size. Secondary extents degrades performance. It is better to allocate sufficient storage in the primary extents.	YES (use 25% of primary extent space for secondary extent allocation), a value (use that percentage of primary extent space for the secondary extent allocation), or NO (do not use automatic secondary extent allocation)	YES
OUTSEC	Similar to WRKSEC, except for temporary or new output data sets	See WRKSEC settings	YES
CINV	Allows use of the control interval access method to improve Virtual Storage Access Method (VSAM) performance	YES (allow control interval access) or NO (do not allow)	YES
VSAMBSP	Allows use of more buffers for VSAM processing to improve performance. This option can cause excessive paging on loaded systems when set to anything other than MIN.	MAX (use the maximum number of buffers), OPTIMAL (good compromise between the minimum and the maximum number of buffers), or MIN (use the minimum number of buffers)	OPTIMAL

7.4.1 Equivalent Installation Parameters between Sort Products

When comparing different sort products, you might find some discrepancies among the installation parameters. Some of the parameters mentioned in Table 1 on page 22 might not be controllable during installation of a given sort product. Minor differences in the installation parameters of sort products can be tolerated, but the following DFSORT parameters that have the greatest effect on sort performance should be matched where possible:

- Virtual storage (SIZE, TMAXLIM, MAXLIM, MINLIM, OVERRGN, VIRTDSP, ODMAXBF and any others that affect the size of the primary address space region)
- Hiperspace (HIPRMAX, EXPMAX, EXPOLD, EXPRES)
- Data space (DSPSIZE)
- Dynamic allocation (DYNAUTO and DYNALOC)
- VIO (VIO)
- EXCPVR (EXCPVR)
- Cache fast write (CFW)
- System-determined block size (SDB).

Some sort products, such as DFSORT, provide extensive capabilities to specify run time options as well as override installation options at run time. You must not override installation options or specify run time options that prevent a sort product from running its most efficient technique. For example, do not specify the NOBLKSET option in DFSORT because it prevents DFSORT from using Blockset, its most efficient technique.

It is also important that you include in the benchmark sort functions and options used in the production environment. Functions such as exits, record-level editing, record filtering, record summarization, and reports can greatly affect the performance of a sort product.

7.4.2 Optimum Mix of Sort Parameters: Benchmark and Production

It would be ideal if sort benchmarks could be run under exactly the same conditions as they would if they were real jobs, competing for the same resources for which they would have to compete in a production environment. Unfortunately, this is both difficult to simulate and difficult to measure consistently. Instead, we usually rely on special benchmarks designed to be run serially on less than fully loaded systems.

The values of a number of the installation parameters discussed above might be set to values for a valid benchmark that differ from the values used for a production environment. Overall system performance as well as individual job performance must be considered for a fully effective production environment evaluation. With few exceptions (most notably, VIRTDSP and WRKREL), the IBM-supplied values for the DFSORT installation parameters produce the best individual job performance. Some DFSORT parameters that might need to be changed for a production environment include:

- Virtual storage parameters such as TMAXLIM and MAXLIM—These can be lowered on heavily loaded systems to reduce the impact of DFSORT on central storage paging activity. They can also be raised for lightly loaded

systems or systems with a substantial number of large sorts to improve overall sort performance.

- HIPRMAX, EXPMAX, EXPOLD, and EXPRES—Use HIPRMAX=OPTIMAL to allow DFSORT to control the amount of hiperspace through the EXPMAX, EXPOLD, and EXPRES parameters. In an environment where the expanded storage is heavily used you can control the overall amount of expanded storage used for Hipersorting and the amount of expanded storage to be reserved for non-sort applications with the EXPMAX, EXPOLD, and EXPRES parameters.
- EQUALS and VERIFY—These may be defaults for a particular site.
- DYNAUTO—Use DYNAUTO=YES to realize automatic dynamic allocation of work data sets. On MVS/ESA systems, use DYNAUTO=IGNWKDD to suppress user-specified DASD work data set allocations in Hipersorting situations.
- VIO—The default should be changed to VIO=YES only if such a change improves overall system performance.
- DYNALOC—The default device type for dynamically allocated work data sets should be changed from SYSDA if there is a more appropriate device type at your site.
- EXCPVR—The default should be changed to EXCPVR=ALL or EXCPVR=NOWRK if the system paging activity at your sie is light or moderate.

