IBM Z Sort and DFSORT Considerations

Dave Betten
betten@us.ibm.com

John Burg
jpburg@us.ibm.com

Version Date: July 2, 2021

Objective and Summary

The purpose of this White Paper is to provide guidance on DFSORT's use of IBM Z Sort to reduce Sort elapsed time and Sort CPU time. This paper will discuss what is Z Sort, what DFSORT characteristics does it require, and with the IBM Z Batch Network Analyzer (zBNA) tool, identify potential candidates eligible to exploit it, and what settings and resources are required to maximize the Sorts that can utilize Z Sort. The scope includes virtual storage, DFSORT settings and available processor storage.

DFSORT Background

DFSORT is IBM's high-performance sort, merge, copy, analysis, and reporting product and is an optional feature of z/OS. For many years, DFSORT has employed highly efficient algorithms to process large volumes of data and continues to be an integral part of z/OS batch applications. Until now, the majority of DFSORT optimizations have been related to I/O and memory efficiency. DFSORT is able to exploit I/O technologies such as compression, data striping and zHPF to accelerate input and output I/O while making use of memory (Memory Objects, Hiperspaces and Dataspaces) to reduce work data set I/O. Recent DFSORT enhancements have positioned DFSORT to leverage large memory configurations while reducing the risk of negative performance impacts to other applications from over commitment of resources.

What is the IBM Z Sort and What are the DFSORT Characteristics Required to Exploit it?

The IBM z15 introduced a new on-chip accelerator available via a new SORTL instruction. DFSORT has been updated with a new sort algorithm designed to exploit this technology and accelerate the sorting process. The combination of these new technologies is referred to as IBM Z Sort. To fully benefit from this high-speed sorting algorithm DFSORT must be able to pass data to the on-chip accelerator at rapid speed. As a result, the initial exploitation does not include sorts that are utilizing the more complex functions such as INREC, OUTREC, OUTFIL, SUM, long sort keys, etc. Additionally, input and output exits commonly used by program invoked sorts are not yet supported as they cannot pass records to/from DFSORT fast enough to realize the benefits of the on-chip accelerator. The exception to this is DB2 Utilities where IBM has optimized the interface used to transfer records between the utility and the sort tasks. DFSORT's Z Sort algorithm also relies on in-memory sorting to reduce delays related to work data set IO that would offset the on-chip accelerator capabilities.

What resource related inhibitors exist to utilizing the IBM Z Sort?

As mentioned above, DFSORT's Z Sort criteria is based mostly on the functions and data characteristics that are out of the customer's control unless they can modify their sorts to fit the criteria. But other environmental factors can further limit the use of IBM Z Sort. The most significant is the amount of 64-bit memory available for sorting. The new IBM Z Sort algorithm requires a large portion of the file fit in memory. This is required to allow the IBM Z Sort algorithm to efficiently leverage the on-chip accelerator without having to move data between memory and intermediate disk storage. There are several factors that can limit the memory available to DFSORT.

1. DFSORT options can be tailored to restrict memory. Installation defaults such as EXPMAX, EXPOLD, EXPRES and MOSIZE should be evaluated to insure they are allowing DFSORT to fully utilize available memory. While MOSIZE can be overridden at run time via an OPTION statement, the EXP* parameters cannot.

2. MEMLIMIT can restrict the amount of 64-bit memory a sort can use. Quite often customers run with rather small default MEMLIMIT values (2GB for example). Usually, the default is set in the SMFPRMxx member of SYS1.PARMLIB but some installations also have an IEFUSI exit to set it. For sorts, often it is recommended to set MEMLIMIT=NOLIMIT to eliminate this restriction and let DFSORT determine the optimum amount of 64-bit memory to allocate without impacting overall systems performance.

It also is recommend the DFSORT defaults be set as follows EXPMAX=MAX, EXPRES=10% However, it is recommended EXPOLD be set to zero percent (EXPOLD=0%), as this will prevent DFSORT from using any "old" pages. Since processor storage is typically plentiful in most environments, there is no need to potentially take pages that may be utilized in later time periods by different workloads. To summarize, for DFSORT the recommendations are:

EXPMAX=MAX
EXPRES=10%
EXPOLD=0%
MOSIZE=MAX
MEMLIMIT=NOLIMIT

This will allow DFSORT to utilize available resources to maximize Memory Objects and thus IBM Z Sort exploitation.

zBNA and how can it identify IBM Z Sort Candidates

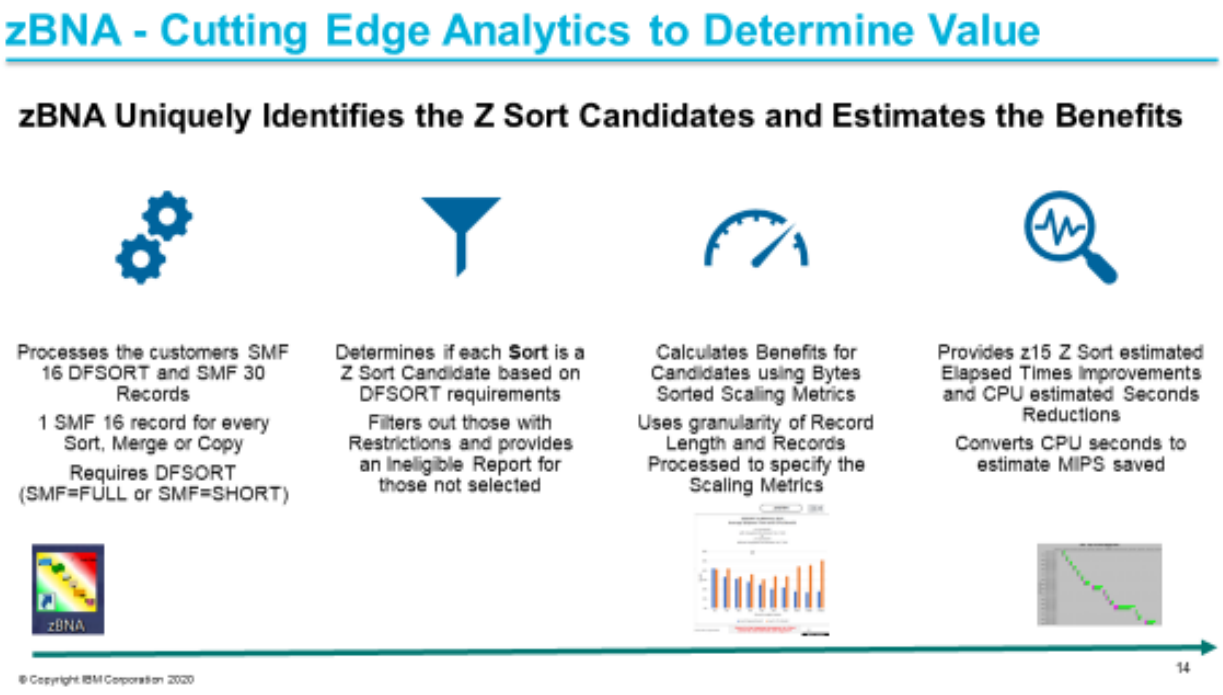
zBNA V2.2.4 is an ‘as is’, no cost tool available to customers. It now includes a new application, the DFSORT Z Sort Application which can identify DFSORT Z Sort candidates and estimated benefits.

