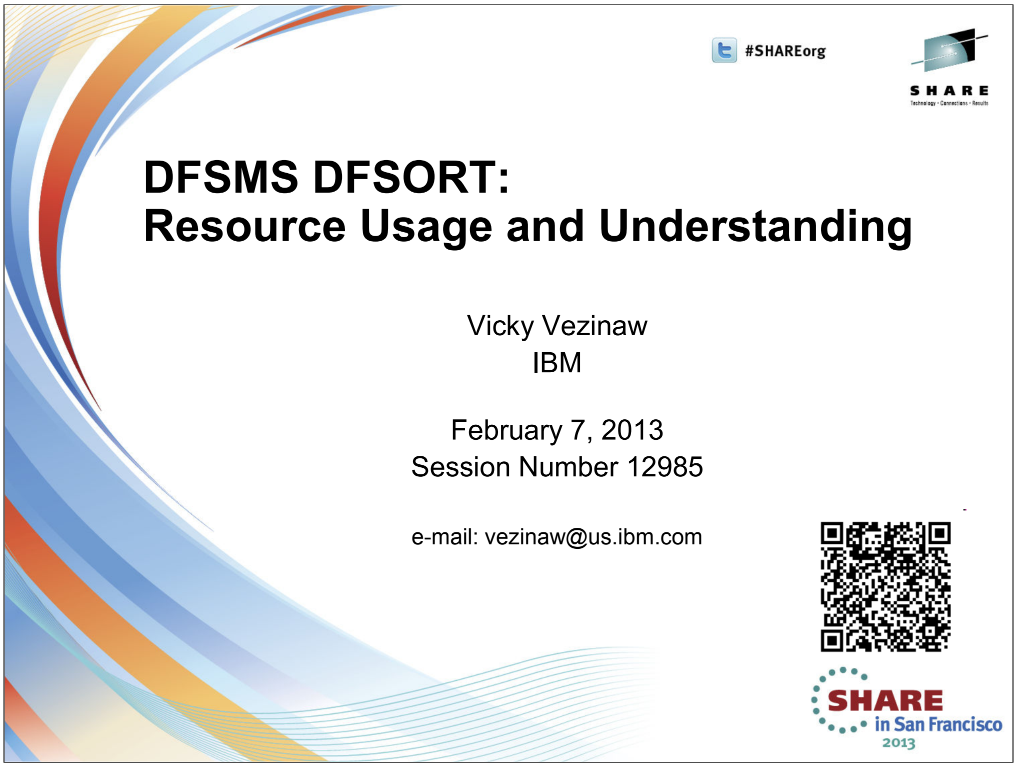


#SHAREorg

SHARE
Technology · Connections · Results

DFSMS DFSORT: Resource Usage and Understanding

Vicky Vezinaw
IBM

February 7, 2013
Session Number 12985

e-mail: vezinaw@us.ibm.com

SHARE
in San Francisco
2013

Trademarks

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

AIX*
BladeCenter*
BookManager*
CICS*
DataPower*
DB2*
DFSMS
DFSMSdss
DFSMShsm
DFSMSrmm
DFSORT
Domino*
DS6000
DS8000*
FICON*
IBM*
IBM eServer
IBM logo*
IMS
InfinBand
Language Environment*
MVS
Parallel Sysplex*
ProductPac*
RACF*
Redbooks*
REXX
RMF
ServerPac*
SYSREXX
System Storage
System x*
System z
System z9
System z10
System z10 Business Class
Tivoli*
WebSphere*
z10
z10 BC
z10 EC
zEnterprise*
zSeries*

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Windows Server and the Windows logo are trademarks of the Microsoft group of countries.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

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

Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

* Other product and service names might be trademarks of IBM or other companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

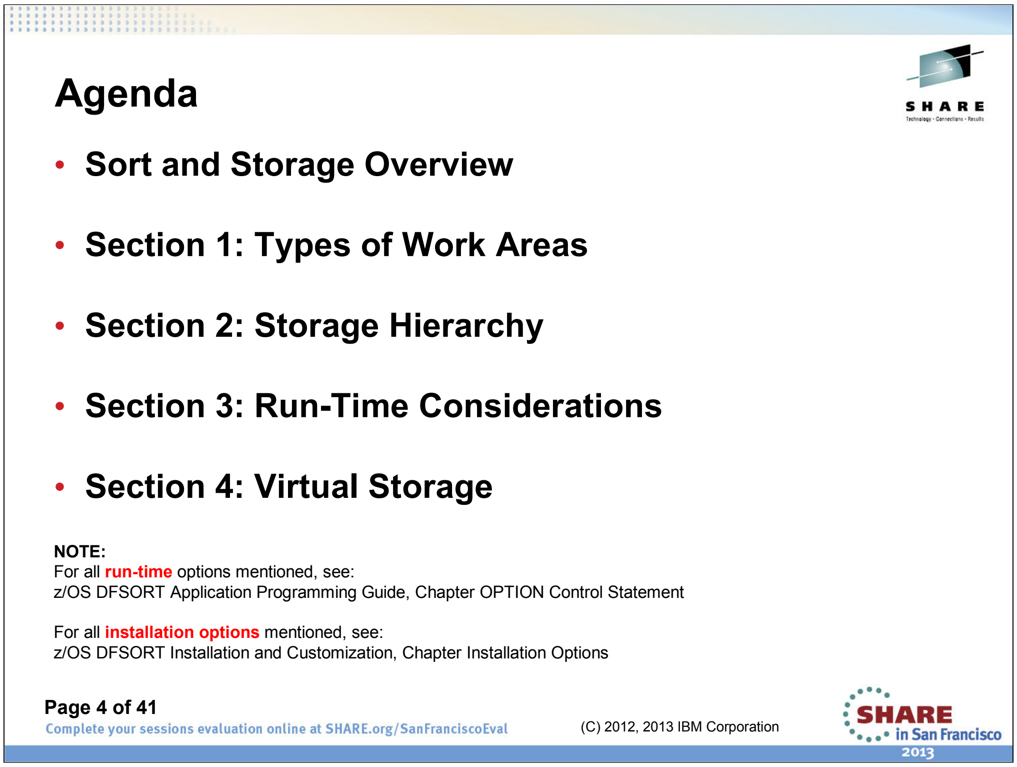
Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

This information provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g, zIIPs, zAAPs, and IFLs) ("SEs"). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the “Authorized Use Table for IBM Machines” provided at www.ibm.com/systems/support/machine_warranties/machine_code/aut.html (“AUT”). No other workload processing is authorized for execution on an SE. IBM offers SE at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.

System z Social Media Channels

- Top Facebook pages related to System z:
  - IBM System z
  - IBM Academic Initiative System z
  - IBM Master the Mainframe Contest
  - IBM Destination z
  - Millennial Mainframer
  - IBM Smarter Computing
- Top LinkedIn groups related to System z:
  - System z Advocates
  - SAP on System z
  - IBM Mainframe- Unofficial Group
  - IBM System z Events
  - Mainframe Experts Network
  - System z Linux
  - Enterprise Systems
  - Mainframe Security Gurus
- Twitter profiles related to System z:
  - IBM System z
  - IBM System z Events
  - IBM DB2 on System z
  - Millennial Mainframer
  - Destination z
  - IBM Smarter Computing
- YouTube accounts related to System z:
  - IBM System z
  - Destination z
  - IBM Smarter Computing
- Top System z blogs to check out:
  - Mainframe Insights
  - Smarter Computing
  - Millennial Mainframer
  - Mainframe & Hybrid Computing
  - The Mainframe Blog
  - Mainframe Watch Belgium
  - Mainframe Update
  - Enterprise Systems Media Blog
  - Dancing Dinosaur
  - DB2 for z/OS
  - IBM Destination z
  - DB2utor



Agenda

- Sort and Storage Overview
- Section 1: Types of Work Areas
- Section 2: Storage Hierarchy
- Section 3: Run-Time Considerations
- Section 4: Virtual Storage

NOTE:
For all **run-time** options mentioned, see:
z/OS DFSORT Application Programming Guide, Chapter OPTION Control Statement

For all **installation options** mentioned, see:
z/OS DFSORT Installation and Customization, Chapter Installation Options

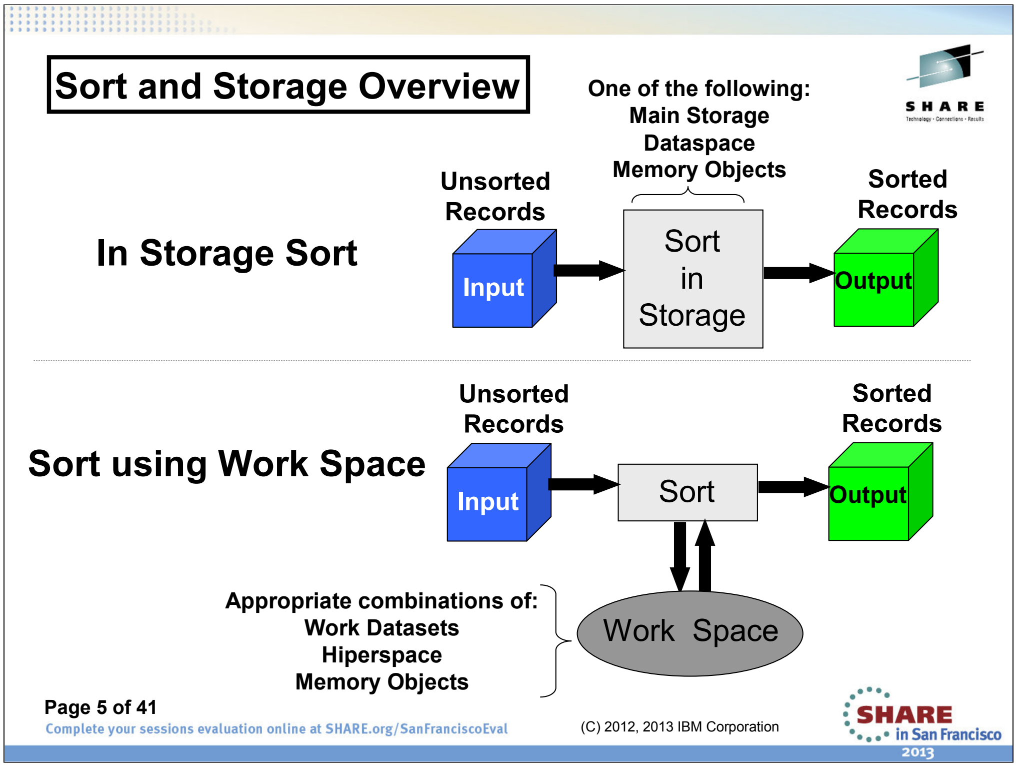
This presentation will focus on sorting and will not have significant information related to merging or copying with DFSORT

- **Types of Work Areas**
  - Sort and Storage Overview
  - Main Storage
  - Intermediate Work datasets
  - Dataspace
  - Hiperspace
  - Memory Objects
  - Central Storage Installation Options

- **Storage Hierarchy**
  - Processor cache
  - Central storage
  - Storage control cache
  - Disk
  - Tape

- **Run-Time Considerations**
  - Selection of Storage Type to be Used
  - General Characteristics of using Dataspace, Hiperspace, and Memory Objects

- **Virtual Storage**
  - Data Set Size and Virtual Storage
  - Virtual Storage Limitations



Sort and Storage Overview

In Storage Sort

Unsorted Records

Input

One of the following:
Main Storage
Dataspace
Memory Objects

Sort in Storage

Sorted Records

Output

Sort using Work Space

Unsorted Records

Input

Sort

Sorted Records

Output

Work Space

Appropriate combinations of:
Work Datasets
Hiperspace
Memory Objects

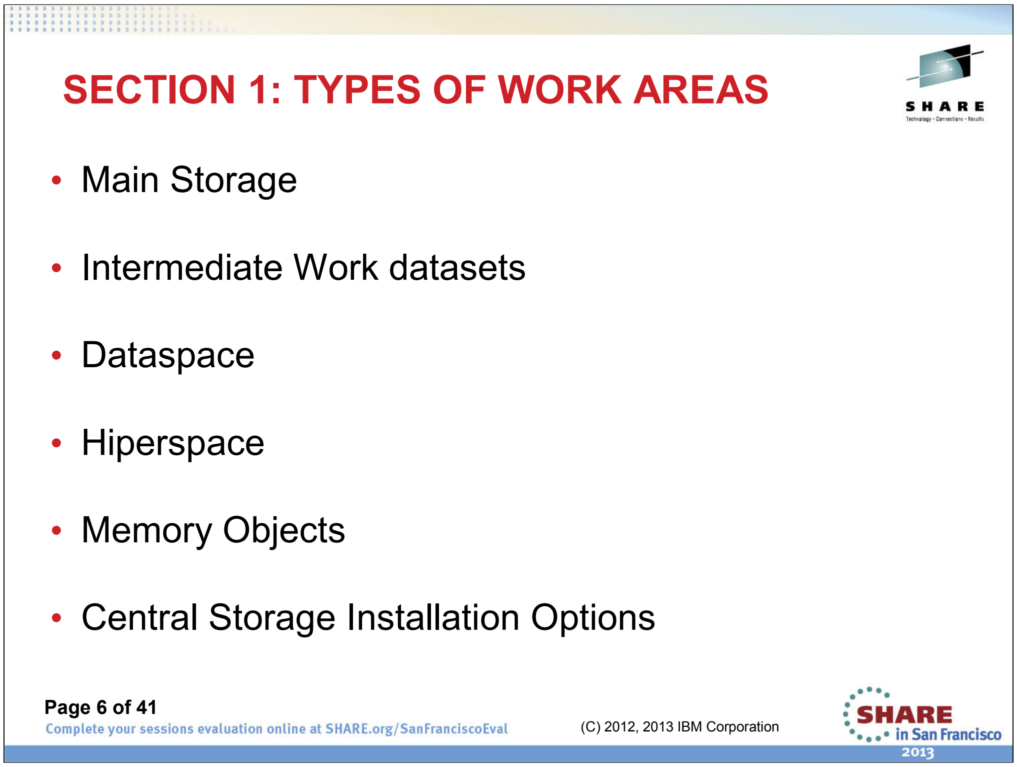
Records are sorted and use storage in one of two ways:

