IBM Technical Brief

IBM zEnterprise System®: DB2 10 for z/OS Large Objects with SAP® Insurance Solutions

Authors:

Seewah Chan
Paul D. DeLessio
Dr. Paul Lekkas
Rose L. Manz
Michael R. Sheets

Document Owner:

Paul D. DeLessio
IBM SAP Performance Center for System z
IBM Poughkeepsie, US

Version: 1.0
Date: May 14, 2014
Filename: System_z_DB2_LOB_SAP_Ins.pdf

Contents

Tables

Figures

Disclaimers

Trademarks

Acknowledgements

The authors of this document would like to acknowledge the following individuals for their contributions to this project:

- Brenda Beane (IBM SAP on System z)
- Akiko Hoshikawa (DB2 for z/OS Development Performance)

Feedback

Please send comments or suggestions for changes to delessio@us.ibm.com

1.0 Introduction

Database applications need to store items such as large text documents, audio, video, drawings and images. To support this need, IBM's DB2 for z/OS introduced Large Object (LOB) support in Version 6. This initial support was "outline". That is, LOB data is not physically stored in the base table space with non-LOB columns. Rather, LOB data is stored in auxiliary LOB table spaces. DB2 for z/OS 10 introduced "inline" LOBs, where at least some of the LOB data can be stored in the base table space. Coincident with this, SAP increased its use of LOBs in products over the last several years. The purpose of this paper is to describe experiences converting from outline to inline LOBs and especially to recount inline LOB space efficiencies. Inline LOBs were introduced for DB2 10, but the topic of converting to inline LOBs is also pertinent to DB2 11.

Further information about running SAP solutions with IBM DB2 10 for z/OS on IBM zEnterprise System is discussed in documents such as "Running SAP Solutions with IBM DB2 10 for z/OS on the IBM zEnterprise System".

2.0 DB2 Inline vs Outline LOB Space Management

Here is a short overview of how DB2 for z/OS structures allocates space for outline verses inline LOBs.

Outline LOBs

With outline LOBs there is a base table and an additional auxiliary table for each defined outline LOB descriptor column. Each base table record would have a pointer to its associated outline LOB data. Figure 1 below is an illustration of a DB2 base table with one LOB descriptor column and pointers to data pages in an associated outline LOB auxiliary table. For simplicity only one of the pointers is drawn.