Here is the link to obtain zBNA tool.
https://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS5132

There are zBNA Education materials on the download site, and one of them “Putting the New Z Sort Named Favorite into Practice Webinar” https://ibm.ent.box.com/v/zBNA-Z-Sort-Webinar is highly recommended on understanding how to use the zBNA tool.

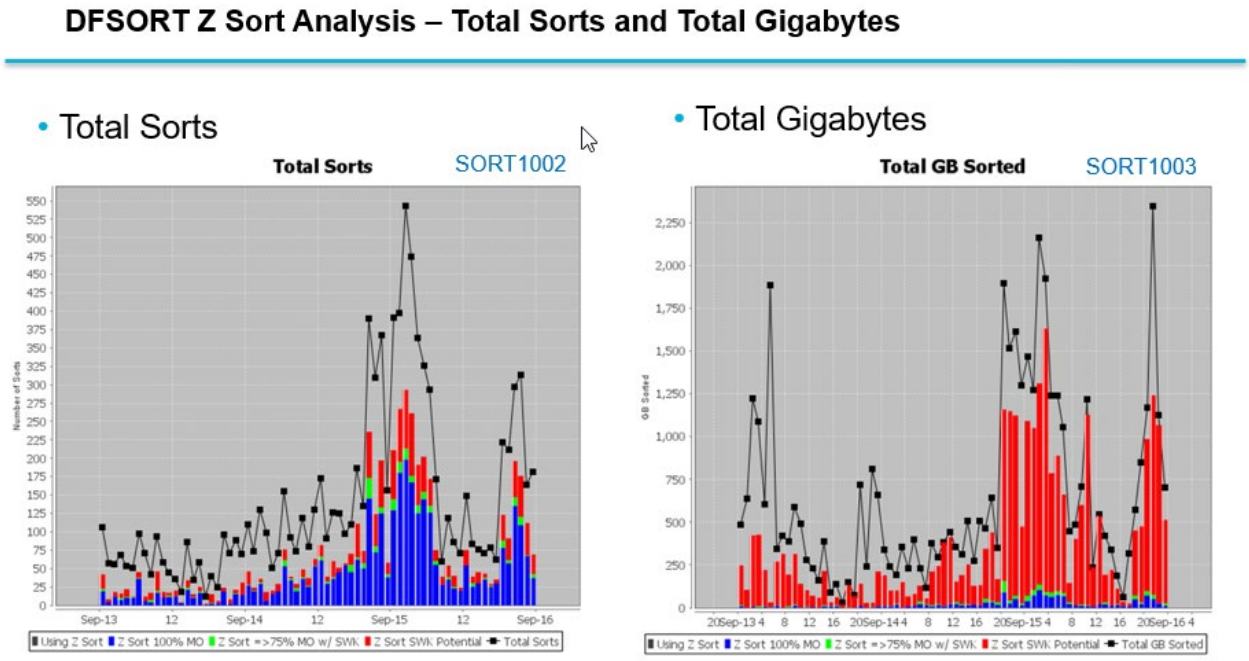
The zBNA Z Sort application uses SMF 16 (DFSORT) records as input and identifies the sorts with characteristics needed to exploit IBM Z Sort. zBNA also estimates the benefits of exploiting IBM Z sort and provides a series of reports and charts with the findings. DFSORT today will only use IBM Z Sort if the entire file fits in a Memory Object or at least 75% of the file size fits in a Memory Object. So, if a file was too big to fit into a Memory Object, it would use a different DFSORT path and likely use SortWork. In that situation, zBNA would not report it as a candidate.

It is common for customers to configure their systems to limit the size of Memory Objects. They may have resource limitations, or perhaps settings are left over from historic events or just carried forward.



The zBNA IBM Z Sort application also provides insights into sorts that cannot currently exploit IBM Z Sort. The purpose of this White Paper is to provided information to help assess if resources or settings may be limiting the usage of IBM Z Sort exploitation.

zBNA functionality can show potential IBM Z Sort candidates if the sort's current use of SortWorks is changed. This is identified in zBNA as "Z Sort SWK Potential" and shown in the charts below in red. zBNA originally showed only the fully eligible IBM Z Sort candidates (shown below in Blue).



The key point of this White Paper is to focus on the sorts in red, because they offer the largest potential for improvement. These sorts have met the IBM Z Sort requirements except they don't have the required Processor Storage or Virtual Storage required to utilize IBM Z Sort. There are typically fewer of these "SortWork" sorts, but they tend to sort larger files (Bytes Sorted), and thus have longer elapsed times and more CPU.

zBNA chart SORT1008 provides a report to showing the Top 15 IBM Z Sort SWK candidates. You can use this information to drive your analysis of which ones you'd want to improve (e.g. longest running, most important to the Batch Window or critical path). Then it's possible to investigate the sort metrics to see why they aren't using Memory Objects today. Possible inhibitors include:

- MEMLIMIT is too small compared to their GB Sorted
- Insufficient Processor Storage available
- DFSORT settings (EXPxx or MOSIZE) are preventing use of a Memory Object
- Running on a IBM z15 with IBM Z Sort is disabled

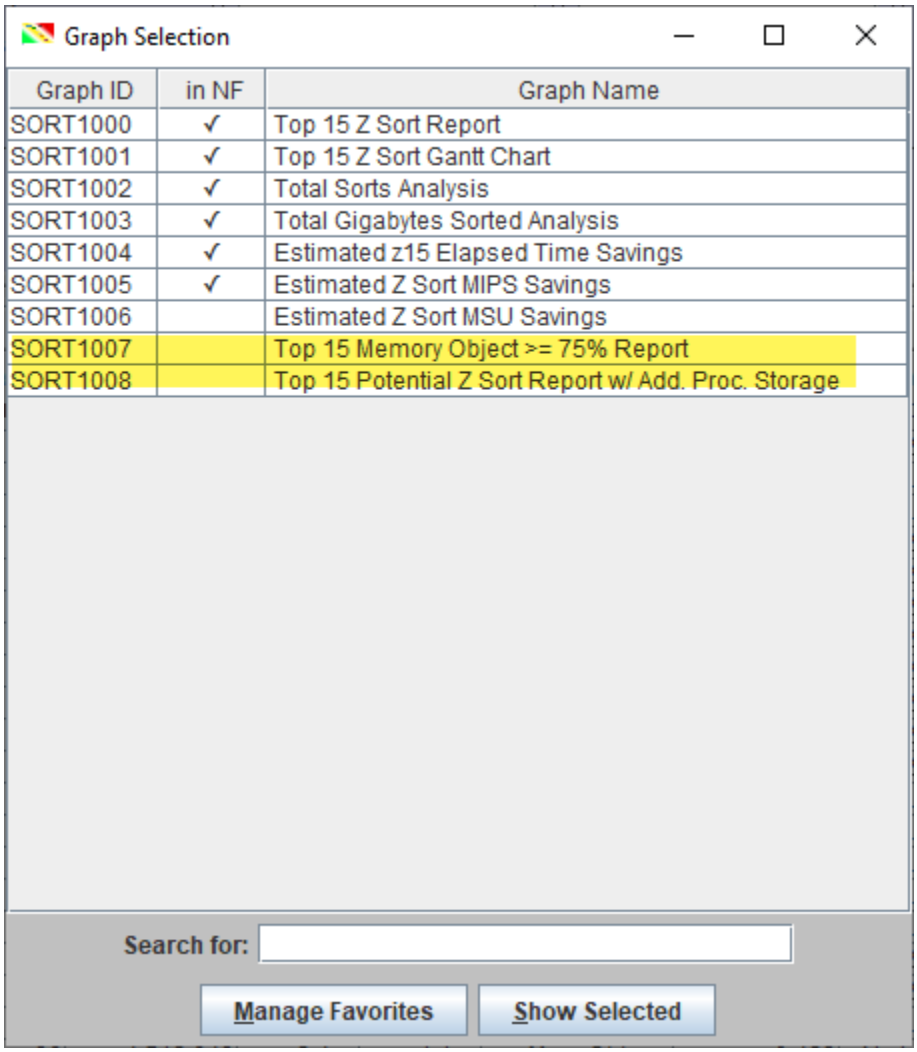
The zBNA chart SORT1003 update has a significant impact as it shows the total bytes sorted (GB) for all the IBM Z Sort candidates, including the SortWork ones. zBNA adds up the GB sorted for any IBM Z Sort candidate sort ended in the hour. It assumes all the sorts ran at the same time in the hour they ended and thus all had their processor storage requirement at the same time (this is a very conservative assumption, and probably unlikely). One could use zBNA chart SORT1003 as a "ballpark" crude estimate of how much additional processor storage is required by DFSORT