In Storage Sorts: Records are sorted in place and then written to output. One of the following is used for storage:

- Main Storage
- Dataspace
- Memory Objects

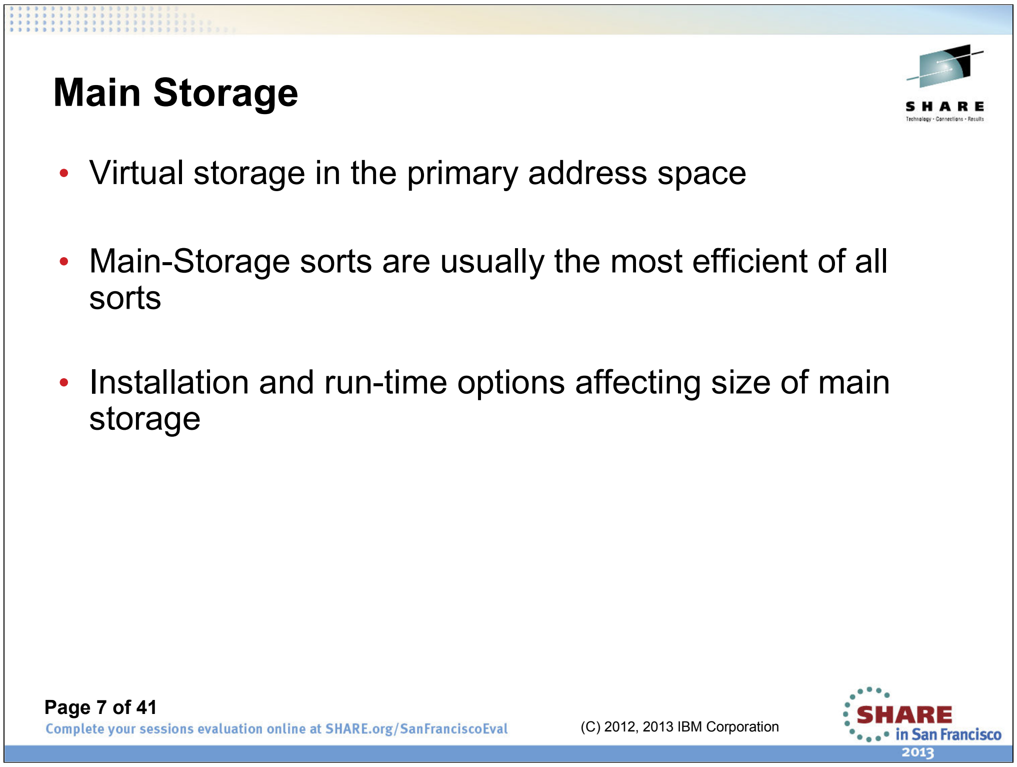
Sort using Work Space: When a sort application cannot be performed entirely in virtual storage, DFSORT must use work space. Records are sorted into groups of sorted records written to intermediate work space and then later those groups are merged and written to output. One of the following is used:

- Disk (allocated in JCL or dynamically allocated by DFSORT) and/or
- Hiperspace or Memory Objects



SECTION 1: TYPES OF WORK AREAS

- Main Storage
- Intermediate Work datasets
- Dataspace
- Hiperspace
- Memory Objects
- Central Storage Installation Options



Main Storage

- Virtual storage in the primary address space
- Main-Storage sorts are usually the most efficient of all sorts
- Installation and run-time options affecting size of main storage

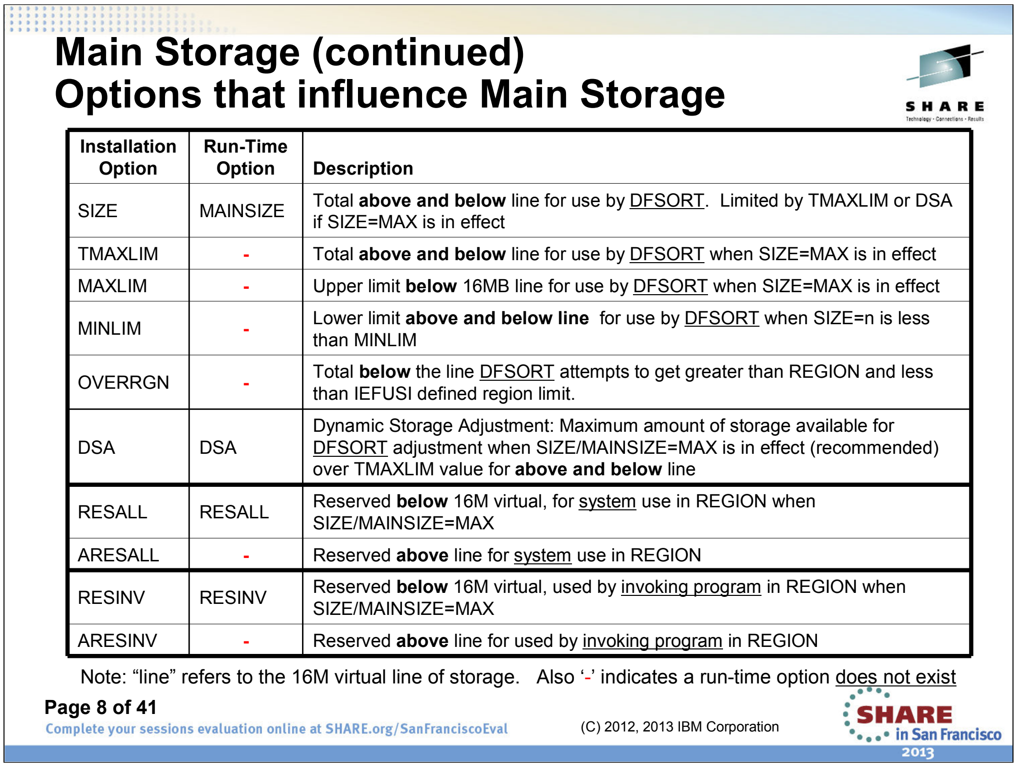
Sorting is a memory-intensive operation and main storage is the most crucial of all DFSORT resources

Virtual storage in the primary address space
Used by both program code and data.

Main Storage	Comments
88KB	Minimal sort may run
128-440KB	A more reasonable minimum
4M or greater	Recommended
6M	Shipped installation default

Main-Storage sorts are usually the most efficient of all sorts
Sorts done entirely in main storage will usually have the best performance. No need for intermediate work storage because we are not moving data out to a work area. Data is read into Main, sorted, then read back out.

Installation and run-time options affecting size of main storage
DFSORT has several options that can be used to control use of storage. It is recommended that installation defaults be set using DFSORT Parmlib support. Since resource usage varies during the day, you may want to use DFSORT's “time of day” feature to set 4 sets of installation defaults. For an overview of the separate sets of installation defaults, see z/OS DFSORT Installation and Customization, Chapter Customizing DFSORT, Section Changing the Installation Defaults.



Main Storage (continued)
Options that influence Main Storage

Installation Option	Run-Time Option	Description
SIZE	MAINSIZE	Total **above and below** line for use by DFSORT. Limited by TMAXLIM or DSA if SIZE=MAX is in effect
TMAXLIM	-	Total **above and below** line for use by DFSORT when SIZE=MAX is in effect
MAXLIM	-	Upper limit **below** 16MB line for use by DFSORT when SIZE=MAX is in effect
MINLIM	-	Lower limit **above and below line** for use by DFSORT when SIZE=n is less than MINLIM
OVERRGN	-	Total **below** the line DFSORT attempts to get greater than REGION and less than IEFUSI defined region limit.
DSA	DSA	Dynamic Storage Adjustment: Maximum amount of storage available for DFSORT adjustment when SIZE/MAINSIZE=MAX is in effect (recommended) over TMAXLIM value for **above and below** line
RESALL	RESALL	Reserved **below** 16M virtual, for system use in REGION when SIZE/MAINSIZE=MAX
ARESALL	-	Reserved **above** line for system use in REGION
RESINV	RESINV	Reserved **below** 16M virtual, used by invoking program in REGION when SIZE/MAINSIZE=MAX
ARESINV	-	Reserved **above** line for used by invoking program in REGION

Note: “line” refers to the 16M virtual line of storage. Also ‘-’ indicates a run-time option does not exist

Complete your sessions evaluation online at SHARE.org/SanFranciscoEval

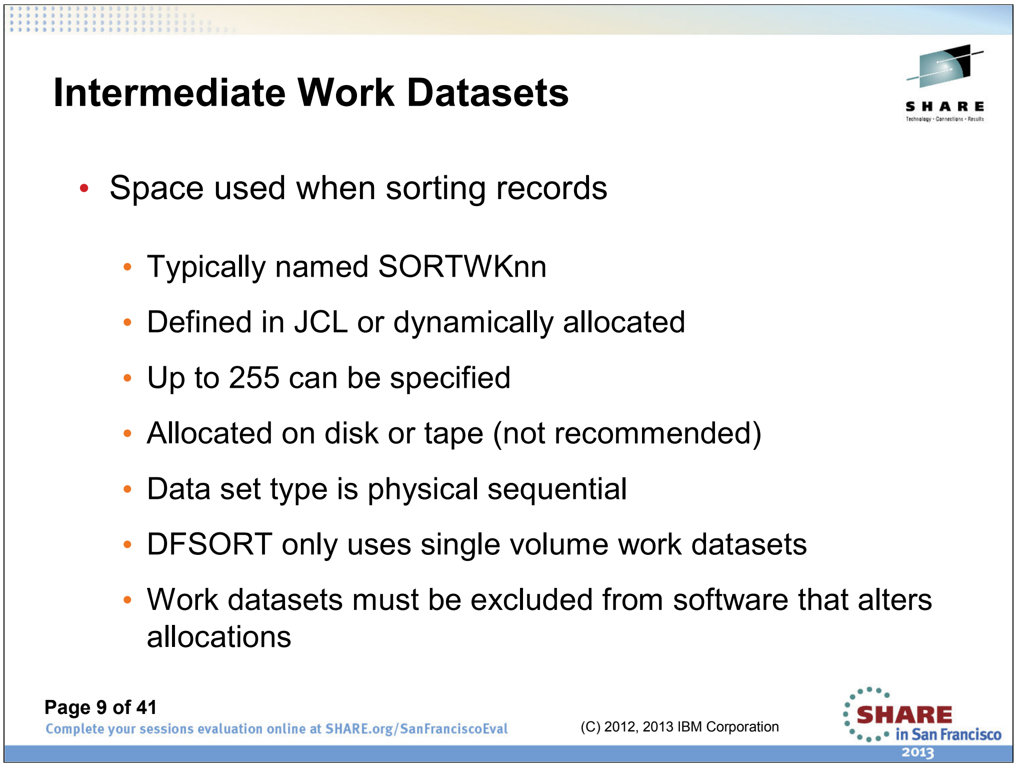
Here is a summary of installation and run-time options that control how much storage is available to DFSORT and reserved for system and invoking program’s use.

It is recommended that the optimal mainsize size be selected for DFSORT by using DFSORT’s DSA (Dynamic Storage Adjustment) option. Will only adjust to above TMAXLIM if needed to improve performance up to the DSA value.

For detailed information about the storage required to run DFSORT, see:
z/OS DFSORT Installation and Customization,
Chapter Planning for Installation,
Section System Planning Considerations,
Subsection DFSORT Storage Considerations

For a description of the installation options listed in the above table see:
z/OS DFSORT Installation and Customization
Chapter Customizing DFSORT,
Section Changing the Installation Defaults,
Subsection Installation Options

For a description of the run-time options, see
z/OS DFSORT Application Programming Guide,
Chapter Using DFSORT Program Control Statement,
Section OPTION Control Statement.