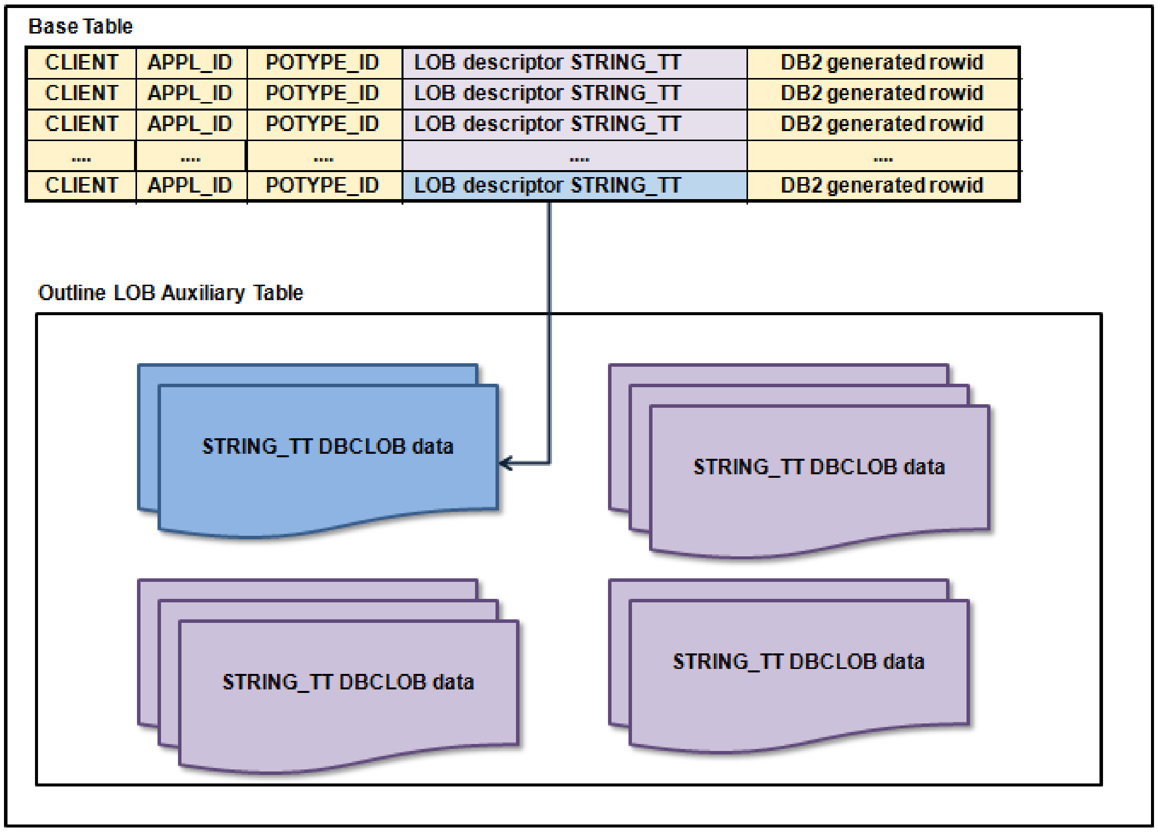


Figure 1: Example of an outline LOB

Outline LOB space efficiency can be affected by page size choice. Each outline LOB has the exclusive use of whole LOB data pages, possibly spanning multiple data pages, and does not use hardware data compression. For example, a 9,000 DBCLOB (double-byte character LOB) is 18,000 bytes in length, and needs five 4K data pages to contain it. This leaves about 2K bytes of unused space per LOB. This same LOB would require three 8K data pages with a remainder of 6K bytes of unused space per LOB. Finally a 16K data page LOB size, with the same 9,000 character double-byte outline LOB, would span two 16K data pages and have 14K bytes of unused space per LOB.

Figure 2 below is an example of how space inefficient outline LOB records are when the average LOB data record does not match up well with the defined outline auxiliary table space data page size.