DFSORT recommends planning for ~2x the Bytes Sorted to estimate the Memory Object size. So, for a crude estimation, using 2x the size of the peak GB Storage would be the system requirement in Processor Storage to support all the eligible sorts. In the example above it appears the peak is 1.5 TB (1,500 GB at 4 AM on Sept 15) of processor storage, and if you 2x that amount it would be over 3 TB. Processors today can support a very large amount of storage. The z15 T01 can support 40 TB of processor storage and the z15 T02 can support 16 TB of processor storage. The maximum a z/OS LPAR can support (z/OS V2R4) is 4 TB. This seems like an extraordinary amount of processor storage to configure (and it probably is), but they key point is to understand just how big your eligible IBM Z Sort candidate Bytes Sorted are, and consider adding or reconfiguring processor storage to the LPARs with the sort workloads which could utilize IBM Z Sort.

In the example above we would suggest a "bottoms up" approach by looking at the sorts in zBNA chart SORT1008 to see which ones you want to improve. Then look when they run and if they are concurrent. Then review how big the sorts are and ensure you have ~ 2x their GB storage available processor storage in the timeframe they run.

That is probably a reasonable approach to get started. Then you can individually enable IBM Z Sort for the specific sorts (via the SYSIN or DFSPARM DD statements. To enable you can specify OPTION ZSORT in the SORT control statement). You can validate the results and continue to implement Z Sort for other sorts where you want to improve elapsed time and reduce CPU time. The key point is to think big on processor storage as DFSORT Z Sort can exploit it to improve both elapsed time and reduce CPU.

zBNA V2.2.4 became available in late May 2021 and further added capability to identify "constrained" IBM Z Sort candidates. zBNA V2.2.4 added the new capability to identity sorts which could be IBM Z Sort candidates but are currently using SORTWK, not memory objects. zBNA provides 2 new reports: SORT1007 and SORT1008



SORT1007 shows those Sorts that are IBM Z Sort candidates with more than 75% of the file size fitting in a memory object. The benchmark metrics are not available to estimate the benefits of converting these sorts to IBM Z Sort. So, like the candidates with 100% in Memory Objects today (Top 15 IBM Z Sort report) these are also candidates, but zBNA cannot provide estimates of benefits. You could use this report to identify how much more Memory Object size is required to fit 100% within a Memory Object.

If the objective is to try and identify additional IBM Z Sort candidates IF additional resources were made available, then SORT1008 should be analyzed. This shows the IBM Z Sort candidates (all DFSORT selection criteria met) except they did not utilize a Memory Object (100% SortWork), or they did not fit 75% into a Memory Object. In both cases they did not meet the IBM Z Sort for that one criteria. So, IF the environment could be altered to use a Memory Object then they would use Z Sort.

The information in SORT1008 can be useful in the analysis. It is sorted in GB Sorted descending. This often results in the largest sorts with the longest elapsed time and most CPU time. If one is looking to maximize the benefit from IBM Z Sort, this is your top candidate list. There is also useful information, like Memory Object Used (GB) and Memory Limit (GB). The Memory Object Used is probably small (relative to the GB Sorted) or 0 since these are the Top SWK IBM Z Sort eligible sorts. The Memory Limit is the virtual storage above the bar limit for the sort. Comparing this value to the GB Sorted can be useful in determining why a Memory Object was not used. If the Memory Limit is, for example, 2 GB, and the GB Sorted significantly higher, then the Memory Limit may be the reason it did not use a Memory Object. In addition, there would have to be processor storage available, and DFSORT settings configured to exploit the processor storage and IBM Z Sort.

It is also important for DFSORT to use as much processor storage as possible, and it is controlled by the EXP* parm values and of course available processor storage which can be utilized by Memory Objects. DFSORT can control access by DAY/ TOD to its parms including EXP*, so it is possible to create custom settings to allow access to more processor storage in the Batch window. See our recommendations above.

In summary, you need three requirements to exploit DFSORT Z Sort:

1. The Job/Step needs access to the virtual storage above the bar (this is controlled by the MEMLIMIT setting)
2. DFSORT needs to be able to exploit processor storage and Memory Objects (EXP values and MOSIZE settings)
3. Have processor storage available to be exploited. (RMF – Paging Report will show Min and Max available processor storage and the pageable 1 MB frames DFSORT uses for Z Sort Memory Objects).

WSC DFSORT IBM Z Sort Test Jobs

Sort test jobs were run on an IBM z15 processor. The tests jobs ran the same 44GB sort multiple times alongside varied workloads that impacted the amount of available memory on the system. For the tests, the DFSORT defaults were set to optimally exploit available memory:

- EXPMAX=MAX, EXPOLD=0, EXPRES=10%
- MOSIZE=MAX, HIPRMAX=OPTIMAL, DSPSIZE=MAX

This system had a default MEMLIMIT of 20GB so in some cases the jobs were limited. I In other jobs, we added REGION=0M to the JCL which causes MEMLIMIT=NOLIMIT to be in effect. Also set was the default of ZSORT=YES except for the last test case with ZSORT=NO to demonstrate a case where a sort completed entirely in memory but was still unable to exploit IBM Z Sort. zEDC compression was used for all SORTIN and SORTOUT files.

Using RMF, we were able to show the maximum available memory during the interval in which each job executed. RMF was set to report on 5-minute intervals and we only ran a single test case in each interval. In the results below, you'll notice when the available memory was more than the 20 GB MEMLIMIT, DFSORT chose Hiperspace since it could use more memory via that method (up to the DFSORT limitation of 32 GB).