Intermediate Work Datasets

- Space used when sorting records
  - Typically named SORTWKnn
  - Defined in JCL or dynamically allocated
  - Up to 255 can be specified
  - Allocated on disk or tape (not recommended)
  - Data set type is physical sequential
  - DFSORT only uses single volume work datasets
  - Work datasets must be excluded from software that alters allocations

Not used for MERGE or COPY.

Can be used alone or along with dataspace, hiperspace or memory object space.

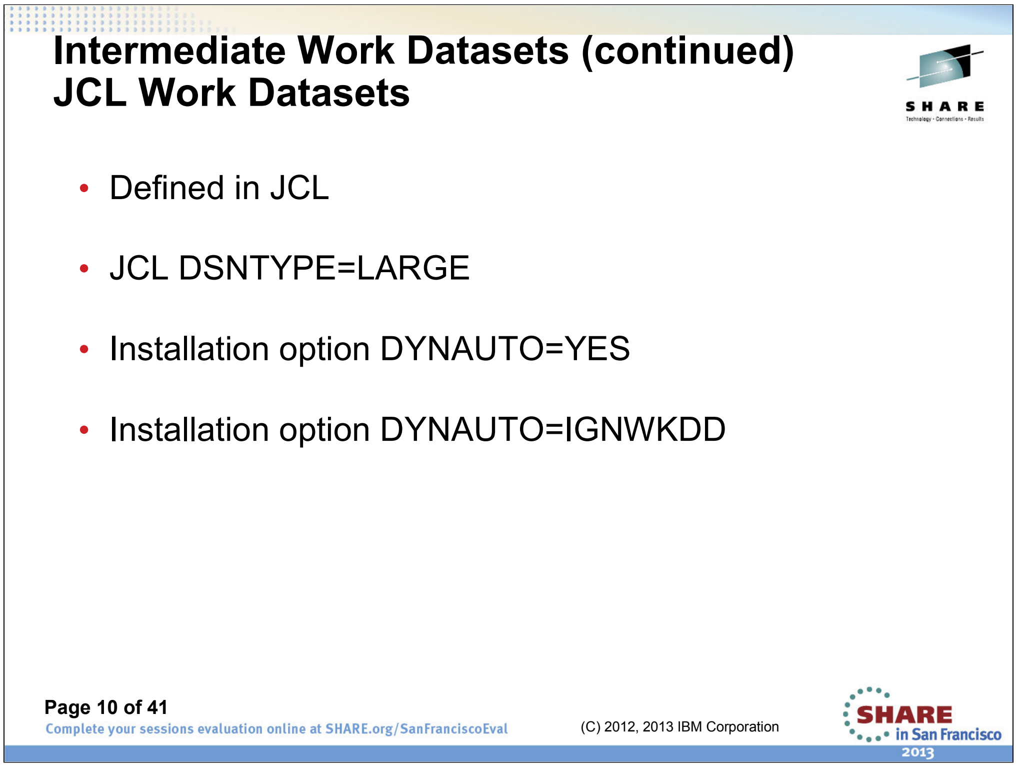
Should have unique name. Duplicates are not used. Can be renamed using run-time OPTION SORTDD (usually used when invoking DFSORT multiple times in the same address space). Enables you to use uniquely named work datasets for each invocation. If this is not done for this situation the work data sets are re-used and is not recommended.

Tape is NOT Recommended as this is a big performance hit due to using slow work space device and also using VERY OLD sorting Conventional Technique.

Not partitioned or extended format because DFSORT exclusively uses EXCPs to read and write work data sets.

Multi-volume work data sets can be allocated, but DFSORT will only use the first volume.

OEM software that changes DFSORT’s allocations can alter the JOB’s TIOT and result in an abendU1095. To avoid this situation, you should take the appropriate steps to have such software exclude these ddnames from those eligible for space reductions (SORTWK*, STATWK*, DATAWK*, DAnnWK*, STnnWK*, SWnnWK*).



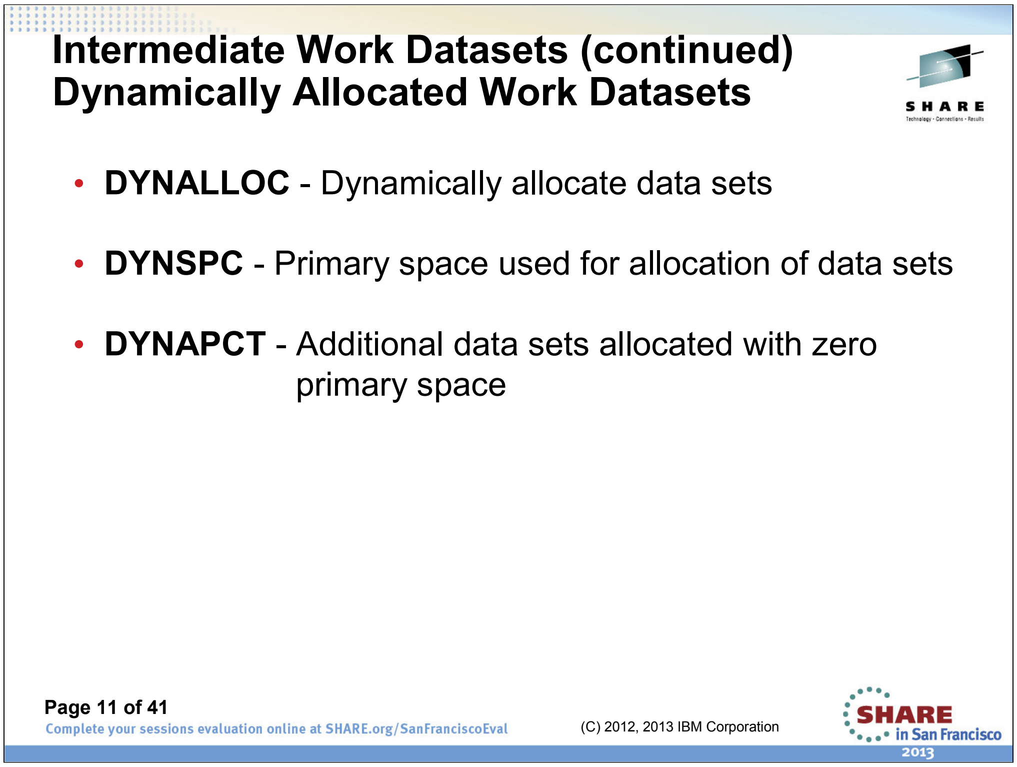
Intermediate Work Datasets (continued)
JCL Work Datasets

- Defined in JCL
- JCL DSNTYPE=LARGE
- Installation option DYNAUTO=YES
- Installation option DYNAUTO=IGNWKDD

When DSNTYPE=LARGE coded, large format is used. Capable of using greater than 65,536 tracks

If installation default DYNAUTO=YES is used, coding JCL work data sets does not allow dynamic allocation

Installation option DYNAUTO=IGNWKDD specifies that work data sets should be dynamically allocated using the DYNALLOC/DYNALOC values in effect even if SORTWKdd DD statements are coded in JCL



Intermediate Work Datasets (continued)
Dynamically Allocated Work Datasets

- **DYNALLOC** - Dynamically allocate data sets
- **DYNSPC** - Primary space used for allocation of data sets
- **DYNAPCT** - Additional data sets allocated with zero primary space

Complete your sessions evaluation online at SHARE.org/SanFranciscoEval

All 3 are installation or run-time options but note the DYNALOC installation option is spelled slightly different.

DYNALLOC

- Not used if work data sets are coded in JCL
- Automatically uses large format

DYNSPC

- Only used by DFSORT if input file size is unknown
- 20% of primary space is used as secondary space allocation
- Installation default is 256M bytes. Approximately (Note the actual amount needed varies with the DFSORT functions used):
  - 1200M bytes can be sorted with DYNSPC=256
  - 150M bytes can be sorted with DYNSPC=32

DYNAPCT was added in DFSORT release V1R12. Specifies additional work data sets to be dynamically allocated with zero primary space. DFSORT only extends these data sets when necessary to complete a sort application. For example the following specified at run-time will allocated a total of 22 intermediate work datasets (10% of 20 + 20=total of 22).

```
 OPTION DYNAPCT=10,DYNALLOC=(SYSDA,20)
```

Dataspace Description

- A Dataspace is a large contiguous piece of virtual storage
- Maximum size is 2 gigabytes
- Storage used similarly to how ‘main storage only’ would be used to sort records
- No programs execute in Dataspace

Dataspace Control

DSPSIZE=[MAX | n]
installation option or run-time option specifies the maximum amount of dataspace that can be used with data space sorting for dataspace sorting.

- **MAX** (default)
  - lets DFSORT determine maximum amount of dataspace to be used

- **n**
  - Specifies the maximum in megabytes that can be used
  - n must be a value between 0 and 9999. Note even though you can specify more than 2000, DFSORT will limit the largest amount of dataspace to 2G.
  - 0 (zero) specifies dataspace will not be used

Hiperspace Description

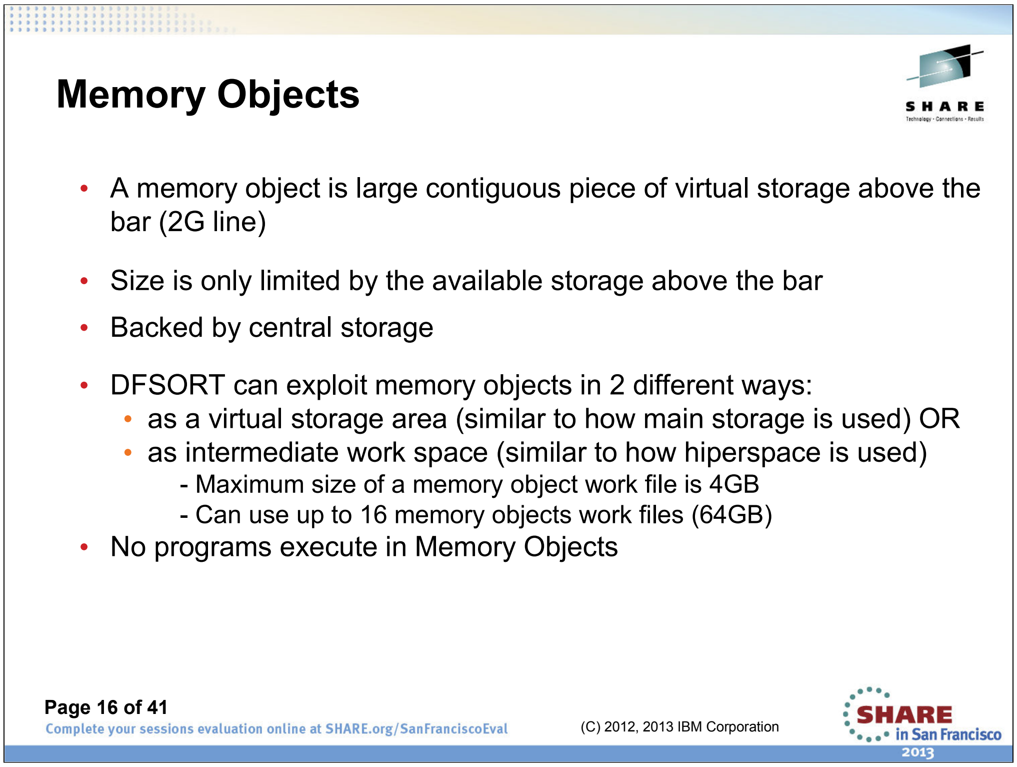
- Large contiguous area of virtual storage
- Maximum size of one hiperspace is 2 gigabytes
- 16 Hiperspaces, total of 32 gigabytes, can be used for single sort
- Data only, no programs execute in Hiperspace
- Hiperspace is used similarly to how to work data sets are used with the difference that using hiperspace is quicker than using disk work data

Hiperspace Control

HIPRMAX=[OPTIMAL | n | p%]

Installation option or run-time option specifies maximum amount of hiperspace to be used with hiperspace sorting application

- **OPTIMAL** (Default)
  - Lets DFSORT determine maximum amount of hiperspace to be used.
- **n**
  - Specifies the maximum in megabytes that can be used.
  - must be a value between 0 and 32767
  - 0 (zero) specifies hiperspace will not be used
- **p%**
  - p percent (of an appropriate portion of central storage) amount of storage to be used for hipersorting
  - p must be a value between 0 and 100



Memory Objects

- A memory object is large contiguous piece of virtual storage above the bar (2G line)
- Size is only limited by the available storage above the bar
- Backed by central storage
- DFSORT can exploit memory objects in 2 different ways:
  - as a virtual storage area (similar to how main storage is used) OR
  - as intermediate work space (similar to how hiperspace is used)
    - Maximum size of a memory object work file is 4GB
    - Can use up to 16 memory objects work files (64GB)
- No programs execute in Memory Objects