Chapter 8. How to Run a Sort Benchmark

Now that you understand how to select sort benchmark jobs, which trade-offs are important for the sort product′s use of virtual storage, how to allocate input, output, and work data sets, and which installation decisions need to be made, it is time to answer the question, How do I run a sort benchmark?

The three most common problems that occur when conducting performance comparisons between sort products are:

- Nonrepeatable results
- Statistical bias
- Lack of planning to determine which data will be measured and how the measurements will be obtained.

The sections that follow will help you avoid such problems.

8.1 Obtaining Repeatable Results

The single most important item when running any performance benchmark is the ability to repeat the results. Results that are not repeatable can invalidate a benchmark or, even worse, lead to misinterpretations of the results and erroneous conclusions. For sort jobs, elapsed time is very susceptible to large variations, CPU time is moderately susceptible, and EXCP counts have very low variation. Careful consideration of the following items will tend to minimize the variability of sort benchmark measurements:

- Avoid device contention. Be certain that:
  - No other application is accessing the devices used by the sort product, especially the work data set devices.
  - The devices are not shared with another system. .
- Avoid channel and control unit contention.

  Try to ensure that the control unit and channel are dedicated to the string of devices used for sorts. This is very important for the work data set devices.
- Run in a controlled environment.
  - The optimal case is to run in stand-alone mode with a single initiator so only one job runs at a time. If this is not possible, run the benchmark during periods of very low system activity.
  - If your benchmark includes workload tests for multiple concurrent sorts or other system activity, prepare a set of jobs that will run identically each time.
  - While the benchmark jobs are running, stop all other manual activity, such as file editing, other job submissions, and frequent interrogations on the operator′s console.
- Run the comparisons in sets.

  If you are comparing different sort products over a number of different jobs, run the first job for each sort product, followed by the second job for each sort product, and so on. This approach minimizes variations

over time, especially if a benchmark must be interrupted and resumed at some time later.

- Do not allocate and deallocate data sets as part of the sort step.
  - Create the input data sets first and use them for all sort steps.
  - Allocate the work (if dynamic allocation is not used) and output data sets in a dummy job step (that is, an IEFBR14 call) before each sort step.
  - Delete the work and output data sets in another dummy job step right after each sort step.

8.2 Avoiding Statistical Bias

Statistical bias is difficult to detect when comparing different products. Before starting the benchmark, make sure that your hardware configuration is optimal for the selected sort jobs and your sort installation decisions do not favor any particular sort product. Consult the checklist in Appendix A, “A Sort Benchmark Checklist” on page 31 for further guidance in avoiding statistical bias.

8.3 Planning for Data Collection and Measuring

You should develop a plan to determine which data you will measure and how you will collect the measurement information.

The data most commonly collected for sort benchmarks includes CPU time, elapsed time, device connect time, channel busy time, and EXCP counts. System management facility (SMF) type 30 (subtype 4) records provide CPU time and elapsed time along with EXCP counts and device connect times for each data set. You can collect channel and device utilization data by using the Resource Measurement Facility (RMF). To get information on each sort job it is best to start and stop RMF before and after each job in the benchmark. You can use an MVS independent domain to control resources allocated to the benchmark and collect workload information from RMF.

8.3.1 Measuring Elapsed Time

Elapsed time is the amount of “wall clock” time from job initiation to job termination. It is best collected by using SMF type 30 records. Elapsed time is important in all shops and is critical when the workload distribution creates limited batch windows.

To obtain valid measurements of elapsed time performance, it is preferable to run the benchmark jobs in stand-alone mode with a single initiator. Running in stand-alone mode avoids elapsed time measurement bias that results from device contention and CPU swapping.

If it is not possible to run the benchmark jobs in stand-alone mode, you should run them on a lightly loaded system, possibly off shift. It is important to run the same job using each sort product, one after the other, so that you can measure elapsed time for both products in the same environment.