Id	End	Job Name	Job Number	RMF Max GB Avail	MEMLIMIT	Z Sort Enabled	Z Sort Used	Memory Used (GB)	Memory Type	Work I/O	Elapsed Time (Sec)	CPU Time (Sec)	Elapsed Improvement %	CPU Improvement %
1	14:06:57	CUST05WK	JOB25358	1.63	NOLIMIT	Y	N	0.00	none	181,100	341	32.96	0.0%	0.0%
2	14:14:19	CUST05MW	JOB25360	24.69	NOLIMIT	Y	N	21.34	MOWRK	109,393	233	43.50	31.7%	-32.0%
3	14:21:06	CUST05MW	JOB25361	24.68	20GB	Y	N	21.23	Hiperspace	221,982	349	52.89	-2.3%	-60.5%
4	14:29:23	CUST05ZW	JOB25364	40.60	20GB	Y	N	32.00	Hiperspace	125,355	258	61.86	24.3%	-87.7%
5	14:33:48	CUST05ZW	JOB25366	40.56	NOLIMIT	Y	Y	35.72	Z Sort MO / Swk	58,564	220	44.78	35.5%	-35.9%
6	14:44:17	CUST05ZS	JOB25367	61.18	20GB	Y	N	32.00	Hiperspace	125,355	251	61.77	26.4%	-87.4%
7	14:45:43	CUST05ZS	JOB25368	61.69	NOLIMIT	Y	Y	45.45	Z Sort MO	64	39	32.32	88.6%	1.9%
8	14:51:30	CUST05ZS	JOB25370	61.19	NOLIMIT	N	N	47.08	MO	64	80	51.89	76.5%	-57.4%

IBM Z Sort was only used in the cases where available memory was large enough, MEMLIMIT did not restrict the use of 64-bit memory and IBM Z Sort was enabled. In most cases the use of Hiperspace or memory objects, resulted in reduced elapsed time compared to the first test case that used only disk work files. However, this change resulted in an increase in CPU cost. Only in the IBM Z Sort in memory case (**Id 7**) did both a reduction in elapsed time and a CPU savings occur.

For the first 2 jobs (**Ids 1 and 2**), there was not enough processor storage available to DFSORT (by design, as other jobs consumed processor storage). **Id 1** had the least amount of processor storage available, and it used no memory option, and used a traditional Sort Work sort. With **Id 2**, more processor storage (about 25 GB) was available, a Memory Object with SortWork was utilized. IBM Z Sort was not used

because only 21.34 GB / 44 GB = 49% of the file size would fit in the Memory Object. IBM Z Sort requires =>75% to fit in a Memory Object.

The next jobs (**Ids 3 and 4**) had the MEMLIMT constrained to 20 GB, DFSORT utilized Hiperspace, since it could utilize up to 32 GB with that option, as discussed above. With more processor storage (about 40 GB) available to **ID 4**, it used up to the 32 GB for Hiperspace, reduced SortWork I/Os and ran faster than **ID 3**.

For job **Id 5**, with still about 40 GB processor storage available, when the MEMLIMT constraint was removed, DFSORT selected the IBM Z Sort Memory Object / SortWork path. In this case 35.72 GB / 44 GB = 81% of the file size fit into a Memory Object. The elapsed time and CPU time both decreased relative to the Hiperspace job (**Id 4**).

The available processor storage continued to increase to about 61 GB when Job **Id 6** ran. However, the MEMLIMIT was set back to 20 GB. So DFSORT used Hiperspace (32 GB). Notice it achieved essentially the same results as job **ID 4** in elapsed and CPU time. The MEMLIMIT setting constrained DFSORT, even though more processor storage was available.

Job **Id 7** was finally able to execute with about 61 GB processor storage, unconstrained MEMLIMIT, IBM Z Sort enabled, and the entire file size (44 GB) fit in a Memory Object, so DFSORT executed with a IBM Z Sort Memory Object. Notice it had the lowest elapsed time and lowest CPU time of any of the runs.

Job **Id 8** was essentially the same environment as Job **Id 7**, except that IBM Z Sort was disabled by the JCL SYSIN Sort Control Option NOZSORT. While job **ID 8** completed entirely in a Memory Object it did not use Z Sort, and it resulted in higher elapsed time and higher CPU time. Job **ID 8** will later show up as a zBNA candidate (SORT1001) as it resides entirely in a Memory Object.

zBNA Results for WSC DFSORT IBM Z Sort Test Jobs

Once the tests were completed, we analyzed the jobs using zBNA V2.2.4

zBNA Chart SORT1001

In this limited scope of 8 jobs, only 1 job (**ID 8**) fits entirely in a Memory Object. It was the last job in our test. zBNA provides estimates of potential elapsed and CPU savings. The actual elapsed time savings was less than predicted but the CPU savings was very close (to job **ID 7**). Overall job **Id 7** (IBM Z Sort enabled) still had an actual 51.3% savings in job elapsed time and 37.7% savings in job CPU time. This underscores the importance of having all resources and settings to allow DFSORT to maximize IBM Z Sort benefits.

Id	Job Name	Step	GB Sorted	Elapsed Time	CPU Time Seconds	Location	Memory Obj Used (GB)	Memory Limit (GB)	Record Format	Est. Z Sort on z15	
										Δ Elapsed Time	Δ CPU Time Seconds
8	CUST05ZS	F1	44.340	79.5s	51.48	Mem Obj	47.085	No Limit	FL	-57.4s	-19.85
	Total		44.340	79.5s	51.48					-57.4s (-72.2%)	-19.85 (-38.6%)

SORT1001

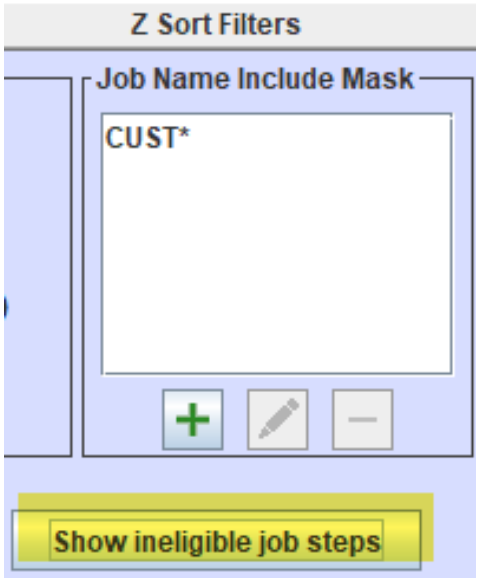
Top 1 Z Sort Eligible Sorts - SYSD

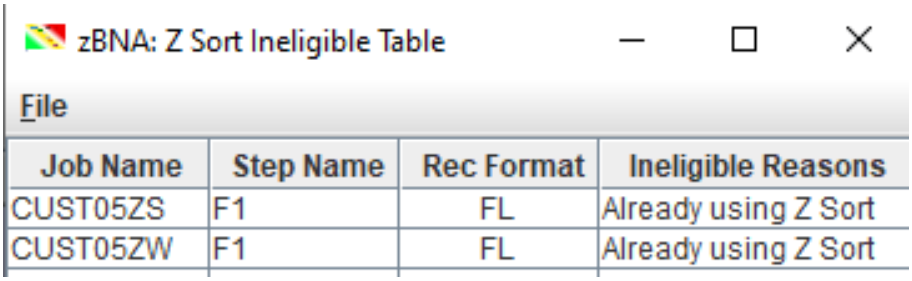
The top 1 Sort Job Steps by GB sorted, according to the user applied filters, are listed in the following table.