Capability to use memory objects as workspace was added:

DFSORT V1R12 2010

APAR PM12281

PTF UK58148

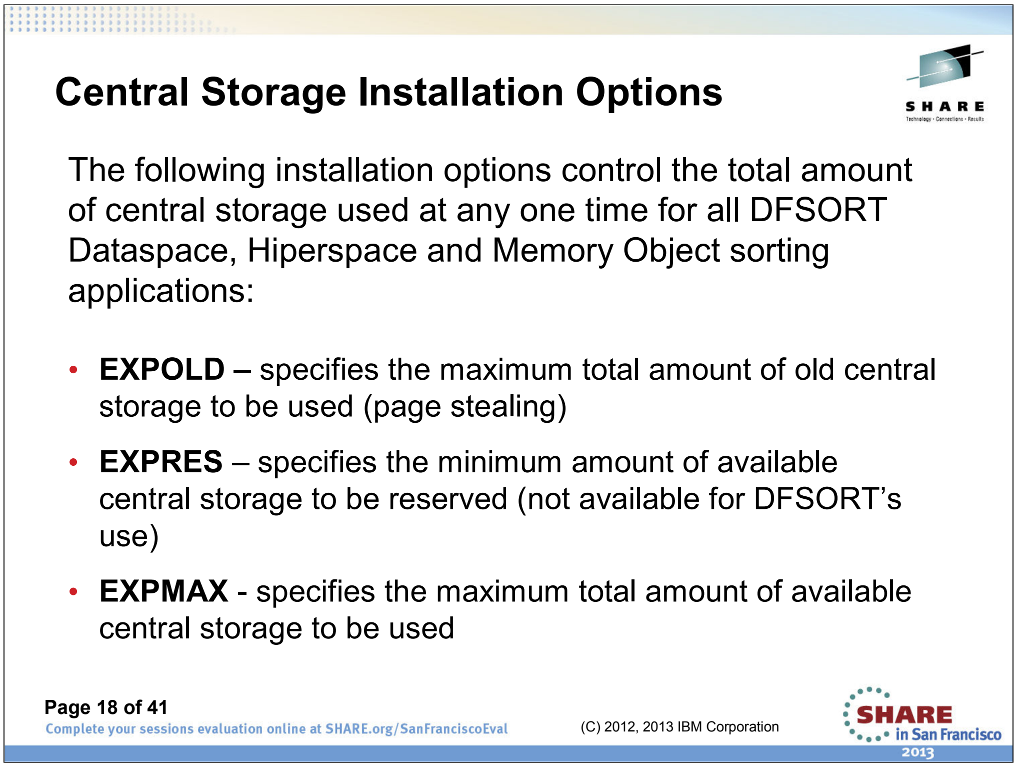
Memory Object Control

MOSIZE=[MAX | n | p%]
Installation option or run-time option specifies the maximum amount of memory object that can be used with memory object sorting.

- **MAX** (default)
  - Lets DFSORT determine maximum amount of memory objects to be used.
- **n**
  - Specifies the maximum in megabytes that can be used.
  - n must be a value between 0 and 2147483646
  - 0 (zero) specified memory objects will not be used
- **p%**
  - p percent (of an appropriate portion of central storage) amount of storage to be used for creating memory objects
  - p must be a value between 0 and 100

MOWRK | NOMOWRK Runtime Option
MOWRK=[YES | NO] Installation option
Specifies whether the memory object storage available to DFSORT can be used as intermediate work

MEMLIMIT on the JOB or EXEC statement



Central Storage Installation Options

The following installation options control the total amount of central storage used at any one time for all DFSORT Dataspace, Hiperspace and Memory Object sorting applications:

- **EXPOLD** – specifies the maximum total amount of old central storage to be used (page stealing)
- **EXPRES** – specifies the minimum amount of available central storage to be reserved (not available for DFSORT's use)
- **EXPMAX** - specifies the maximum total amount of available central storage to be used

See z/OS DFSORT Installation and Customization publication for details for:

EXPOLD= [MAX | n | p%]

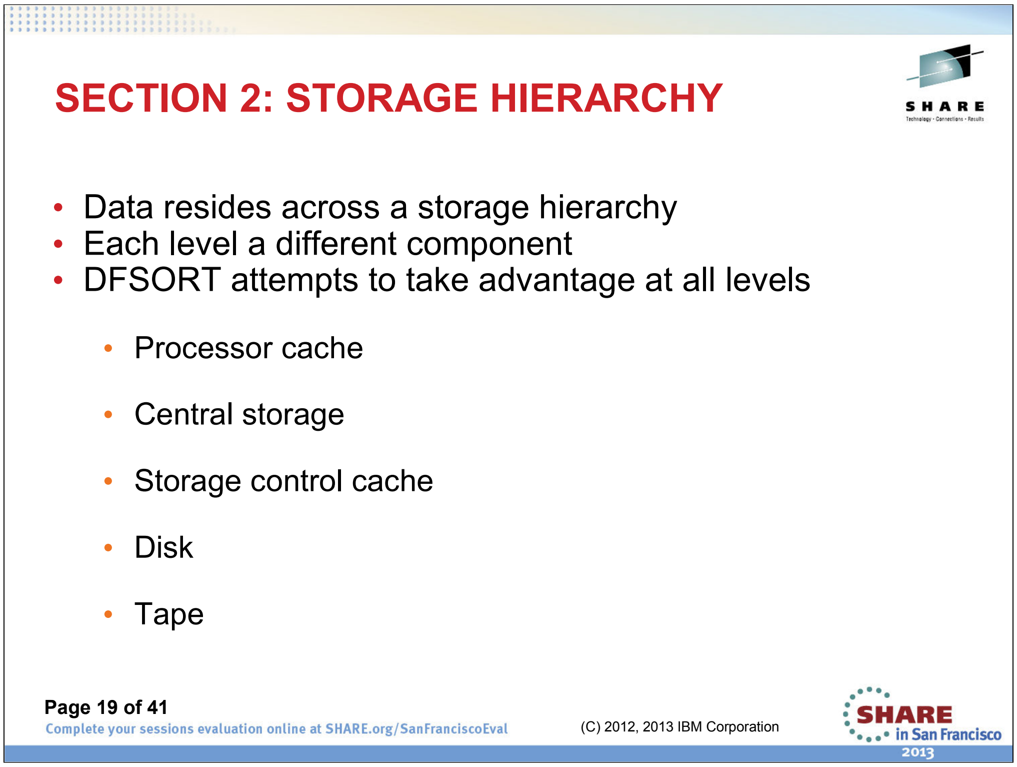
EXPRES= [MAX | n | p%]

EXPMAX= [MAX | n | p%]

- **MAX** allows DFSORT to determine the amount to use based on size of input and available resources
- **n** specifies a fixed value
- **p%** specifies p percent of the configured central storage

Old storage is the storage that is being used by other applications, but whose data has been unreferenced for a sufficiently long period of time that the system considers it eligible to be paged out to auxiliary storage to make room for new hiperspace, memory object or data space to be created.

If auxiliary storage shortages are a problem when running DFSORT applications, a small value or zero for EXPOLD is recommended.



SECTION 2: STORAGE HIERARCHY

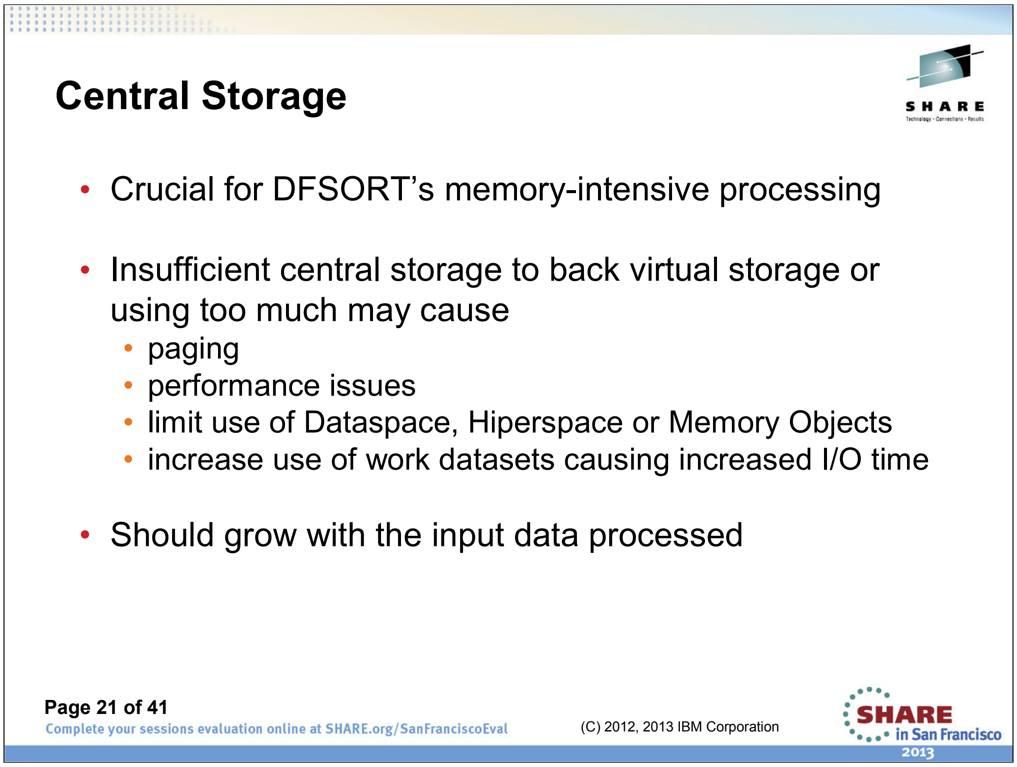
- Data resides across a storage hierarchy
- Each level a different component
- DFSORT attempts to take advantage at all levels
  - Processor cache
  - Central storage
  - Storage control cache
  - Disk
  - Tape

Processor Cache

- Special kind of high-speed memory
  - Faster access rate than central storage

- Accesses copies of instructions/data recently referenced
  - cache misses=higher CPU times

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



Central Storage

- Crucial for DFSORT’s memory-intensive processing
- Insufficient central storage to back virtual storage or using too much may cause
  - paging
  - performance issues
  - limit use of Dataspace, Hiperspace or Memory Objects
  - increase use of work datasets causing increased I/O time
- Should grow with the input data processed

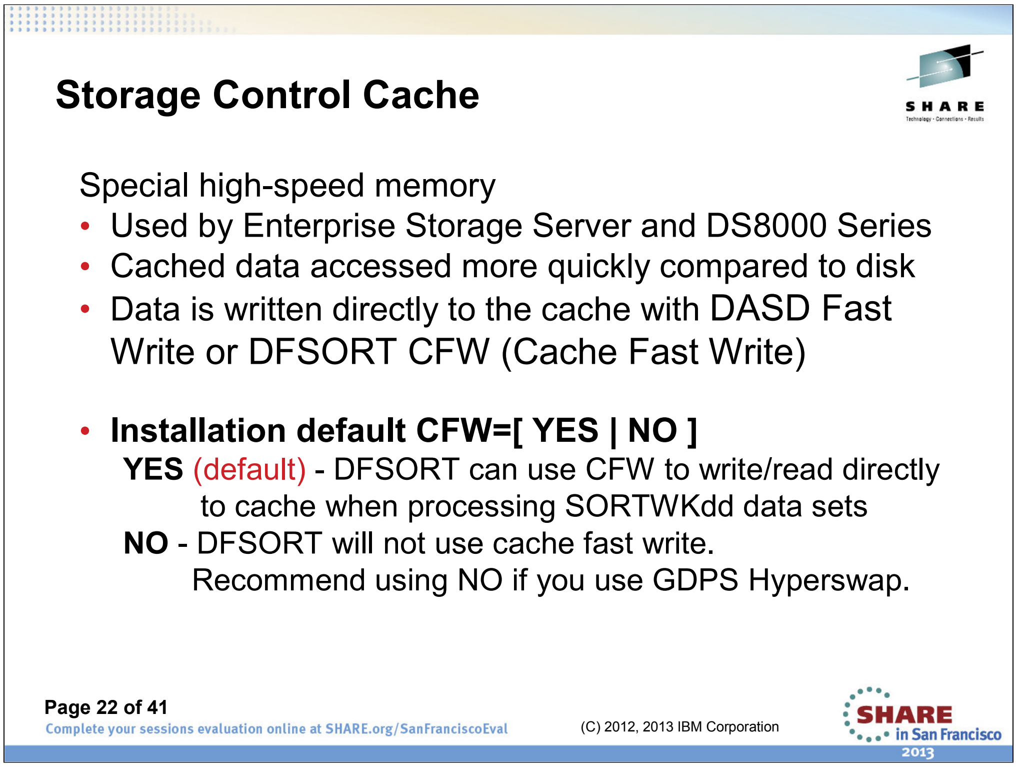
Complete your sessions evaluation online at SHARE.org/SanFranciscoEval

Recommend DSA be used so DFSORT can automatically increase the amount of virtual storage used by DFSORT based on DFSORT knowing the file size.