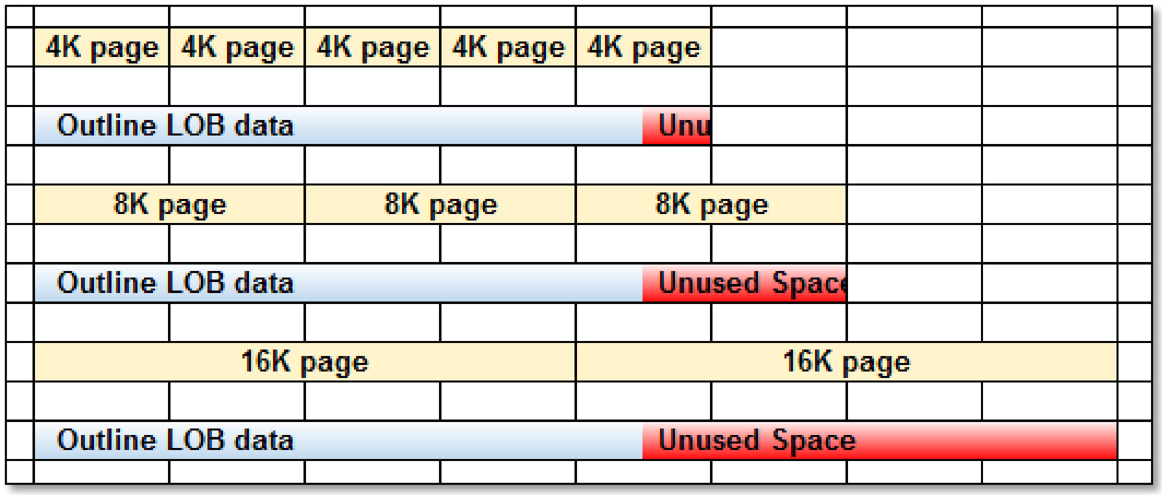


Figure 2: Space usage for a 9K DBCLOB auxiliary table space by page size

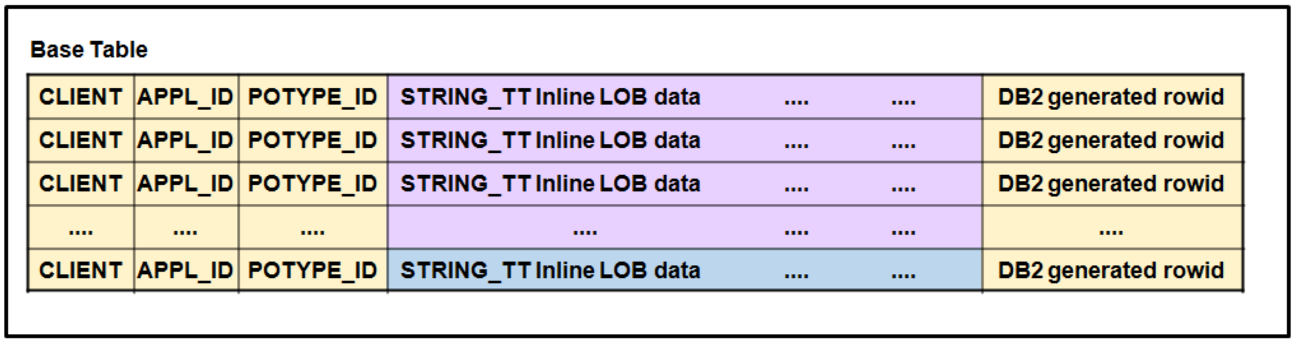
Inline LOBs

DB2 10 allows multiple inline LOB records per data page and exploits hardware data compression. This can result in significant DASD and memory space savings compared to outline LOBs that are stored uncompressed in whole exclusive use data pages.

Additional base table space may be required for an inline LOB but with efficient compression it would be significantly less storage than uncompressed outline LOB records. There is a maximum limit of 32,680 single-byte characters per inline LOB definition, when a 32K data page size is used for a table space. If the defined limit is exceeded, DB2 will overflow into an outline LOB auxiliary table as shown in the figure 1 outline LOB example.

The data page size must be large enough to contain the total length of all columns including the inline LOB column(s). For example to inline a 9000 DBCLOB column a 32K data page size must be used provided the total length of all other columns fits within the remaining 14K bytes.

Figure 3 illustrates LOB data stored inline to base table records.



Base Table

CLIENT	APPL_ID	POTYPE_ID	STRING_TT Inline LOB data	....	....	DB2 generated rowid
CLIENT	APPL_ID	POTYPE_ID	STRING_TT Inline LOB data	....	....	DB2 generated rowid
CLIENT	APPL_ID	POTYPE_ID	STRING_TT Inline LOB data	....	....	DB2 generated rowid
....	....	....	....	....	....	....
CLIENT	APPL_ID	POTYPE_ID	STRING_TT Inline LOB data	....	....	DB2 generated rowid

Figure 3: Example of an inline LOB

Further information on when to use inline LOBs verses outline LOBs can be found in the "DB2 10 for z/OS Performance Topics" guide.

3.0 Workload Description

To quantify the effects of different LOB characteristics, several measurements of SAP's Insurance application were executed. All of these measurements involved updating 500,000 FS-PM [1] Life Annuity policies for a one month processing period. SAP's FS-PM "Update Policies/Contracts" function was used to execute parallel background processes. This was accomplished by 100 parallel batch jobs with 200 policy number intervals.

This particular SAP application uses LOBs to store text related to SAP document line items. These line items are related to SAP internal application data and business logic. LOBs were used because there are often a large number of line items. To reduce their size, SAP uses application logic to "compress" the line items. When LOBs are converted to be inline, data is compressed significantly further using DB2's hardware data compression.

Three outline LOB cases and one inline LOB case were run. Results of all runs were then compared. The average LOB size for these tests was 18,000 bytes.

The three outline LOB measurement points were:

- Base table with a 4K page size and outline LOB auxiliary table space with 4K page size.
- Base table with a 4K page size and outline LOB auxiliary table space with 8K page size.
- Base table with a 4K page size and outline LOB auxiliary table space with 16K page size.

The inline LOB measurement point was:

- Base table with a 32K page size and outline LOB auxiliary table space with 4K page size.

[1] FS-PM - Financial Services - Policy Management

3.1 Inline LOB Conversion Implemenation

With the DB2 10 introduction of inline LOBs, SAP rolled out the DB2 Conversion Tool. This tool can be used to analyze a number of storage saving features added to DB2, and help the user implement conversion findings within an SAP system. Included in the tool is the ability to convert an outline LOB to an inline LOB column. This would be the primary method to convert to inline LOBs under SAP. Further information on the SAP DB2 Conversion Tool can be found in the "User Guide for SAP DB2 Conversion Tool" manual.

As an alternative, and on non-SAP systems, DB2 utilities and commands are used for inline LOB conversions.

An outline LOB auxiliary table space, exercised by the FS-PM Life Annuity policy update process, was altered to use various page sizes before being converted to an inline LOB column, for a final measurement point. DB2 utilities and commands, UNLOAD, TEMPLATE, ALTER, REORG and LOAD were used in this effort and documented in Appendix section "8.1 Inline LOB Conversion using DB2 Utilities".