8.3.2 Measuring CPU Time

You can collect CPU time for each job through SMF and, supplementally, through RMF. On MVS/ESA systems, SMF breaks down CPU into five fields:

- Task control block (TCB) time. This is the dominant component in a sort job.
- Service request block (SRB) time. This is a fairly small percentage of the total CPU time.
- Region control task (RCT) time. This is not a significant component of the total CPU time.
- I/O interrupt processing (IIP) time. This is not a significant component of the total CPU time.
- Hiperspace processing time (HPT). This component is significant if you are using hiperspaces (such as with Hipersorting).

You can use these fields for capacity planning, performance management, or resource billing. You should consider including all five fields as an aggregate sum in your total CPU time measurement.

For MVS/XA systems, only the first two fields (TCB and SRB) are accounted for. CPU times for this environment should be the sum of the two fields.

Accurate SMF CPU time requires that jobs spending a significant amount of their time running in supervisor state post their CPU time to SMF. SMF automatically receives CPU consumption data when a job is in problem program state. It is the responsibility of a program running authorized, however, to post its CPU time by means of SMF exits. DFSORT performs all sort activity in problem program state, so the SMF type 30 (subtype 4) records accurately reflect DFSORT CPU consumption.

RMF takes samplings of all system activity (including CPU time by job) and shows CPU time that a sort job might not have reported to SMF. The RMF Monitor II Address Space Resource Data report gives CPU TCB time for selected jobs (see *MVS/ESA Analyzing RMF Version 4 Monitor I & II Reports*). You can compare this time with the SMF TCB time to identify the percentage of CPU time not captured for the sort job.

Appendix A. A Sort Benchmark Checklist

This appendix summarizes all of the factors that you should consider when planning and running a sort benchmark.

Table 2 provides a checklist of the guidelines, hints, and tips that you should consider in your sort benchmark planning. If you answer "NO" to any question in the table, please reread the appropriate chapter or section in this book for further clarification. If cannot answer "YES" to all of the questions (and that is certainly possible), at least you will be aware of the factors that will decrease the validity of your benchmark.

Table 2 (Page 1 of 3). Sort Benchmark Factors Checklist

Factors to Be Considered	**Have They Been Considered?**	
	YES	**NO**
Do the selected sort jobs have one or more of the following characteristics? (See Chapter 2, "Sort Benchmark: Job Selection" on page 3.)		
Are they long running (that is, must they fit into a "batch window")?		
Are they run frequently?		
Are they part of critical applications?		
Do they use large amounts of CPU time?		
Do they use large amounts of I/O?		
Are they invoked from other applications (such as COBOL)?		
Do they use copy or merge functions?		
Do they use functions commonly used a in production environment (such as exits, record level editing, record filtering, record summarization, and reports)?		
Do the selected sort jobs have sort parameters that represent production jobs? (See Chapter 2, "Sort Benchmark: Job Selection" on page 3.)		
Type of input data set (VSAM, striped, compressed, pipe)		
Record length		
Sort key length		
Record format (fixed, variable, spanned)		
Data set size (number of records, number of bytes)		
Randomness of data (whether the data is presequenced)		
Virtual storage size		
Amount of hiperspace		
Amount of data space		
JCL REGION size		
Input and output devices (DASD or tape)		
Number and location of work data sets		
Functions and options (such as exits, INREC, OUTREC, INCLUDE, OMIT, SUM, OUTFIL)		