File size can not be determined by DFSORT when:

- an E15 inserts all or some input records or
- information about an input tape data set is not available from a tape management system or
- Blockset technique is not selected

See DFSORT Application Programming Guide, Appendix File Size and Dynamic Allocation and Section OPTION Control Statement, option FILSZ/SIZE for information on how to pass file size information to DFSORT.



Storage Control Cache

Special high-speed memory

- Used by Enterprise Storage Server and DS8000 Series
- Cached data accessed more quickly compared to disk
- Data is written directly to the cache with DASD Fast Write or DFSORT CFW (Cache Fast Write)

- **Installation default CFW=[YES | NO]**

  YES (default) - DFSORT can use CFW to write/read directly to cache when processing SORTWKdd data sets

  NO - DFSORT will not use cache fast write.
  Recommend using NO if you use GDPS Hyperswap.

The term cache fast write (CFW) is normally used to refer to the capability of the enterprise storage subsystems to write temporary data to cache memory only and eliminate the need for writing to disk.

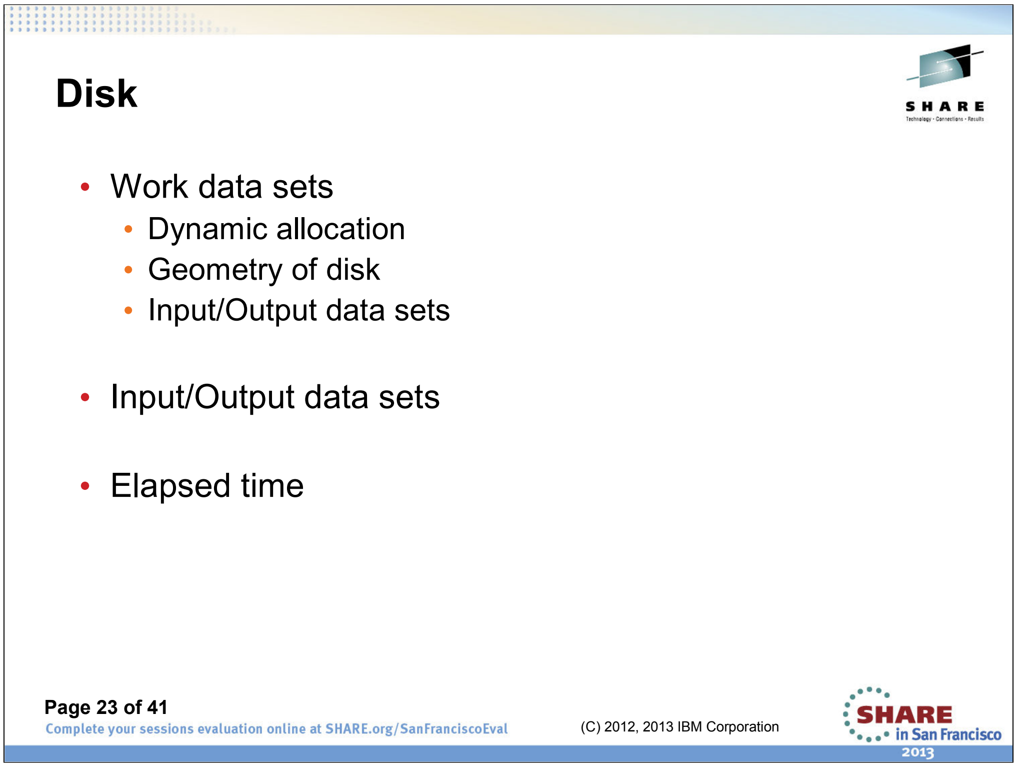
You DO NOT want to use CFW when GDPS HYPERSWAP is in effect. This can cause data still in the cache to be lost if you have a storage subsystem failure.

With today's larger/faster Enterprise Storage Server technology,

- cache and non-volatile storage sizes are much larger
- cache to disk interfaces are much faster
- data is striped across arrays

Therefore the potential degradation of turning off CFW is minimized.

To determine if CFW can benefit you, perform performance test with your more important DFSORT jobs with CFW=NO and CFW=YES and compare the results.



Disk

- Work data sets
  - Dynamic allocation
  - Geometry of disk
  - Input/Output data sets
- Input/Output data sets
- Elapsed time

Work Data sets:

- If DFSORT’s dynamic allocation is used, allocations are done automatically
- DFSORT writes using the geometry of the disk
- DFSORT application data sets are accessed concurrently and should reside on separate devices to help avoid channel, control unit path and device contention (which causes performance degradation). Use separate devices for the
  - input and work data sets, and for
  - output and work data sets

Input/Output Data sets

Use fastest disks for at least the input/output datasets to maximize performance

Elapsed time

May increase if using non-synchronous storage control units or ESCON® channels. Especially important to follow virtual storage guidelines

Disk (continued)

- Block sizes/data also crucial to performance
  - If dynamic allocation is used, DFSORT will select optimal sort work data set blocksize
  - If coding work data sets in JCL, select efficient blocksize (for example use System Determined Blocksize (SDB))

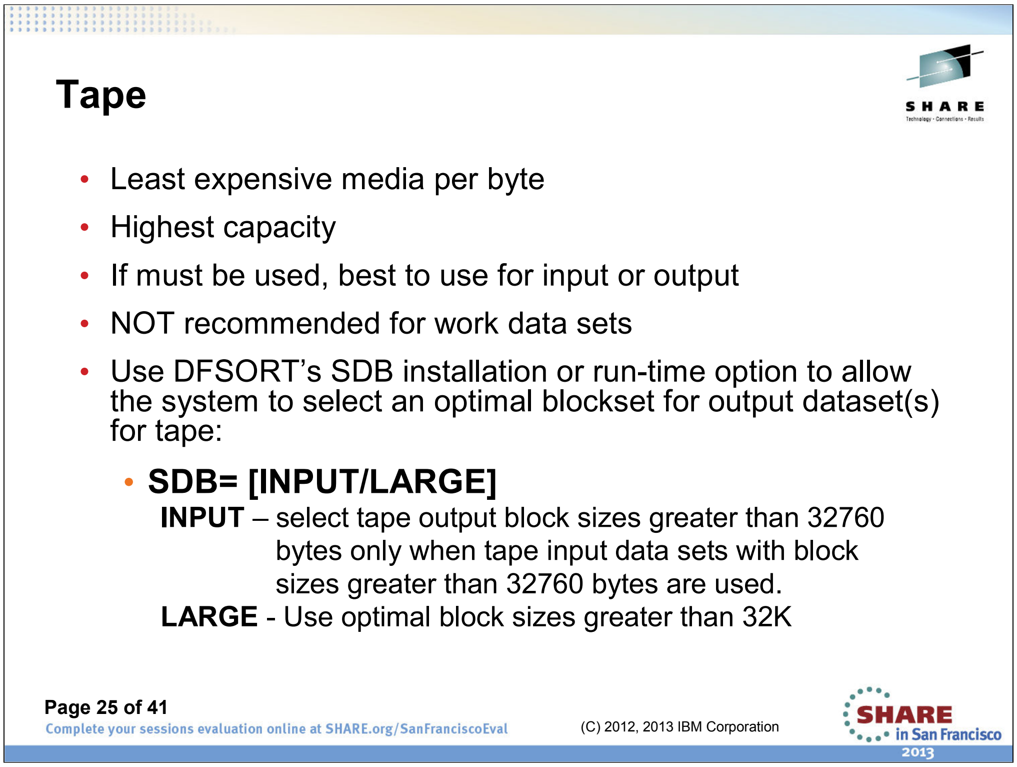
- SDB installation option or run-time option allows system to select an optimal block size for output dataset(s), when you do not code it yourself

  SDB= [YES | NO | INPUT | LARGE]

  YES - System selects up to 32760 bytes

  NO - Do not use System-determined blocksizes

  INPUT/LARGE - for tape (see next slide)



Tape

- Least expensive media per byte
- Highest capacity
- If must be used, best to use for input or output
- NOT recommended for work data sets
- Use DFSORT's SDB installation or run-time option to allow the system to select an optimal blockset for output dataset(s) for tape:
  - **SDB= [INPUT/LARGE]**
    INPUT – select tape output block sizes greater than 32760 bytes only when tape input data sets with block sizes greater than 32760 bytes are used.
    LARGE - Use optimal block sizes greater than 32K

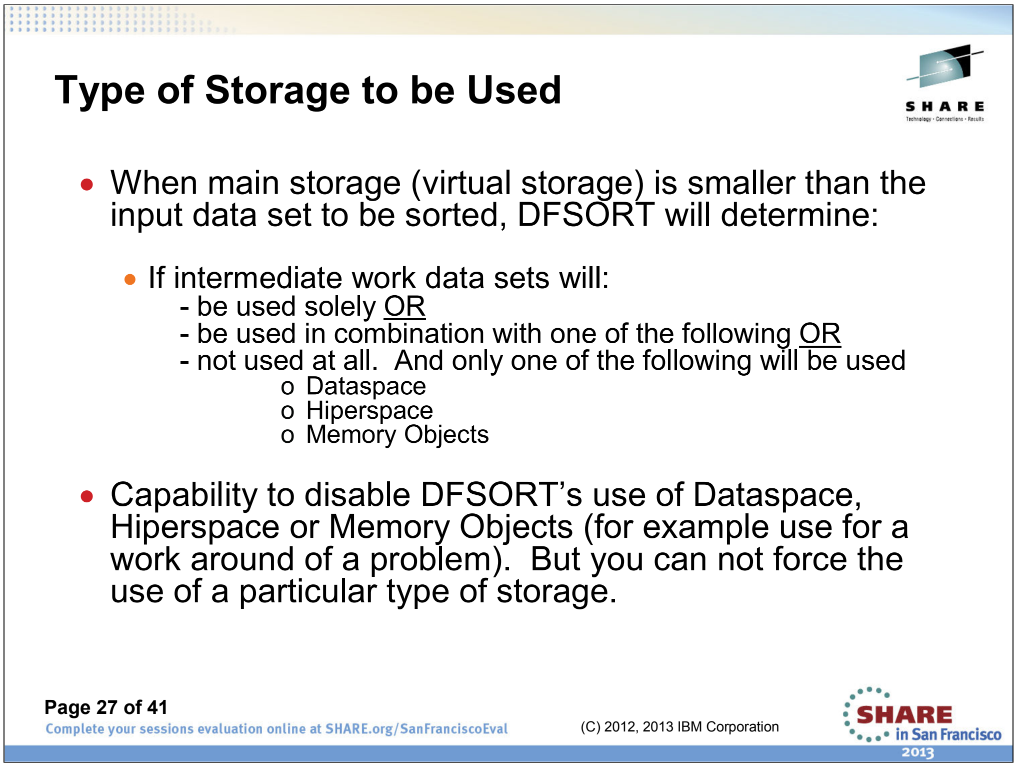
Input and output is read and written sequentially. So if tapes must be used you could use for tape. Also compacted tapes are a good idea for the input or output.

NOT recommended for work data sets (not accessed sequentially) which causes poor performance.

See z/OS DFSORT Application Programming Guide or DFSORT z/OS Installation Customization for details of SDB run-time option or installation option

SECTION 3: Run-Time Considerations

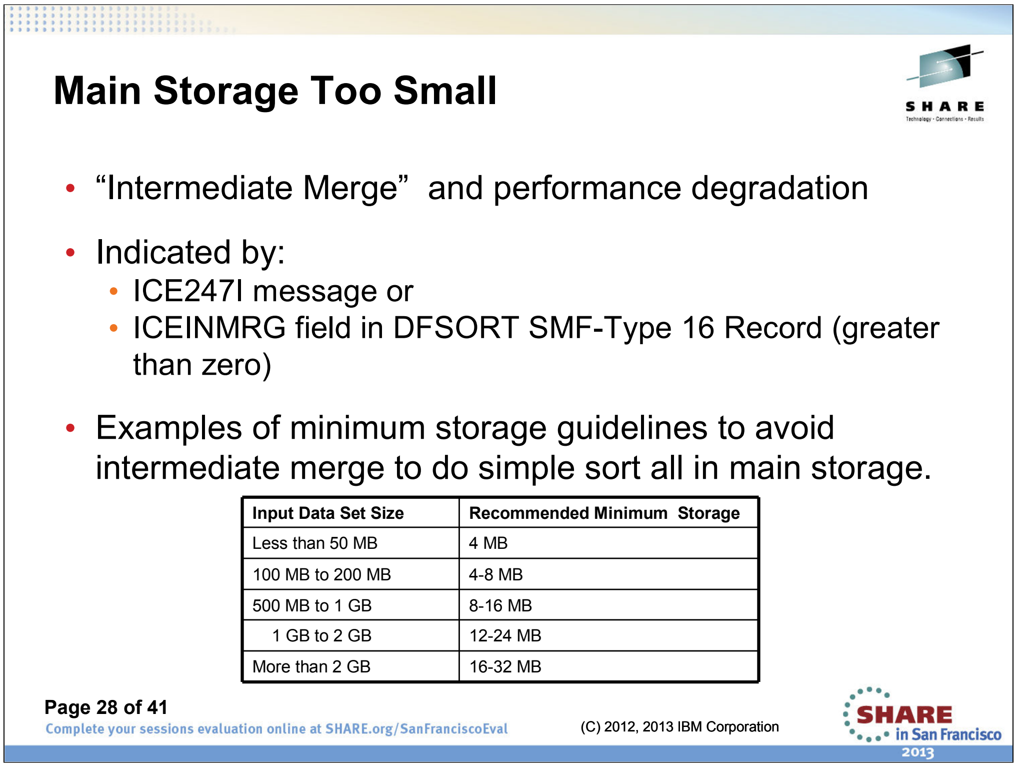
- Selection of Storage Type to be Used
- General Characteristics of using Dataspace, Hiperspace, and Memory Objects



Type of Storage to be Used

- When main storage (virtual storage) is smaller than the input data set to be sorted, DFSORT will determine:
  - If intermediate work data sets will:
    - be used solely <u>OR</u>
    - be used in combination with one of the following <u>OR</u>
    - not used at all. And only one of the following will be used
      - o Dataspace
      - o Hiperspace
      - o Memory Objects
- Capability to disable DFSORT's use of Dataspace, Hiperspace or Memory Objects (for example use for a work around of a problem). But you can not force the use of a particular type of storage.