Id	Job Name	Step	GB Sorted	Elapsed Time	CPU Time Seconds	Location	Memory Obj Used (GB)	Memory Limit (GB)	Record Format	Est. Z Sort on z15	
										Δ Elapsed Time	Δ CPU Time Seconds
8	CUST05ZS	F1	44.340	79.5s	51.48	Mem Obj	47.085	No Limit	FL	-57.4s	-19.85
	Total		44.340	79.5s	51.48					-57.4s (-72.2%)	-19.85 (-38.6%)

IBM Z Sort Ineligible Jobs – Including those already using Z Sort

On the main panel under the filters there is a button to “Show Ineligible job steps” sorts. These include the exceptions to all the DFSORT requirements, so you can see the reasons. In the case of our tests, the only reason was if a sort was already using Z Sort. zBNA does not double count those sorts. Below are the 2 jobs that used Z Sort, Id 7 (CUST05ZS) and Id 5 (CUST05ZW).





Job Name	Step Name	Rec Format	Ineligible Reasons
CUST05ZS	F1	FL	Already using Z Sort
CUST05ZW	F1	FL	Already using Z Sort

zBNA Chart SORT1008

Top 5 Potential Z Sort Eligible Sorts SWK w/ Add. Memory - SYSD

The Top 15 Potential Sort Job Steps w/ SWK by GB sorted, according to the user applied filters, are listed in the following table.

Job Name	Step	GB Sorted	Elapsed Time	CPU Time Seconds	Location	Memory Obj Used (GB)	Memory Limit (GB)	Record Format	ID
CUST05MW	F1	44.340	232.5s	43.01	Mem Obj SWK	21.337	No Limit	FL	2
CUST05MW	F1	44.340	348.3s	52.32	Hiperspace SWK	0.000	20	FL	3
CUST05ZW	F1	44.340	257.3s	61.37	Hiperspace SWK	0.000	20	FL	4
CUST05ZS	F1	44.340	250.0s	61.27	Hiperspace SWK	0.000	20	FL	6
CUST05WK	F1	44.340	341.1s	32.39	SWK	0.000	No Limit	FL	1
Total		221.701	23.8m	250.36					

Here are the remaining 5 of the 8 test jobs. One is a candidate, and 2 are already using IBM Z Sort. The zBNA chart SORT1008 represents the most opportunity for identifying the IBM Z Sort benefit.

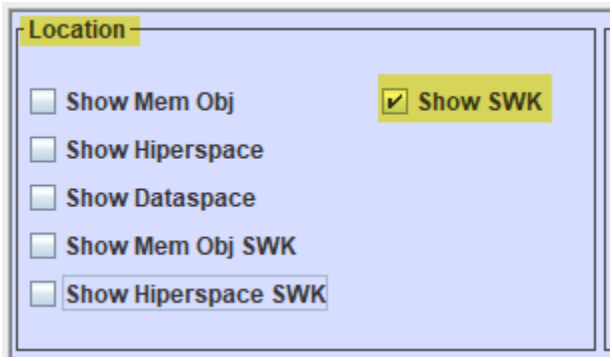
First identify the Job / sort you want to improve. Then look for the potential inhibitors. In the above, Ids 3, 4 and 6 were limited by MEMLIMIT. Ids 2 and 1 were limited by the lack of available processor storage. Id 2 had more processor storage and already was using a part of a Memory Object (51%), but not enough to get to 75% of the file size with a Memory Object for Z Sort.

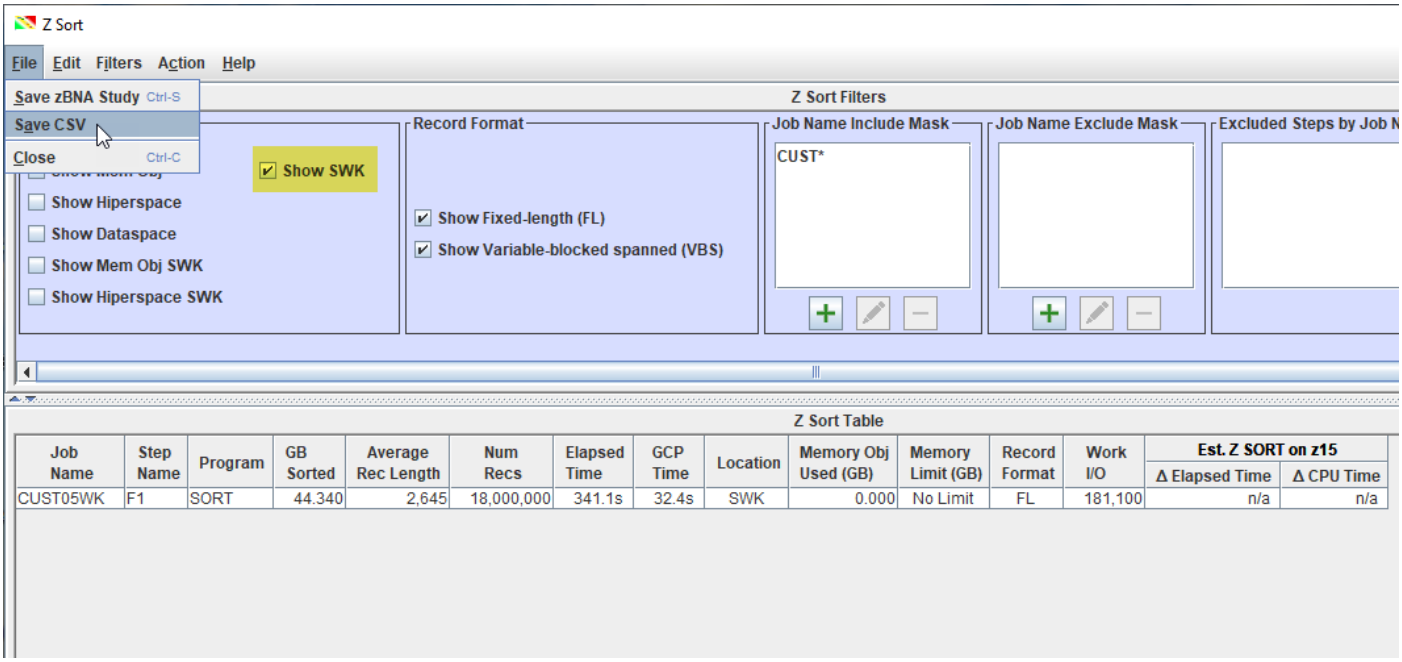
Analyze RMF reports to understand how much available processor storage is available. If more is required, can it be obtained from other LPARs or should it be acquired.

Finally, remember to review the DFSORT Installation options to ensure you can utilize all the processor storage.

zBNA SORT1008 – Additional Depth

zBNA chart SORT1008 only produces the Top 15 Sorts with SortWork. If you want a deeper list, you can go back to the zBNA Z Sort main panel and select only “Show SWK” on the Location filter. This will then display only that subset. From there you can save the output as a CSV. In the tests there was only one candidate, **Id 1**.