Table 2 (Page 2 of 3). Sort Benchmark Factors Checklist

Factors to Be Considered	Have They Been Considered?	
	YES	NO
Is the system configuration representative of the production environment? (See Chapter 2, "Sort Benchmark: Job Selection" on page 3.)		
CPU model		
Amount of central storage available		
Amount of expanded storage available		
Configuration of DASD, storage control units, and channels		
Operating system release level (MVS/ESA or MVS/XA)		
Operating system maintenance level		
Input, output, and work data set allocation questions (See Chapter 6, "Allocation of Input, Output, and Work Data Sets" on page 17.)		
Is there access to equivalent number of work data sets?		
Are there several work data sets?		
Are the work data sets on separate devices?		
Are the storage devices not shared with other systems?		
Are input and output data sets not on the same channel or device as any work data set?		
Are all data sets (especially work data sets) on fast devices (such as RAMACs or 3390-3)?		
Are the work data sets on devices controlled by a cached 3990?		
Are input and output devices on a separate control unit from work devices?		
Is dynamic allocation being used for work devices?		
Sort product installation decisions (See Chapter 7, "Installation Considerations" on page 21.)		
Is the sort product installed nonresident?		
Is the SVC installed?		
Are the sorting instructions enabled?		
Is there equivalent total virtual storage available (are the storage parameters such as SIZE and TMAXLIM set consistently)?		
Is there equivalent virtual storage available below 16MB (are the storage parameters such as MAXLIM and OVERRGN set consistently)?		
Is there equivalent hiperspace available (MVS/ESA systems)?		
Is there equivalent data space available?		
Is the virtual dataspace feature enabled (MVS/XA systems)?		
Is VIO=NO in effect?		
Is EQUALS set consistently?		
Is VERIFY set consistently?		
Are dynamic allocation defaults (DYNALOC) set consistently?		
Is cache fast write (CFW) allowed?		

Table 2 (Page 3 of 3). Sort Benchmark Factors Checklist

Factors to Be Considered	Have They Been Considered?	
	YES	NO
Is EXCPVR allowed?		
Is system-determined block size (SDB) allowed?		
Is WRKREL/OUTREL=NO in effect?		
Is WRKSEC/OUTSEC=NO in effect?		
Is CINV=YES in effect?		
Is VSAMBSP set to OPTIMAL or MAX?		
Is the NOBLKSET option not set for DFSORT?		
Obtaining repeatable results questions (See 8.1, "Obtaining Repeatable Results" on page 27.)		
Are there no other applications accessing work devices?		
Are there no other applications accessing input and output device(s)?		
Are the jobs being run with minimal system activity?		
Are the comparisons run in sets?		
Are the input data sets created before any sort steps?		
Are the work and output data sets preallocated before the sort step?		
Are the work and output data sets deallocated after the sort step?		
Is the optimal system configuration for sorting in place?		
Are the sort installation decisions identical (where possible)?		
What data is to be collected? (See 8.3, "Planning for Data Collection and Measuring" on page 28.)		
Elapsed time?		
CPU time?		
Channel busy time?		
Device connect time?		
EXCP counts?		

List of Abbreviations

CFW	cache fast write
CPU	central processing unit
DASD	direct access storage device
DFSORT	Data Facility Sort
EC	engineering change
ESA/370	Enterprise Systems Architecture/System 370
ESA/390	Enterprise Systems Architecture/System 390
EXCP	execute channel program
EXCPVR	execute channel program virtual request
IBM	International Business Machines Corporation
IPL	initial program load
ITSO	International Technical Support Organization
JCL	job control language
LPA	link pack area
MVS/ESA	Multiple Virtual Storage/Enterprise Systems Architecture
MVS/XA	Multiple Virtual Storage/Extended Architecture
RMF	Resource Management Facility
SDB	system-determined block size
SMF	System Management Facility
SVC	supervisor call
VSAM	Virtual Storage Access Method

ITSO Technical Bulletin Evaluation RED000

International Technical Support Organization
DFSORT Release 13 Benchmark Guide
May 1995

Publication No. GG24-4476-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction	____		
Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries:

Do you provide billable services for 20% or more of your time? Yes____ No____

Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____

If no, please explain:

_____

_____

What other topics would you like to see in this Bulletin?

_____

_____

What other Technical Bulletins would you like to see published?

_____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

_____ _____
Name Address

_____ _____
Company or Organization

_____ _____
Phone No.

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

IBM International Technical Support Organization
Department 471, Building 070B
5600 COTTLE ROAD
SAN JOSE CA
USA 95193-0001

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

Cut or Fold
Along Line

Printed in U.S.A.