If a defect in the DFSORT code was encountered, you can disable the use of dataspace, hiperspace or memory objects. For example, if an ABEND0C4-3B (often related to 64-bit addressing mode errors) occurs in a job during memory object sorting, you can investigate the problem with IBM service. But if you need the job to complete. Re-run the job using MOSIZE= 0.



Main Storage Too Small

- “Intermediate Merge” and performance degradation
- Indicated by:
  - ICE247I message or
  - ICEINMRG field in DFSORT SMF-Type 16 Record (greater than zero)
- Examples of minimum storage guidelines to avoid intermediate merge to do simple sort all in main storage.

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

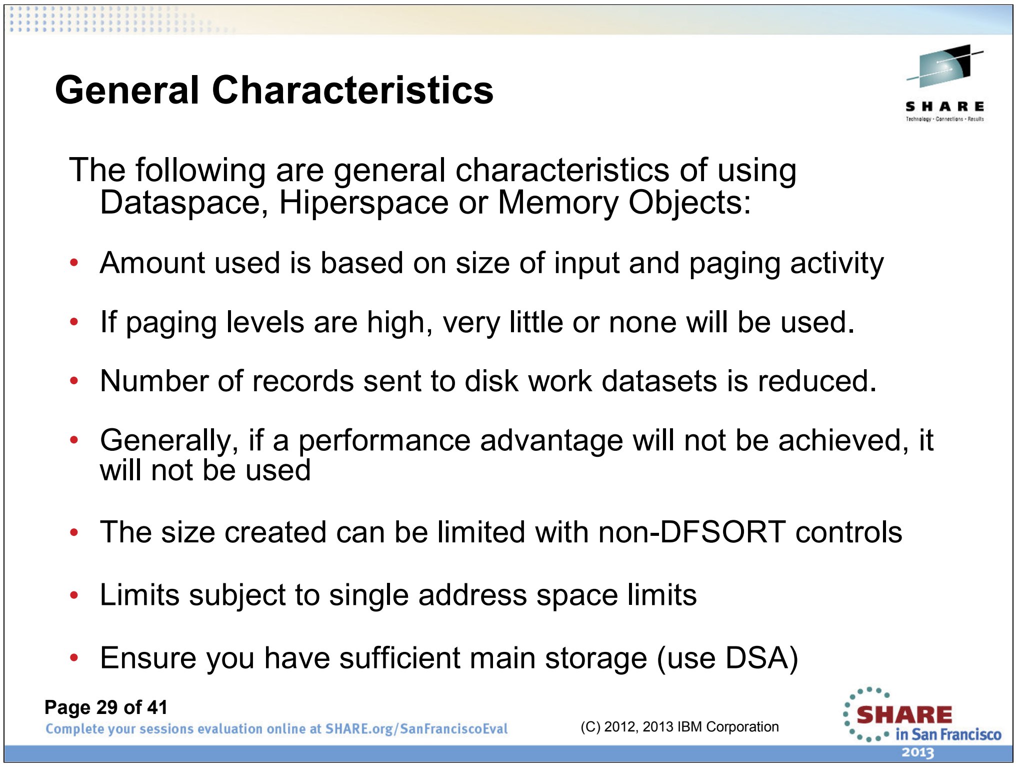
When the amount of virtual storage available to DFSORT is a small fraction of the amount of data to be SORTed, DFSORT will require additional (intermediate merge) passes over the data to perform the sort. This often results in significant performance degradation with a consequent increase in the required intermediate storage and an increased elapsed/CPU time.

Table is only an example to demonstrate the magnitude of what is meant by “small fraction”. The actual amount of storage needed will vary depending on the DFSORT features used. It is NOT recommended that you use this table to determine actual amounts of main storage to be set for your individual sorts. Using DSA is recommended method to allow DFSORT to select the main storage used. See z/OS DFSORT Tuning Guide, Chapter Run-Time Considerations, Section Storage, Subsection Virtual Storage Guidelines

All other factors being equal, the range of data set sizes that DFSORT can sort efficiently (or sort without requiring intermediate merging) grows roughly with the amount of virtual storage size. For example:

- doubling the virtual storage in an application enables the application to
  handle data sets four times as large with the same degree of efficiency OR

- halving the virtual storage causes the application to
  handle data sets only one-fourth as large with the same efficiency



General Characteristics

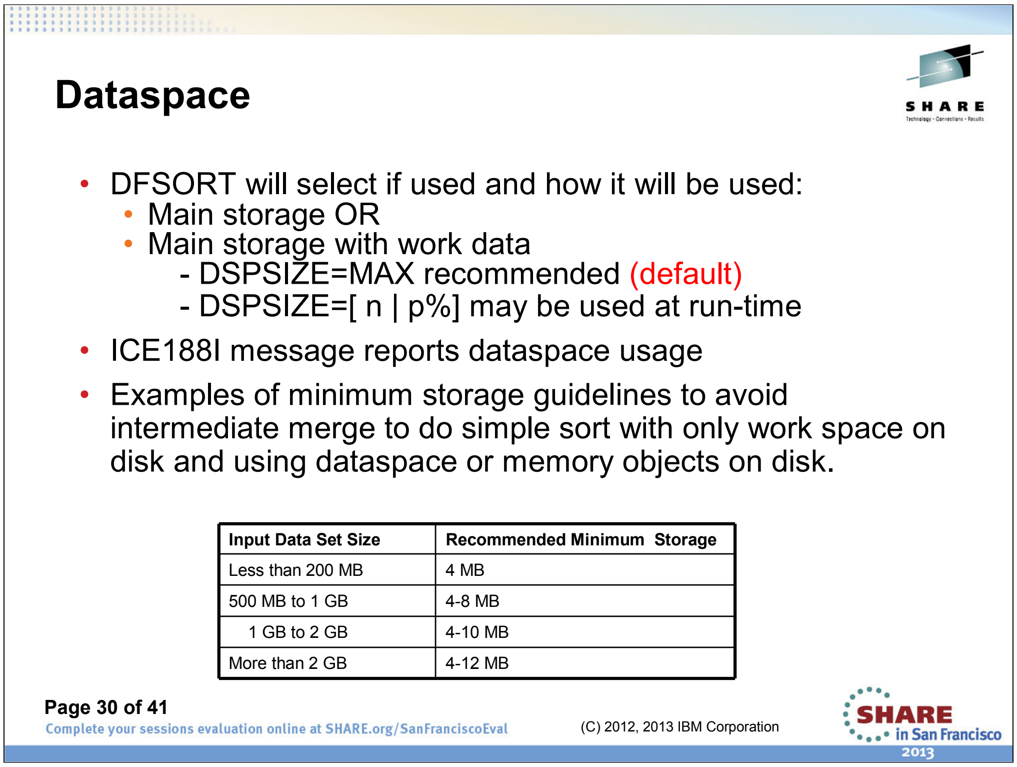
The following are general characteristics of using Dataspace, Hiperspace or Memory Objects:

- Amount used is based on size of input and paging activity
- If paging levels are high, very little or none will be used.
- Number of records sent to disk work datasets is reduced.
- Generally, if a performance advantage will not be achieved, it will not be used
- The size created can be limited with non-DFSORT controls
- Limits subject to single address space limits
- Ensure you have sufficient main storage (use DSA)

DFSORT will perform better if DFSORT can determine the input file size.

Limit on size created can be due to:

- Available central storage
- DSPSIZE, HIPRMAX, MOSIZE installation defaults or run-time options specified for an individual DFSORT job
- EXPOLD, EXPMAX, EXPRES installation defaults used for all DFSORT jobs at any one time
- Time of Day set of defaults
- Override in ICEIEXIT
- System limitations such as those imposed by an IEFUSI exit



Dataspace

- DFSORT will select if used and how it will be used:
  - Main storage OR
  - Main storage with work data
    - DSPSIZE=MAX recommended (default)
    - DSPSIZE=[n | p%] may be used at run-time
- ICE188I message reports dataspace usage
- Examples of minimum storage guidelines to avoid intermediate merge to do simple sort with only work space on disk and using dataspace or memory objects on disk.

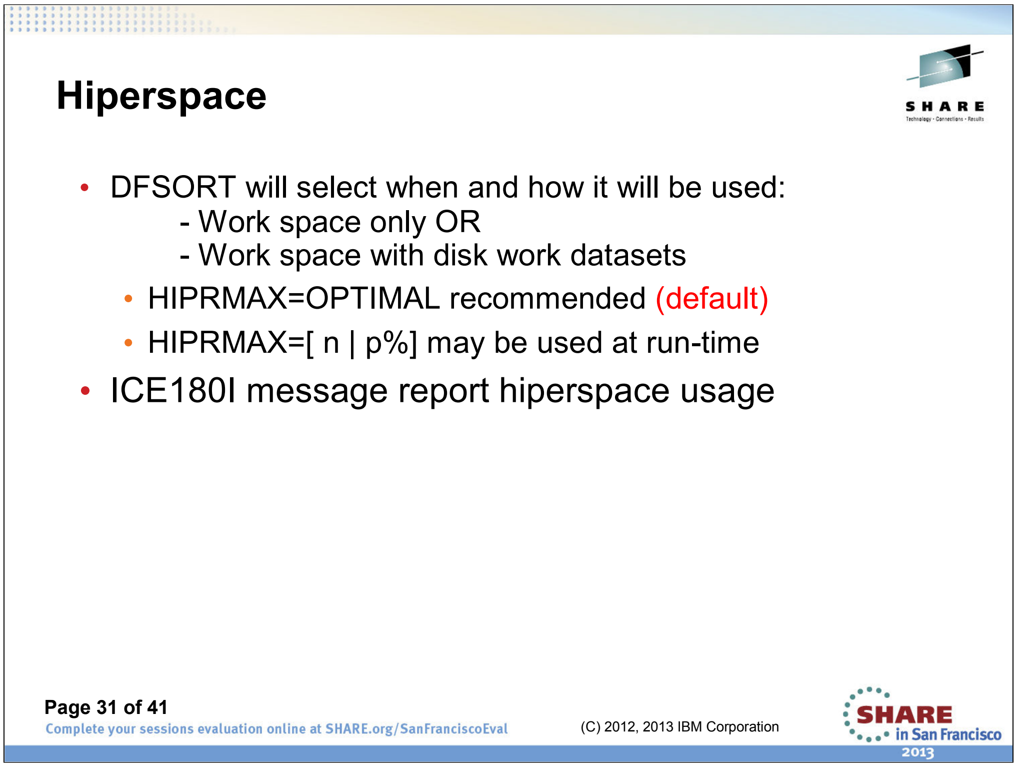
Input Data Set Size	Recommended Minimum Storage
Less than 200 MB	4 MB
500 MB to 1 GB	4-8 MB
1 GB to 2 GB	4-10 MB
More than 2 GB	4-12 MB

Complete your sessions evaluation online at SHARE.org/SanFranciscoEval

ICE188I   DATA SPACE STORAGE USED = nK BYTES

Explanation: n is the number of Kilobytes of data space storage used during this sort. If n is zero, data space storage was not used.