Metrics to Identify IBM Z Sort Candidates and Resources

This section will describe several sources useful in identifying RMF Processor Storage and DFSORT Installation Options.

RMF Post Processor – Paging Report

The RMF paging report can be used to analyze the available memory on your system as well as the use of fixed and pageable large pages. The following JCL can be used to process SMF type 71 records with the RMF post processor to create the report.

```
//RMFPP    EXEC PGM=ERBRMFPP,REGION=0M
//MFPINPUT DD DISP=(SHR),DSN=your.input.smf.dsn
//MFPMSGDS DD   SYSOUT=*
//SYSIN    DD   *
  SYSOUT(O)
  DINTV(0005)
  REPORTS(PAGING)
/*
```

Paging Report

In the sample paging report below the maximum available frames was 6,544,599. There are 262,144 4K frames per 1 GB of memory. So dividing 6,544,599 by 262,144 we calculate 24.96 GB was the maximum available memory. When our test sort ran (Id 2), it used almost all of that memory which is why the minimum available frames during the interval was only 920,249 frames / 261,444 = 3.5 GB.

```
P A G I N G   A C T I V I T Y
                                                                                                                         PAGE    2
          z/OS V2R4               SYSTEM ID SYSD            START 06/23/2021-14.10.00  INTERVAL 000.05.00
                                  RPT VERSION V2R4 RMF      END   06/23/2021-14.15.00  CYCLE 1.000 SECONDS
OPT = IEAOPT00                         CENTRAL STORAGE MOVEMENT AND REQUEST RATES - IN PAGES PER SECOND
-----------------------------------------------------------------------------------------------------------------------------------
SYSTEM UIC: MIN =  65535    MAX =  65535    AVG =  65535
CENTRAL STORAGE   PAGE WRITE   PAGE READ   -------- FRAME COUNTS ---------
----------------  -- RATE --  -- RATE --   -- MIN --  -- MAX --  -- AVG --
    HIPERSPACE         0.00        0.00          1          1          1
    VIO                0.00        0.00          1          1          1
                  ------- GETMAIN -------  -------- FIXED --------  --- REF FAULTS ---
STORAGE REQUESTS  REQUESTS  FRAMES BACKED  REQ < 2GB  FRAMES < 2GB       1ST   NON-1ST
    RATE            754.60          48.58       3.17      47,577.7  18,932.3      0.00
                                                   FRAME AND SLOT COUNTS
-----------------------------------------------------------------------------------------------------------------------------------
(31 SAMPLES)
CENTRAL STORAGE FRAMES        TOTAL  AVAILABLE        SQA        LPA        CSA       LSQA  REGIONS+SWA  HV SHARED  HV COMMON
----------------------    ---------  ---------  ---------  ---------  ---------  ---------  -----------  ---------  ---------
      MIN                  60817408    920,249      9,513      6,416      8,693     56,169     54110957      4,285     70,228
      MAX                  60817408  6,544,599      9,521      6,416      8,710     56,299     59735181      4,285     70,228
      AVG                  60817408  3,249,013      9,516      6,416      8,703     56,259     57406452      4,285     70,228
FIXED FRAMES                  TOTAL    NUCLEUS        SQA        LPA        CSA       LSQA  REGIONS+SWA     <16 MB   16MB-2GB
----------------------    ---------  ---------  ---------  ---------  ---------  ---------  -----------  ---------  ---------
      MIN                   308,422      3,623      8,732        120     56,457     12,523      226,950         60     16,615
      MAX                   331,545      3,623      8,740        121     56,457     12,624      249,992         60     16,736
      AVG                   324,987      3,623      8,734        120     56,457     12,599      243,452         60     16,665
SHARED FRAMES / SLOTS         TOTAL      CENTRAL STORAGE  FIXED TOT  FIXED BEL       HV 1M        HV 4K  AUX DASD    AUX SCM
----------------------    ---------  ---------  ---------  ---------  ---------  ---------  -----------  ---------  ---------
      MIN                 268573714                11,726         11          1          0        6,840          0          0
      MAX                 268573714                11,726         11          1          0        6,840          0          0
      AVG                 268573714                11,726         11          1          0        6,840          0          0
LOCAL PAGE DATA SET SLOTS     TOTAL  AVAILABLE        BAD    NON-VIO        VIO
------------------------  ---------  ---------  ---------  ---------  ---------
      MIN                   179,999    179,995          0          0          4
      MAX                   179,999    179,995          0          0          4
      AVG                   179,999    179,995          0          0          4
SCM PAGING BLOCKS             TOTAL  AVAILABLE        BAD     IN-USE
----------------------    ---------  ---------  ---------  ---------
      MIN                  67108864   67045853          0     63,011
      MAX                  67108864   67045853          0     63,011
      AVG                  67108864   67045853          0     63,011
```

During the interval when IBM Z Sort was used to complete the sort entirely in memory (Id 7), the paging report shows a much larger amount of memory with a maximum available frames of 16,111,638 / 262,144 = 61.5 GB. DFSORT allocated a large memory object to complete the IBM Z Sort entirely in memory which is why the minimum available frames was 7,748,410 / 262,144 = 29.5GB.

```
                                        P A G I N G   A C T I V I T Y
                                                                                                                    PAGE    2
          z/OS V2R4                SYSTEM ID SYSD             START 06/23/2021-14.45.00  INTERVAL 000.05.00
                                   RPT VERSION V2R4 RMF       END   06/23/2021-14.50.00  CYCLE 1.000 SECONDS
OPT = IEAOPT00                          CENTRAL STORAGE MOVEMENT AND REQUEST RATES - IN PAGES PER SECOND
-------------------------------------------------------------------------------------------------------------------------------
SYSTEM UIC: MIN =  65535    MAX =  65535    AVG =  65535
CENTRAL STORAGE   PAGE WRITE   PAGE READ   -------- FRAME COUNTS ---------
---------------  -- RATE --  -- RATE --   -- MIN --  -- MAX --  -- AVG --
     HIPERSPACE         0.00        0.00          1          1          1
     VIO                0.00        0.00          1          1          1
                  ------- GETMAIN -------  -------- FIXED --------  --- REF FAULTS ---
STORAGE REQUESTS  REQUESTS  FRAMES BACKED  REQ < 2GB  FRAMES < 2GB       1ST   NON-1ST
     RATE           760.30          49.09       3.30        273.93    248.37      0.00
                                              FRAME AND SLOT COUNTS
-------------------------------------------------------------------------------------------------------------------------------
(31 SAMPLES)
CENTRAL STORAGE FRAMES         TOTAL  AVAILABLE       SQA        LPA        CSA       LSQA  REGIONS+SWA  HV SHARED  HV COMMON
----------------------     ---------  ---------  ---------  ---------  ---------  ---------  -----------  ---------  ---------
      MIN                   60817408  7,748,410      9,479      6,416      8,697     56,172     44543942      4,285     70,229
      MAX                   60817408   16111638      9,495      6,416      8,710     56,237     52907116      4,285     70,229
      AVG                   60817408   15365000      9,488      6,416      8,703     56,192     45290560      4,285     70,229
FIXED FRAMES                   TOTAL    NUCLEUS       SQA        LPA        CSA       LSQA  REGIONS+SWA     <16 MB   16MB-2GB
----------------------     ---------  ---------  ---------  ---------  ---------  ---------  -----------  ---------  ---------
      MIN                     257,108      3,623      8,698        120     56,457     12,534      175,665         60     14,788
      MAX                     257,819      3,623      8,714        120     56,457     12,552      176,369         60     14,941
      AVG                     257,245      3,623      8,706        120     56,457     12,538      175,800         60     14,851
SHARED FRAMES / SLOTS          TOTAL       CENTRAL STORAGE  FIXED TOT  FIXED BEL      HV 1M        HV 4K   AUX DASD    AUX SCM
----------------------     ---------  ---------  ---------  ---------  ---------  ---------  -----------  ---------  ---------
      MIN                  268573714                11,726         11          1          0        6,840          0          0
      MAX                  268573714                11,726         11          1          0        6,840          0          0
      AVG                  268573714                11,726         11          1          0        6,840          0          0
LOCAL PAGE DATA SET SLOTS      TOTAL  AVAILABLE        BAD    NON-VIO        VIO
-------------------------  ---------  ---------  ---------  ---------  ---------
      MIN                     179,999    179,995          0          0          4
      MAX                     179,999    179,995          0          0          4
      AVG                     179,999    179,995          0          0          4
SCM PAGING BLOCKS              TOTAL  AVAILABLE        BAD     IN-USE
----------------------     ---------  ---------  ---------  ---------
      MIN                   67108864   67086772          0     22,092
      MAX                   67108864   67086772          0     22,092
      AVG                   67108864   67086772          0     22,092
```