4.0 Test Environment

All tests were done using a physical 3-tier environment where each of the three layers resided on separate machines. The SAP Database Server was on an IBM zEnterprise system (zEC12) running z/OS. The SAP application server functions were run on an IBM Power 780 server running one AIX LPAR with nine SAP instances – one for the Central Instance and eight Dialog Instances.

Figure 4 illustrates the physical 3-tier hardware landscape used by all documented test measurements.

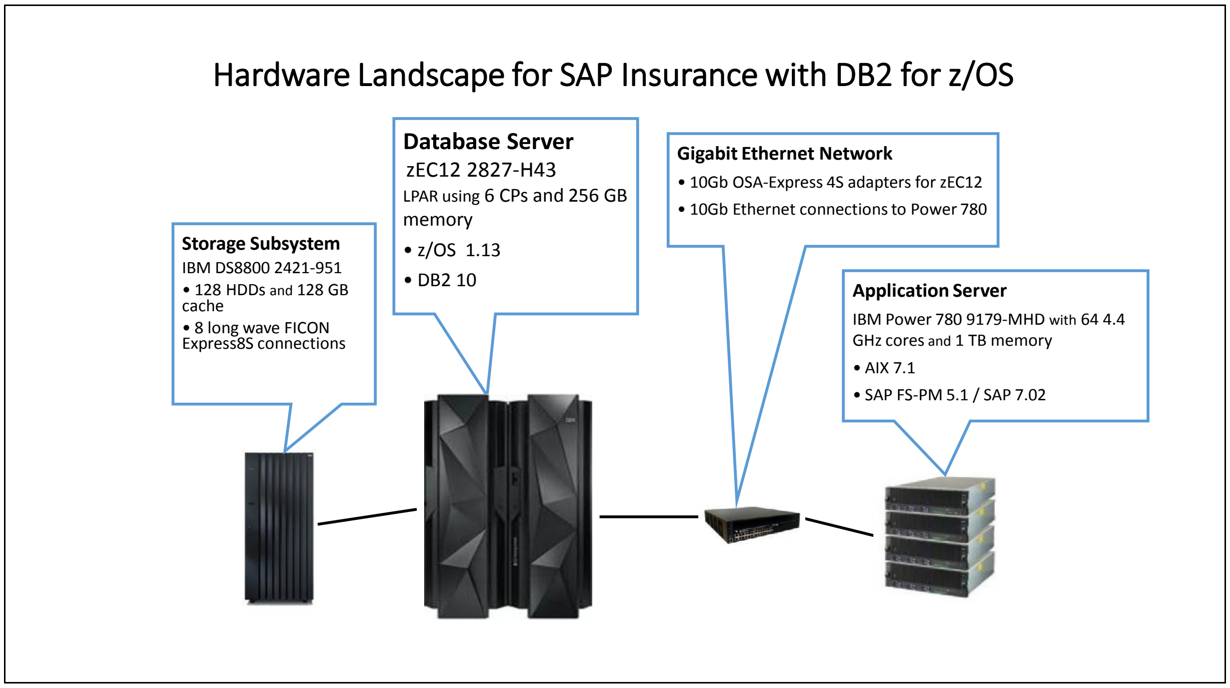

Hardware Landscape for SAP Insurance with DB2 for z/OS
Database Server
zEC12 2827-H43
LPAR using 6 CPs and 256 GB memory
• z/OS 1.13
• DB2 10
Gigabit Ethernet Network
• 10Gb OSA-Express 4S adapters for zEC12
• 10Gb Ethernet connections to Power 780
Storage Subsystem
IBM DS8800 2421-951
• 128 HDDs and 128 GB cache
• 8 long wave FICON Express8S connections
Application Server
IBM Power 780 9179-MHD with 64 4.4 GHz cores and 1 TB memory
• AIX 7.1
• SAP FS-PM 5.1 / SAP 7.02

Figure 4: Test landscape installed at IBM Poughkeepsie, NY, USA

4.1 Hardware

System z Database Server

IBM zEnterprise Class 2827-H43 (zEC12) with 6 dedicated CPs configured online was used for all the measurements. This system had 256 GB of real storage configured on one z/OS LPAR. DB2 10 exploited 1 MB pages.