Regarding this table, see:
z/OS DFSORT Tuning Guide,
Chapter Run-Time Considerations, Section Storage,
Subsection Virtual Storage and Sorting with
Data Space or Memory Objects

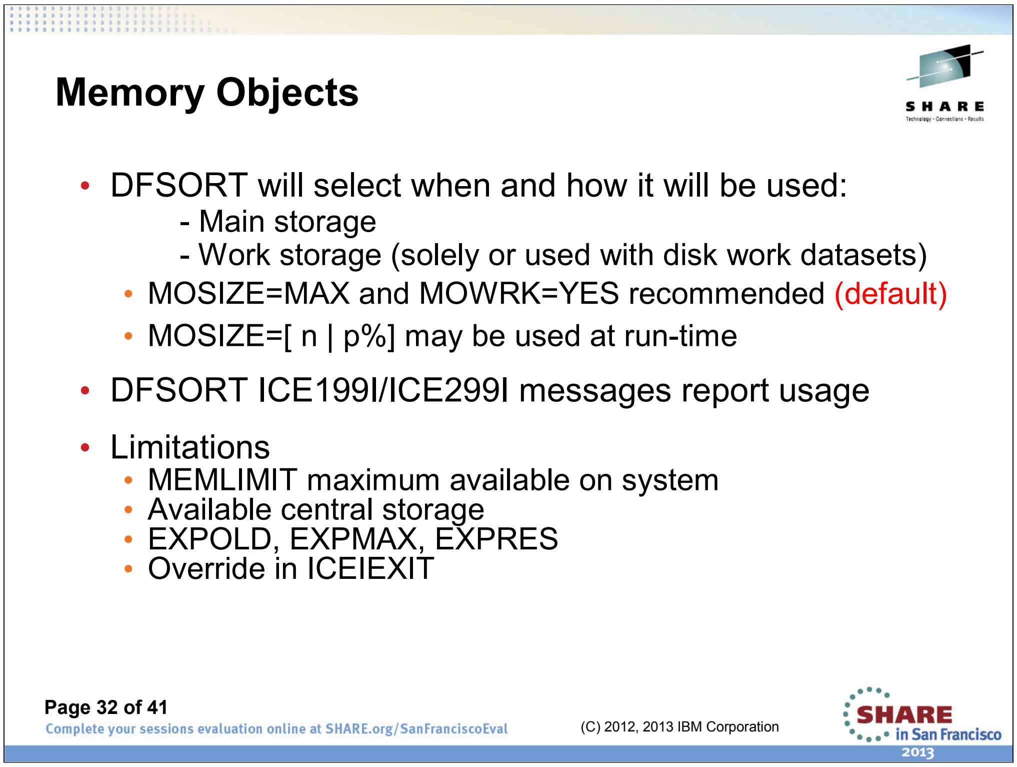


Hiperspace

- DFSORT will select when and how it will be used:
  - Work space only OR
  - Work space with disk work datasets
  - HIPRMAX=OPTIMAL recommended (default)
  - HIPRMAX=[n | p%] may be used at run-time
- ICE180I message report hiperspace usage

ICE180I HIPERSPACE STORAGE USED = *n*K BYTES

Explanation: n is the number of Kilobytes of Hiperspace storage used during this sort. If *n* is zero, Hiperspace storage was not used.



Memory Objects

- DFSORT will select when and how it will be used:
  - Main storage
  - Work storage (solely or used with disk work datasets)
  - MOSIZE=MAX and MOWRK=YES recommended (default)
  - MOSIZE=[n | p%] may be used at run-time
- DFSORT ICE199I/ICE299I messages report usage
- Limitations
  - MEMLIMIT maximum available on system
  - Available central storage
  - EXPOLD, EXPMAX, EXPRES
  - Override in ICEIEXIT

MOSIZE= [n | p%] may be used at run-time for individual jobs that you want to use significantly less or more than the installation default.

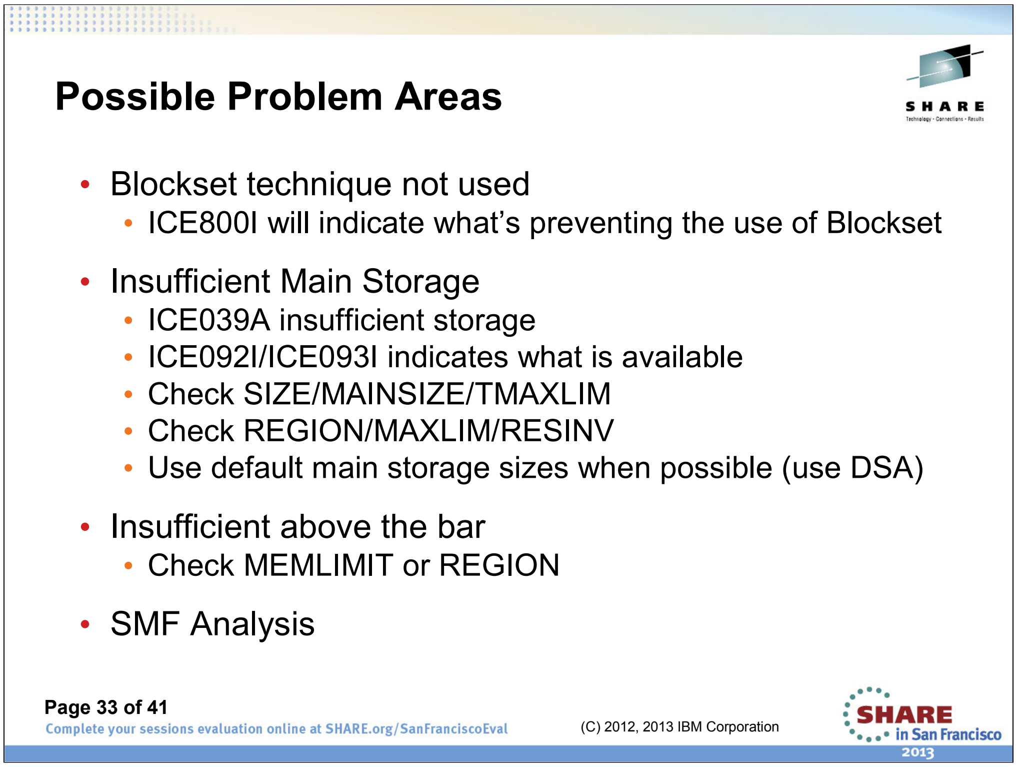
ICE199I MEMORY OBJECT STORAGE USED AS MAIN STORAGE = *n*M BYTES

Explanation: *n* is the number of megabytes of memory object storage DFSORT used as main storage during this sort. If *n* is zero, DFSORT did not use memory object storage as main storage during this sort.

ICE299I MEMORY OBJECT USED AS WORK STORAGE = *n*M BYTES
Explanation: *n* is the number of megabytes of memory object storage DFSORT used as intermediate work space during this sort. If *n* is zero, DFSORT did not use memory object storage as intermediate work space during this sort.

System wide, MEMLIMIT can be limited/set in a number of ways:

- SMF (SMFPRMxx) default
- JCL
- Unlimited with REGION=0K/0M)
- IEFUSI
- UNIX functions
- System authorized functions



Possible Problem Areas

- Blockset technique not used
  - ICE800I will indicate what's preventing the use of Blockset
- Insufficient Main Storage
  - ICE039A insufficient storage
  - ICE092I/ICE093I indicates what is available
  - Check SIZE/MAINSIZE/TMAXLIM
  - Check REGION/MAXLIM/RESINV
  - Use default main storage sizes when possible (use DSA)
- Insufficient above the bar
  - Check MEMLIMIT or REGION
- SMF Analysis

DFSORT uses 3 techniques, Blockset, Peervale and Conventional. New functions are only added/supported in Blockset since it is the most efficient technique

SORTDIAG DD must be coded OR installation option DIAGSIM=YES must be in effect for the ICE800I to be seen. See z/OS DFSORT Messages, Codes and Diagnosis, Section Diagnosis Messages, Subsection ICE800I to determine the exact condition that is preventing the use of Blockset.

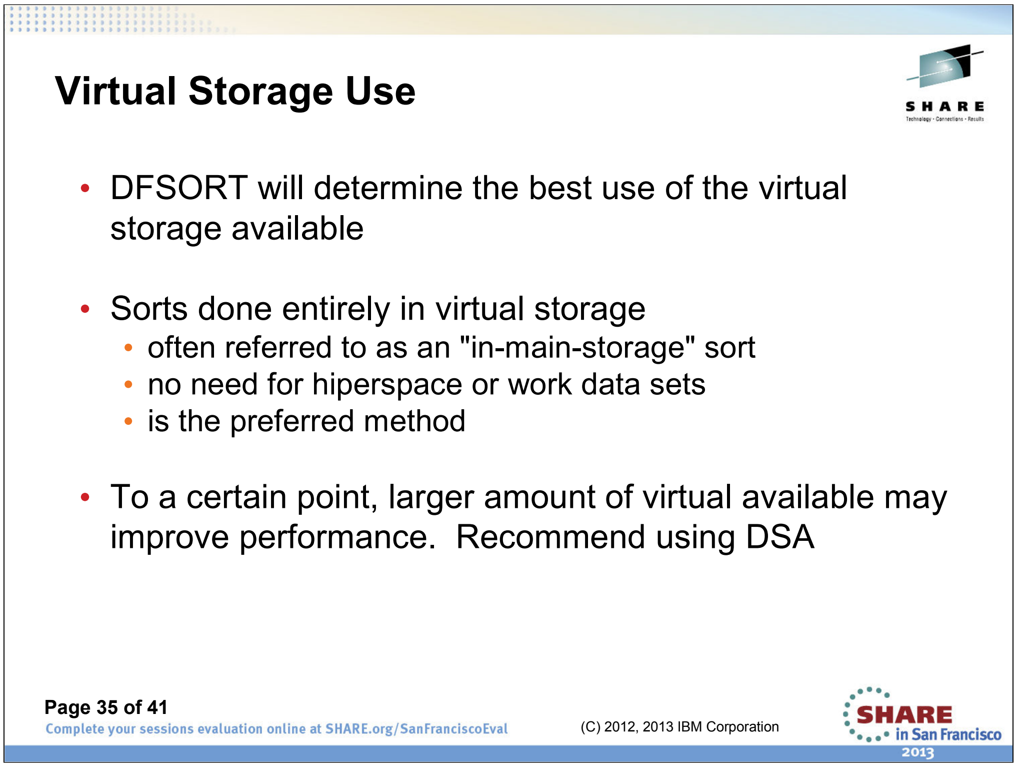
ICE092I MAIN STORAGE = (x,y,z)
Explanation: Information related to the use of main storage for this DFSORT application:
x The main storage value specified (subject to the SIZE/MAINSIZE limit), or supplied by default.
y The main storage theoretically available to DFSORT, considering the MINLIM value specified when the program was installed.
z The main storage actually available to DFSORT, after any other program took what it needed from the region (invoking program or exit routines).

ICE093I MAIN STORAGE = (MAX,y,z)
Explanation: Information related to the use of main storage for this DFSORT application:
MAX The value MAX was in effect.
y The main storage theoretically available to DFSORT, considering the TMAXLIM or MAXLIM values specified when the program was installed.
z The main storage actually available to DFSORT, after any other program took what it needed from the region (invoking program or user exit routines).

DFSORT SMF Record Type 16 can used to analyze your DFSORT workload. For example, you can identify all the jobs that are using intermediate merge by looking at the Data Section, field ICEINMRG. If it is greater than 0, then tune the job (consider if DSA is used, if file size should be used etc).

SECTION 4: Virtual Storage

- Virtual Storage Use
- Data Set Size and Virtual Storage
- Virtual Storage Limitations

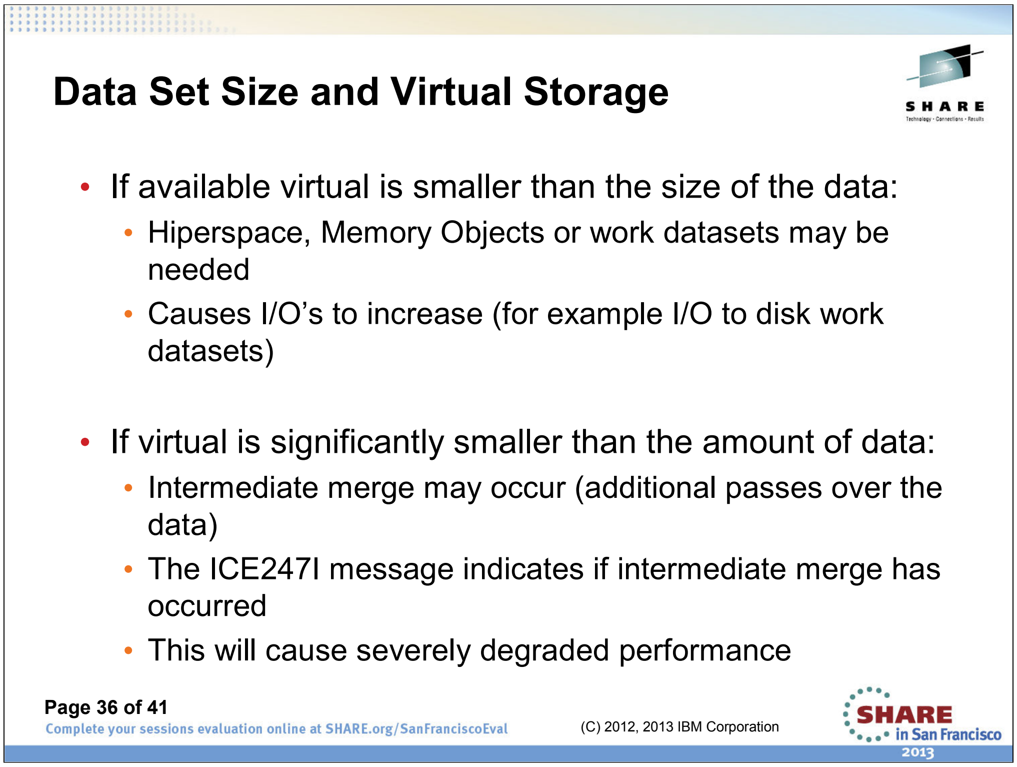


Virtual Storage Use

- DFSORT will determine the best use of the virtual storage available
- Sorts done entirely in virtual storage
  - often referred to as an "in-main-storage" sort
  - no need for hiperspace or work data sets
  - is the preferred method
- To a certain point, larger amount of virtual available may improve performance. Recommend using DSA

When using larger amounts of virtual, ensure you have sufficient real storage to backup the amount of storage used.