On page 3 of this same RMF paging report we can see the use of pageable 1 MB frames by DFSORT when using IBM Z Sort (46,795 1 MB frames). Since we're now dealing with 1MB frames we merely divide by 1,024 to convert that to 45.7 GB.

```
          z/OS V2R4              SYSTEM ID SYSD            START 06/23/2021-14.45.00  INTERVAL 000.05.00
                                 RPT VERSION V2R4 RMF      END   06/23/2021-14.50.00  CYCLE 1.000 SECONDS
OPT = IEAOPT00                        MEMORY OBJECTS AND HIGH VIRTUAL STORAGE FRAMES
-------------------------------------------------------------------------------------------------------------------
LFAREA                MAXIMUM
------------------    ---------
1 MB FRAMES               26G
2 GB FRAMES               24G
MEMORY OBJECTS       FIXED 1M   FIXED 2G     COMMON     SHARED  SHARED 1M
------------------   ---------  ---------  ---------  ---------  ---------
      MIN                   6          0        139         17          0
      MAX                   6          0        139         17          0
      AVG                   6          0        139         17          0
1 MB FRAMES          ------------ FIXED ------------   PAGEABLE  AVAILABLE      TOTAL
------------------     MAXIMUM  AVAILABLE     IN-USE  ---------  ---------  ---------
      MIN               25,804     18,752         14        272     18,752    231,251
      MAX               25,804     25,790         14     46,795     51,662    231,251
      AVG               25,804     25,336         14      4,521     48,624    231,251
2 GB FRAMES          ------------ FIXED ------------
------------------     MAXIMUM  AVAILABLE     IN-USE
      MIN                   12         12          0
      MAX                   12         12          0
      AVG                   12         12          0
HIGH SHARED FRAMES       TOTAL       CENTRAL STORAGE  BACKED 1M                        AUX DASD    AUX SCM
------------------   ---------       --------------  ---------                       ---------  ---------
      MIN            136902.1M                4,285          0                               0         40
      MAX            136902.1M                4,285          0                               0         40
      AVG            136902.1M                4,285          0                               0         40
HIGH COMMON FRAMES       TOTAL       CENTRAL STORAGE  BACKED 1M      FIXED   FIXED 1M   AUX DASD    AUX SCM
------------------   ---------       --------------  ---------  ---------  ---------  ---------  ---------
      MIN             17301504               70,229        188      9,150         14          0         11
      MAX             17301504               70,229        188      9,150         14          0         11
      AVG             17301504               70,229        188      9,150         14          0         11
```

RMF Overview Reports – Available and Used

RMF overview reports can be useful to show available and used memory over time. This sample JCL shows the online, maximum available and minimum available frame counts along with maximum Hiperspace usage and average pageable 1 MB frames usage.

```
//RMFPP    EXEC PGM=ERBRMFPP,REGION=0M
//MFPINPUT DD DISP=(SHR),DSN=your.input.smf.dsn
//MFPMSGDS DD   SYSOUT=*
//SYSIN    DD   *
  SYSOUT(O)
 OVW(ONLINE(STORAGE))
  OVW(MAXAVAIL(CSTORAVX))
  OVW(MINAVAIL(CSTORAVM))
  OVW(MAXHIP(RSHSPX))
  OVW(PGBL1M(LPFRPX))
/*
```

This creates an overview report showing the values for each RMF interval.

```
                                        R M F   O V E R V I E W   R E P O R T

PAGE  001
           z/OS  V2R4              SYSTEM ID SYSD            START 06/23/2021-14.00.00  INTERVAL 00.05.00
                                   RPT VERSION V2R4 RMF      END   06/23/2021-14.55.00  CYCLE 1.000 SECONDS

NUMBER OF INTERVALS 11           TOTAL LENGTH OF INTERVALS 00.55.00
DATE    TIME     INT       ONLINE  MAXAVAIL  MINAVAIL    MAXHIP    PGBL1M
MM/DD HH.MM.SS HH.MM.SS
06/23 14.00.00 00.04.59   60817408    426578    404302         1       272
06/23 14.05.00 00.04.59   60817408   6472721    404276         1       272
06/23 14.10.00 00.05.00   60817408   6472715    848361         1       272
06/23 14.15.00 00.04.59   60817408   6470525    887516   1265137       272
06/23 14.20.00 00.05.00   60817408  59063744   4772788   1638425       272
06/23 14.25.00 00.04.59   60817408  10644055   2218460                 272
06/23 14.30.00 00.05.00   60817408  10633622   1281140         1     25169
06/23 14.35.00 00.04.59   60817408  16039219   7092173         1       272
06/23 14.40.00 00.05.00   60817408  16039219   7622537                 272
06/23 14.45.00 00.05.00   60817408  16039758   7676526         1      4521
06/23 14.50.00 00.04.59   60817408  16039758   4088670         1       272
```

DFSORT Installation Options

In DFSORT it may be useful to see what installation options are currently in effect. One can run the ICETOOL with the following options:

```
//LISTDEF EXEC PGM=ICETOOL
//TOOLMSG DD SYSOUT=*
//DFSMSG DD SYSOUT=*
//SHOWDEF DD SYSOUT=*
//TOOLIN DD *
  DEFAULTS LIST(SHOWDEF)
/*
```