Database DASD

IBM Storage Subsystem DS8800 2421-951 with 128 15K rpm HDDs and 128 GB cache connected via eight long wave FICON Express8S connections. HyperPAV and High Performance FICON for System z (zHPF) with multi-track support were used to reduce disk I/O queuing and improve the efficiency of I/O resources.

Application Servers

IBM Power 780 9179-MHD with (64) SMT4 4.4 GHz cores, 1024 GB of RAM, and 16 MB pages running in one AIX LPAR.

Network

The application servers were connected via 10 Gb Ethernet adapters through a 10 Gb Ethernet switch to the zEC12 via two OSA-Express4S adapters. The Optimized Latency Mode (OLM) option of the OSA-Express4S adapters was used to improve the elapsed time of this communication.

4.2 Software

SAP software stack

- NetWeaver 7.02
- ECC 6.0 Ehp5
- FS-PM 5.1
- MSGPMCON 5.1 SP6 with CAIMAN 3480.44 and TomatosX R1

IBM software stack

- z/OS 1.13
- DB2 for z/OS 10
- IBM Data Server for CLI that is shipped as part of DB2 Connect 9.7 FP6
- AIX 7.1.0

5.0 Measurement Results and Analysis

5.1 Measurement Results

All these measurements took approximately 12 minutes of elapsed time, driving a six engine z/EC12 database server in the low 60% busy range and a sixty-four core Power 780 model 9179-MHD client about 65% busy.

Table 1 below contains summarized results from these measurements. Data was collected from RMF, DB2PM accounting and runstats data.

	4KB outline LOBs	**8KB outline LOBs**	**16KB outline LOBs**	**Compressed inline LOBs**
Runid	S30712I1	S30711I1	S30713I1	S30730I1
Base Table pagesize (KB)	4	4	4	32
LOB pagesize (KB)	4	8	16	4
Elapsed time (sec)	778	746	732	731
DB2 Class 1 CPU time per commit (sec)	0.0477	0.0475	0.0453	0.0485
DB2 Class 2 CPU time per commit (sec)	0.0324	0.0322	0.0299	0.0327
# of commits	42,500	42,500	42,500	42,500
Total DB2 Class 1 CPU time (sec)	2,027	2,019	1,926	2,061
Total DB2 Class 2 CPU time (sec)	1,376	1,367	1,271	1,392
Used Base table npage	6,995	7,268	7,092	99,751
Used Base Index nleaf	14,097	13,978	13,910	13,886
Used LOB nactive	2,514,240	1,503,450	1,000,890	21
Used LOB Index nleaf	4,769	4,686	4,789	0
Used Base table + Index size (MB)	82	83	82	3,171
Used LOB table + Index size (MB)	9,840	11,764	15,658	0
Used Storage Space (Base + LOB in MB)	9,922	11,847	15,740	3,172

Table 1: LOB measurement results

5.2 Analysis

The measurement runs focused on LOB space usage and the execution elapsed time of the SAP FS-PM "Update Policies/Contracts" function. Three measurements used DB2 data page sizes of 4KB, 8KB and 16KB on an outline LOB auxiliary table space were compared. This set was then compared to one measurement that used a 32K data page size on a base table space with the same LOB data in an inline column.

Figure 5 shows the elapsed time and database (DB) CPU time of the four measurements. Using the 4KB outline LOB as a base point ratio of 1, the elapsed time comparisons were very close. However, there was a significant difference in space usage. Outline LOB space requirements can be minimized by knowing the sizes and distribution[2] of the LOBs and choosing a data page size close to the mean size. This technique can be efficient when the standard deviation of the LOB size is very small. The average size of the LOB data, per DB2 record in these tests, was 18,000 bytes or 9,000 double-bytes.

Of the outline LOB measurements, the most space efficient used a 4KB data page size for the outline LOB auxiliary table space. This measurement also had the longest elapsed time. If a 16KB data page size was used, elapsed time was improved over the 4KB measurement, however with significant storage space use. For outline LOB measurements an 8K data page size was a good compromise for managing elapsed time and space usage.

The most significant space saving choice was inline LOBs. This is because inline character LOBs allow DB2 to exploit System z hardware data compression, compared to outline LOBs that don't. Also, the base table spaces are more efficiently used. A significant factor is the use of hardware data compression. Using DB2 utilities, a 72% reduction was observed in the base table space due strictly to hardware data compression.

Of course this advantage of inline LOBs depends on the compressibility of the data. For example, if the LOBs consist of pre-compressed data such as PDFs or commonly used video formats, they will not see as much benefit from DB2 data compression.