With the possible exception of in-main storage sorts, providing more storage than needed to do an efficient sort (see z/OS DFSORT Tuning Guide, Chapter Run-Time Considerations, Section Storage, Virtual Storage Guidelines Table 4 in topic 5.6.3 for storage guidelines) will probably not result in any significant performance improvement. In fact, elapsed time (and possibly CPU time) may even increase slightly. 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.



Data Set Size and Virtual Storage

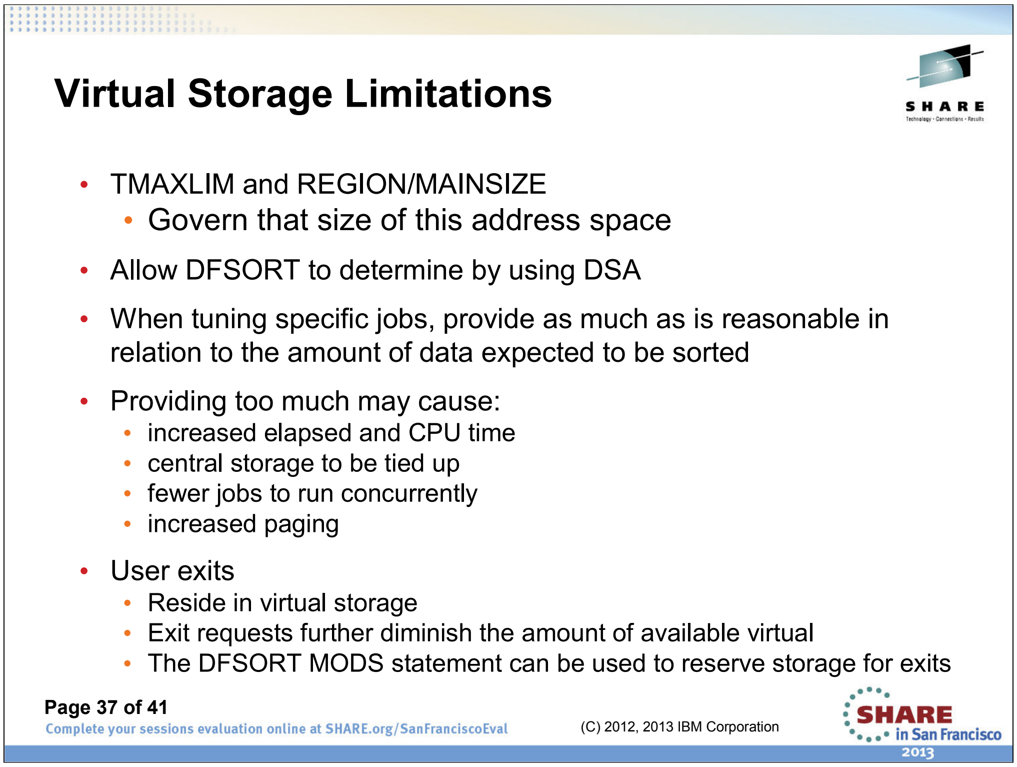
- If available virtual is smaller than the size of the data:
  - Hiperspace, Memory Objects or work datasets may be needed
  - Causes I/O’s to increase (for example I/O to disk work datasets)
- If virtual is significantly smaller than the amount of data:
  - Intermediate merge may occur (additional passes over the data)
  - The ICE247I message indicates if intermediate merge has occurred
  - This will cause severely degraded performance

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

In-Main Storage Sort
When virtual storage is larger than the data set, DFSORT may be able to perform the sort entirely within virtual storage, Dataspace or Memory Objects, without need to store intermediate data. This is called an in-main-storage sort minimizes I/O usage as well as CPU and elapsed time.

Sort with Hiperspace or Memory Objects and/or work data sets on disk
When virtual storage is smaller than the data set, Hiperspace or Memory Objects and/or work data sets are needed to store the intermediate data. Provided virtual storage is sufficient (see z/OS DFSORT Tuning Guide, Chapter Run-Time Considerations, Section Storage, Subsection Virtual Storage and Sorting with Data Space or Memory Objects 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 or Memory object usage is increased, however, reflecting the need to write intermediate data to Hiperspace or Memory Objects or work data sets. As the ratio of data set size to available storage increases, DFSORT may be forced to use Hiperspace, Memory Objects or work data sets less efficiently. The loss of efficiency adversely affects elapsed time and EXCP counts.

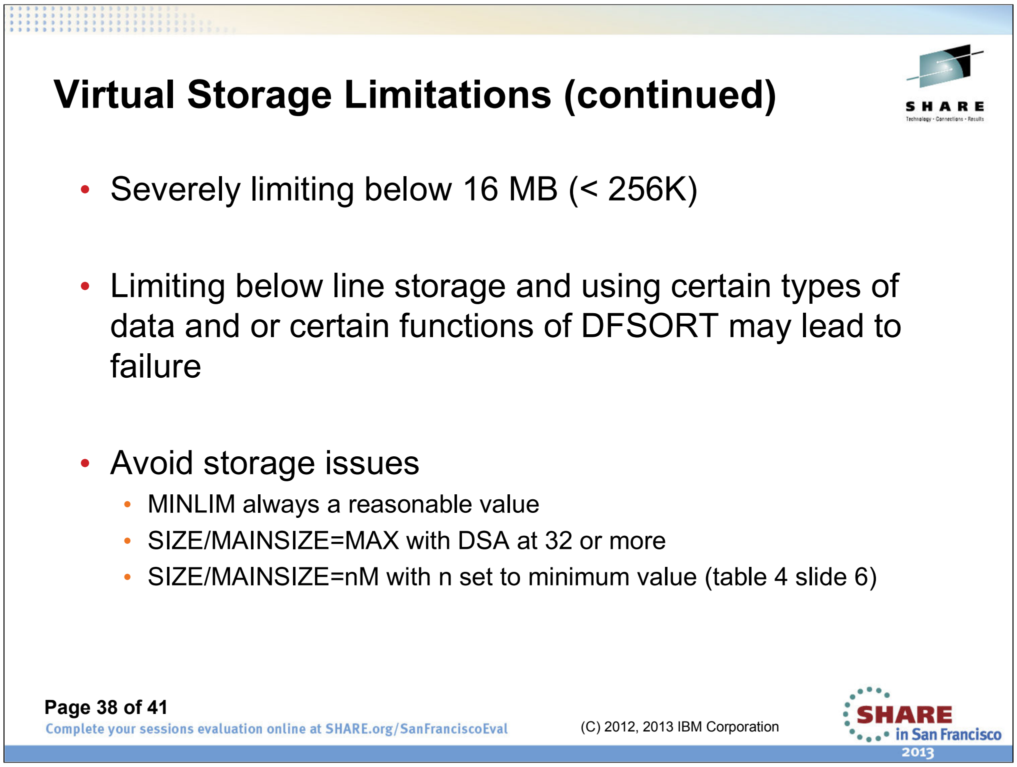
Intermediate Merge
When virtual storage is very small or the data set size is very large, DFSORT may require several additional passes over the data to perform the sort. This phenomenon is known as intermediate merging. DFSORT issues message ICE247I to indicate intermediate merging was required; processing continues with degraded performance.



Virtual Storage Limitations

- TMAXLIM and REGION/MAINSIZE
  - Govern that size of this address space
- Allow DFSORT to determine by using DSA
- When tuning specific jobs, provide as much as is reasonable in relation to the amount of data expected to be sorted
- Providing too much may cause:
  - increased elapsed and CPU time
  - central storage to be tied up
  - fewer jobs to run concurrently
  - increased paging
- User exits
  - Reside in virtual storage
  - Exit requests further diminish the amount of available virtual
  - The DFSORT MODS statement can be used to reserve storage for exits

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.



Virtual Storage Limitations (continued)

- Severely limiting below 16 MB (< 256K)
- Limiting below line storage and using certain types of data and or certain functions of DFSORT may lead to failure
- Avoid storage issues
  - MINLIM always a reasonable value
  - SIZE/MAINSIZE=MAX with DSA at 32 or more
  - SIZE/MAINSIZE=nM with n set to minimum value (table 4 slide 6)

Limiting below line storage and using the following types of data or functions of DFSORT may lead to insufficient storage situation:

- 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

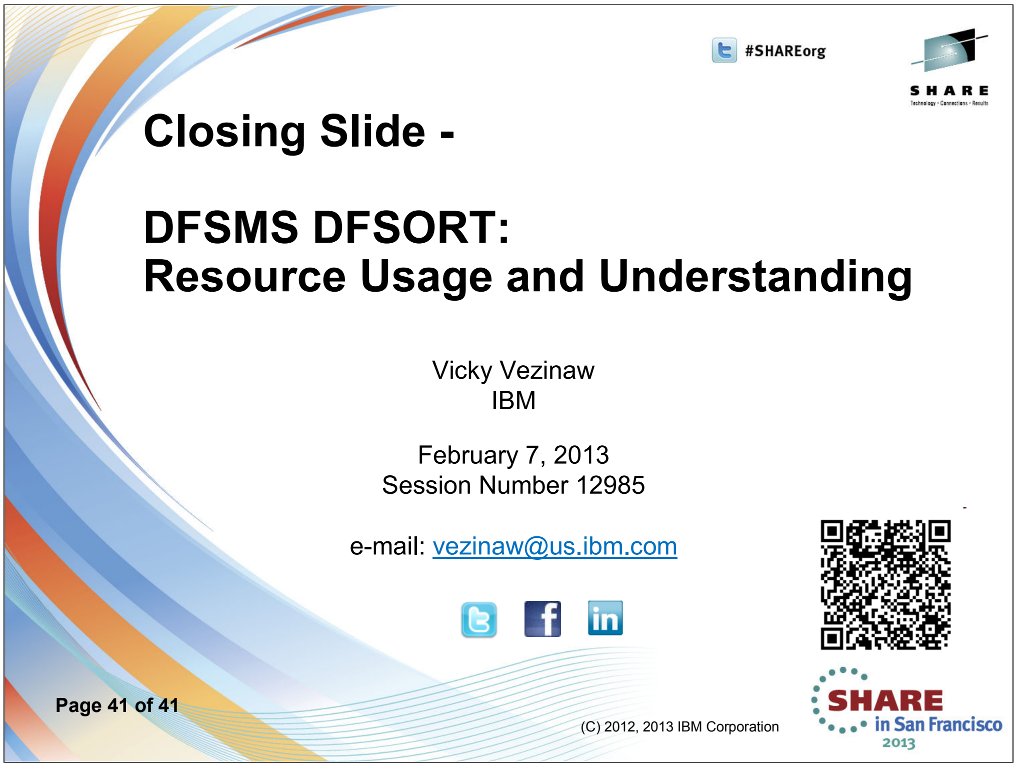
Contact Information

Send non-defect related questions to the DFSORT id:

dfsort@us.ibm.com

References

- z/OS DFSORT Application Programming Guide (SC26-7523)
- z/OS DFSORT Tuning Guide (SC26-7526)
- z/OS DFSORT Installation and Customization (SC26-7524)
- z/OS DFSORT Messages, Codes and Diagnosis Guide (SC26-7525)
- DFSORT home page (articles, online documents, news, tips, techniques, examples, and more)
  http://www.ibm.com/storage/dfsort



#SHAREorg

SHARE
Technology · Connections · Results

Closing Slide -

DFSMS DFSORT:
Resource Usage and Understanding

Vicky Vezinaw
IBM

February 7, 2013
Session Number 12985

e-mail: vezinaw@us.ibm.com

SHARE in San Francisco 2013