On the next page is a sample of the output which is written to the SHOWDEF DD. In this case we have the recommended settings as discussed above. Note that the DFSORT default is listed below with an *, so you can see we've set EXPOLD to 0%, from the 50% default, and you can see we've set ZSORT to be enabled "YES" vs it being disabled "NO". This can be very useful to understand what options are in effect in your environment.

```
1Z/OS DFSORT V2R4  MERGED PARMLIB/ICEMAC DEFAULTS     - 1 -

   * IBM-SUPPLIED DEFAULT (ONLY SHOWN IF DIFFERENT FROM THE SPECIFIED DEFAULT)
```

ITEM	JCL (ICEAM1) VALUE	INV (ICEAM2) VALUE	TSO (ICEAM3) VALUE	TSOINV (ICEAM4) VALUE
ENABLE	NONE	NONE	NONE	NONE
ABCODE	MSG	MSG	MSG	MSG
ALTSEQ	SEE BELOW	SEE BELOW	SEE BELOW	SEE BELOW
ARESALL	0	0	0	0
ARESINV	NOT APPLICABLE	0	NOT APPLICABLE	0
CFW	YES	YES	YES	YES
CHALT	NO	NO	NO	NO
CHECK	YES	YES	YES	YES
CINV	YES	YES	YES	YES
COBEXIT	COB2	COB2	COB2	COB2
COLLKEY	UCA600	UCA600	UCA600	UCA600
DIAGSIM	NO	NO	NO	NO
DSA	128	128	128	128
DSPSIZE	MAX	MAX	MAX	MAX
DYNALOC	(SYSDA,4)	(SYSDA,4)	(SYSDA,4)	(SYSDA,4)
DYNAPCT	10	10	10	10
DYNAUTO	YES	YES	YES	YES
DYNSPC	256	256	256	256
EFS	NONE	NONE	NONE	NONE
EQUALS	VLBLKSET	VLBLKSET	VLBLKSET	VLBLKSET
ERET	RC16	RC16	RC16	RC16
ESTAE	YES	YES	YES	YES
EXITCK	STRONG	STRONG	STRONG	STRONG
EXPMAX	MAX	MAX	MAX	MAX
EXPOLD	0	0	0	0
	* 50%	* 50%	* 50%	* 50%
EXPRES	10%	10%	10%	10%
FSZEST	NO	NO	NO	NO
GENER	NOT APPLICABLE	IEBGENR	NOT APPLICABLE	IEBGENR
GNPAD	NOT APPLICABLE	RC0	NOT APPLICABLE	RC0
GNTRUNC	NOT APPLICABLE	RC0	NOT APPLICABLE	RC0
HIPRMAX	OPTIMAL	OPTIMAL	OPTIMAL	OPTIMAL
IDRCPCT	NONE	NONE	NONE	NONE
IEXIT	NO	NO	NO	NO
IGNCKPT	YES	YES	YES	YES
IOMAXBF	35651584	35651584	35651584	35651584
LIST	YES	YES	YES	YES
LISTX	YES	YES	YES	YES
LOCALE	NONE	NONE	NONE	NONE
MAXLIM	1048576	1048576	1048576	1048576
MINLIM	450560	450560	450560	450560
MOSIZE	MAX	MAX	MAX	MAX
MOWRK	YES	YES	YES	YES
MSGCON	NONE	NONE	NONE	NONE
MSGDDN	SYSOUT	SYSOUT	SYSOUT	SYSOUT
MSGPRT	ALL	ALL	ALL	ALL
NOMSGDD	QUIT	QUIT	QUIT	QUIT
NULLOFL	RC0	RC0	RC0	RC0
NULLOUT	RC0	RC0	RC0	RC0
ODMAXBF	2097152	2097152	2097152	2097152
OUTREL	YES	YES	YES	YES
OUTSEC	YES	YES	YES	YES
OVERRGN	65536	16384	65536	16384
OVFLO	RC0	RC0	RC0	RC0
PAD	RC0	RC0	RC0	RC0
PARMDDN	DFSPARM	DFSPARM	DFSPARM	DFSPARM
RESALL	4096	4096	4096	4096
RESET	YES	YES	YES	YES
RESINV	NOT APPLICABLE	0	NOT APPLICABLE	0
SDB	INPUT	INPUT	INPUT	INPUT
SDBMSG	NO	NO	NO	NO
SIZE	MAX	MAX	MAX	MAX
SMF	FULL	NO	NO	NO
	* NO			
SOLRF	YES	YES	YES	YES
SORTLIB	PRIVATE	PRIVATE	PRIVATE	PRIVATE
SPANINC	RC16	RC16	RC16	RC16
SVC	109	109	109	109
SZERO	YES	YES	YES	YES
TEXIT	NO	NO	NO	NO
TMAXLIM	6291456	6291456	6291456	6291456
TRUNC	RC0	RC0	RC0	RC0
TUNE	STOR	STOR	STOR	STOR
VERIFY	NO	NO	NO	NO
VIO	NO	NO	NO	NO
VLLONG	NO	NO	NO	NO
VLSCMP	NO	NO	NO	NO
VLSHRT	NO	NO	NO	NO
VSAMBSP	OPTIMAL	OPTIMAL	OPTIMAL	OPTIMAL
VSAMEMT	YES	YES	YES	YES
VSAMIO	NO	NO	NO	NO
WRKREL	YES	YES	YES	YES
WRKSEC	YES	YES	YES	YES
Y2PAST	80	80	80	80
ZDPRINT	YES	YES	YES	YES
ZSORT	YES	NO	NO	NO
	* NO			

Summary

In summary, IBM Z Sort can provide benefits with reduced elapsed time and reduced CPU time for eligible DFSORT sorts. Configuring the environment to 1) provide the virtual storage for jobs/steps, 2) allow DFSORT to exploit processor storage and memory objects and 3) provide ample processor storage to allow DFSORT to exploit it, are all important requirements to successfully exploit IBM Z Sort.

If you were an early of zBNA Z Sort application, you may want to rerun the tool to identify other potential candidates.

Special Notices

This publication is intended to discuss the behavior observed within a uniquely defined system environment in effort to understand the system behavior when utilizing DFSORT, z/OS and z15 IBM Z Sort within a uniquely defined test system. The information in this publication is not intended as the specification of any programing interfaces provided by DFSORT or z/OS. See the publication section of the IBM programming announcement for the appropriate DFSORT, z/OS and z15 releases for more information about which publications are considered to be product documentation. Where possible it is recommended to follow-up with product related publications to understand the specific impact of the information documented in this publication.

The information contained in this document has not been submitted to any formal IBM test and is distributed on an “as is” basis without any warranty either expressed or implied. 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 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.

Performance data contained in this document was determined in a controlled environment; therefore, the results which may be obtained in other operating environments may vary significantly. No commitment as to your ability to obtain comparable results is any intended or made by this release of information.

Appendix

Here are additional resources:

DFSORT User Guide for IBM Integrated Accelerator for IBM Z Sort (PH03207)

https://www.ibm.com/support/pages/node/6335819

This is an excellent White Paper that describes Z Sort. It documents the important enhancement of DFSORT and DFSORT's ICETOOL which are provided by z/OS DFSORT V2R3 PTF UI90067 and DFSORT V2R4 PTF UI90068. This enhancement exploits a new sort accelerator chip known as the IBM Integrated Accelerator for Z Sort.

Also for DFSORT information visit for ICETOOL papers, examples and more

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