[2] See Appendix Section "8.2 LOB Distribution for a DBCLOB" for determining LOB distributions.

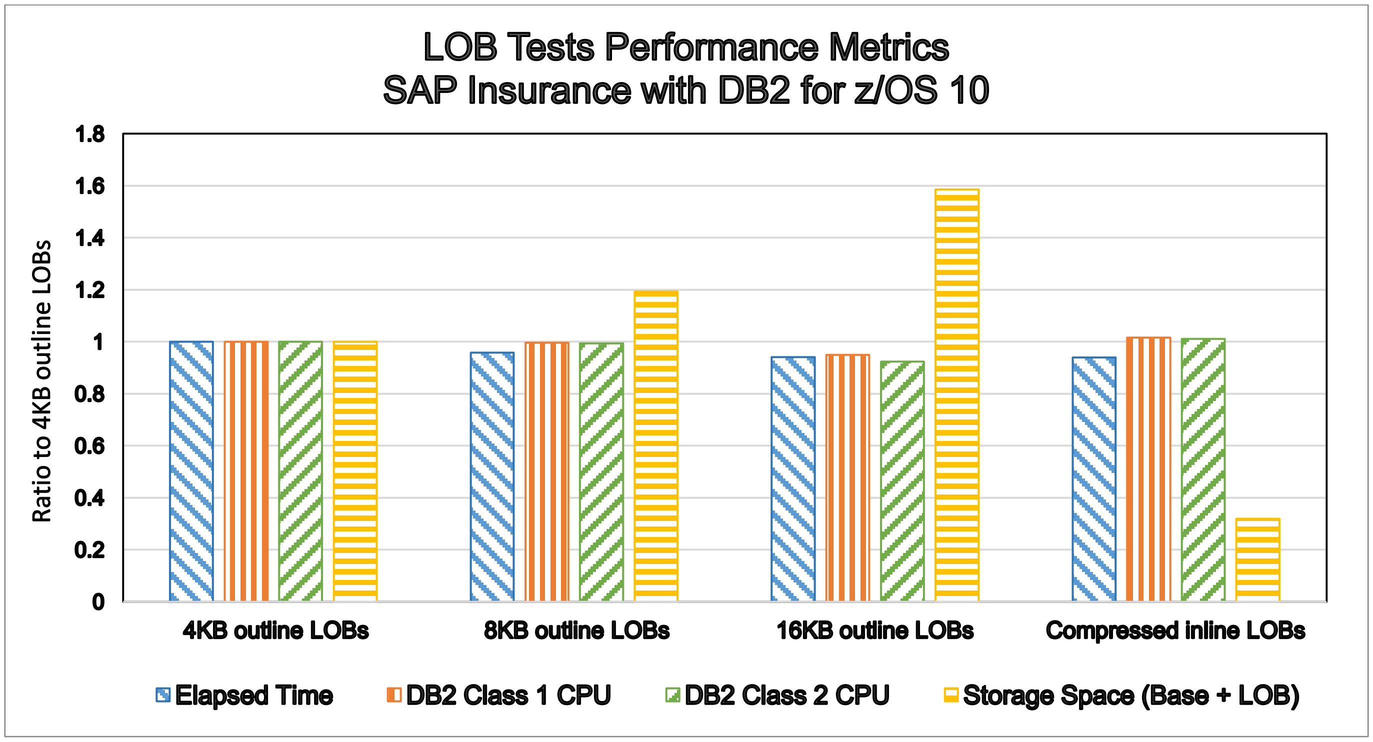


Figure 5: LOB Tests Performance Metrics

6.0 Conclusions

Choosing an appropriate data page size for the outline LOB auxiliary table space can improve DASD and memory space usage. Exploiting the DB2 inline LOB feature, a further 68% to 80% DASD space savings was realized with elapsed time reduction and minimal CPU time impact.

Key Points:

- An understanding of the outline LOB data length distribution is important. Selecting a data page size that best fits the majority of outline LOB records can have significant DASD space savings.
- LOB data length distribution formulas are found in, "DB2 10 for z/OS Performance Topics" with a further adjustment for DBCLOBs in appendix section "8.2 LOB Distribution for a DBCLOB".
- For DB2 10 and DB2 11, a conversion to inline LOBs should be considered especially if DASD space use is of concern. Huge DASD space savings can be found, as was seen by measurements summarized in this document.
- For inline LOB conversions the SAP DB2 Conversion Tool is documented in the "User Guide for SAP DB2 Conversion Tool". For non-SAP inline LOB conversions refer to appendix "8.1 Inline LOB Conversion using DB2 Utilities"

7.0 References

1. *IBM Corp 2012. Running SAP Solutions with IBM DB2 10 for z/OS on the IBM zEnterprise System, Document Number SG24-7978*
   http://www.redbooks.ibm.com/abstracts/sg247978.html?Open
2. *SAP Corp. User Guide for SAP DB2 Conversion Tool*
   www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/1011d4a7-fa47-2f10-acad-afbd5f6638fb?QuickLink=index&overridelayout=true&53824530611167
3. *IBM Corp. 2013. DB2 10 for z/OS Utility Guide and Reference*
   http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=%2Fcom.ibm.db2z10.doc.ugref%2Fsrc%2Fugref%2Fdb2z_ugref.htm
4. *IBM Corp. 2014. DB2 10 for z/OS Command Reference*
   http://publib.boulder.ibm.com/epubs/pdf/dsncrm08.pdf
5. *IBM Corp 2006. LOBs with DB2 for z/OS: Stronger and Faster.*
   http://www.redbooks.ibm.com/abstracts/sg247270.html
6. *IBM Corp 2011 DB2 10 for z/OS Performance Topics.*
   http://www.redbooks.ibm.com/redbooks/pdfs/sg247942.pdf
7. *IBM Corp. 2011 SAP for Insurance Tests.*
   http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101890
8. *IBM Corp. 2013. IBM United States Software Announcement 213-376, IBM DB2 11 for z/OS: The database for data and analytics*
   http://www-01.ibm.com/common/ssi/rep_ca/6/897/ENUS213-376/ENUS213-376.PDF
9. *SAP Best-Practices Document: Migrating SAP Systems to DB2 11 for z/OS*
   https://websmp108.sap-ag.de/~sapidb/011000358700001057822013E (requires SAP user id)
10. *SAP DBA Guide: DB2 11 for z/OS), Database Administration Guide: SAP on IBM DB2 for z/OS*
    https://websmp108.sap-ag.de/~sapidb/011000358700001053572013E (requires SAP user id)
11. *IBM Corp. 2013. Large Systems Performance Reference, Document Number SC28-1187-17*
    https://www-304.ibm.com/servers/resourcelink/lib03060.nsf/pages/lsprindex/$file/SC28118717.pdf
12. *IBM Corp. 2011. DB2 10 for z/OS with SAP on IBM System z Performance Report*
    http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101845
13. *IBM Corp. 2013. DB2 11 for z/OS Technical Overview*
    http://www.redbooks.ibm.com/Redbooks.nsf/RedpieceAbstracts/sg248180.html?Open
14. *SAP AG 2012. SAP for Insurance on IBM System z Reference Architecture*
    http://scn.sap.com/docs/DOC-14381

8.0 Appendix

8.1 Inline LOB Conversion using DB2 Utilities and Commands

When converting to inline LOBs for a non-SAP workload, DB2 utilities and commands are used. This section highlights a few key points implementing an inline LOB conversion using DB2 utilities and commands UNLOAD, TEMPLATE, ALTER, REORG and LOAD. For complete DB2 10 utility and command options refer to manuals "DB2 10 for z/OS Utility Guide and Reference" and "DB2 10 for z/OS Command Reference ".

- **UNLOAD**–The UNLOAD utility is used to unload data from a DB2 table space and associated LOB table spaces. When using the UNLOAD utility, option SPANNED YES, allows the SYSREC data output, or unloaded data, to span multiple volumes, and improves performance of read-write operations. The SPANNED YES option requires the SYSREC output to be defined as RECFM VBS (Variable Block Spanned), and all DB2 table column entries to be provided in the UNLOAD statement, with the LOB columns at the end. If the table space record length is more than 32 KB, this method of UNLOAD is needed.

```
UNLOAD TABLESPACE A070XTGP.ABDAPOX SPANNED YES
FROM TABLE "SAPR3"."/PM0/ABDAPOXML"
(
CLIENT VARGRAPHIC(3),
APPL_ID VARGRAPHIC(22),
POTYPE_ID VARGRAPHIC(10),
DB2_GENERATED_ROWID_FOR_LOBS ROWID,
STRING_TT DBCLOB
)
SHRLEVEL CHANGE;
```

- **TEMPLATE**–UNLOAD with the TEMPLATE control statement is an alternative way of unloading a DB2 table space and associated LOB data. The SYSREC output data, represented by the INSUNLD template name below, defaults to a RECFM=VB (Variable Block), which does not allow the unloaded data to span multiple volsers. Using the NOPAD option indicates that variable length fields will be unloaded without padding, giving a better chance of fitting on a volume.

```
LISTDEF INSDATA
 INCLUDE TABLESPACE A070XTGP.ABDAPOX
  TEMPLATE INSPUN
        DSN(SAPDB2.IAPDB2.&DB..&TS..LOAD0.PUN)
        STORCLAS (DB2UTIL)
        UNIT 3390 SPACE (500,100) CYL
        DISP (NEW,CATLG,CATLG)
  TEMPLATE INSUNLD
        DSN(SAPDB2.IAPDB2.&DB..&TS..P&PART.)
        STORCLAS (DB2UTIL)
        UNIT 3390 VOLCNT 10 SPACE (2500,1000) CYL
        DISP (NEW,CATLG,CATLG)
   UNLOAD LIST INSDATA
        PUNCHDDN(INSPUN)
        UNLDDN(INSUNLD)
        NOPAD
```

- **LOAD**–The LOAD DB2 utility is used to load data back into a table space and associated LOB table space or inline LOB. For DB2 10 the SYSREC and punch output datasets, from the UNLOAD or TEMPLATE UNLOAD processes, are all that are needed. Using parameter LOAD REPLACE will avoid duplicate record errors.

- **ALTER**–The DB2 ALTER command is used in the conversion to inline LOBs. Note that a LENGTH 9000 on a DBCLOB (Double Byte Character LOB) means 18,000 characters reside in the containing table's base table space rather than in a column's auxiliary table space. The maximum inline LOB column length is 16380 double byte characters or 32760 single byte characters.

```
ALTER TABLE "SAPR3"."/PM0/ABDAPOXML"
  ALTER COLUMN STRING_TT SET INLINE LENGTH 9000;
```

- **REORG**–To REORG a table space with an auxiliary LOB table space, the AUX YES parameter is used. This option causes LOB table spaces to be reorganized along with the associated base table space, in a single utility call. With an inline LOB you can also use the PREFORMAT option, which is not allowed against outline LOB auxiliary table spaces.

8.2 LOB Distribution for a DBCLOB

To find the distribution of DBCLOB data by length in a LOB table, the SQL below was used for measurements in this document. The difference between this sample SQL and SQL found in, "DB2 10 for z/OS Performance Topics" is in the calculation for single byte CLOBs (Character LOBs) and BLOBs (Binary LOBs) verses DBCLOB (Double-Byte Character LOBs). For DBCLOB data the LOB string length is divided by 2000 instead of 4000 to account for double bytes being twice the length of single bytes.

```
WITH LOB_DIST_TABLE (LOB_LENGTH, LOB_COUNT) AS
(
SELECT LOBCOL_LENGTH, COUNT(*)
FROM
(
SELECT ((LENGTH(STRING_TT) / 2000) + 1) * 4000
AS LOBCOL_LENGTH
FROM SAPR3."/PM0/ABDAPOXML"
)  LOB_COL_LENGTH_TABLE
GROUP BY LOBCOL_LENGTH
)
SELECT '04000', SUM(LOB_COUNT)
FROM LOB_DIST_TABLE
WHERE LOB_LENGTH <= 4000
UNION
SELECT '08000', SUM(LOB_COUNT)
FROM LOB_DIST_TABLE
WHERE LOB_LENGTH <= 8000
UNION
SELECT '12000', SUM(LOB_COUNT)
FROM LOB_DIST_TABLE
WHERE LOB_LENGTH <= 12000
UNION
SELECT '16000', SUM(LOB_COUNT)
FROM LOB_DIST_TABLE
WHERE LOB_LENGTH <= 16000
UNION
SELECT '20000', SUM(LOB_COUNT)
FROM LOB_DIST_TABLE
WHERE LOB_LENGTH <= 20000
UNION
SELECT '24000', SUM(LOB_COUNT)
FROM LOB_DIST_TABLE
WHERE LOB_LENGTH <= 24000
UNION
SELECT '28000', SUM(LOB_COUNT)
FROM LOB_DIST_TABLE
WHERE LOB_LENGTH <= 28000
UNION
SELECT '32000', SUM(LOB_COUNT)
FROM LOB_DIST_TABLE
WHERE LOB_LENGTH <= 32000
UNION
SELECT '99999', SUM(LOB_COUNT)
FROM LOB_DIST_TABLE
WHERE LOB_LENGTH <= 99999 ;
```