Suppress Depression with Compression

James Magill

CenCan DB2 Users Group
April 30, 2019
1:30 – 2:30

Agenda

Why Compression
Background
Data Compression
Fixed length compression
Variable length compression
DB2 Implementation
New Page Formats
Expectations

This presentation is meant to level set everyone who may work on this feature, and to answer any questions you may have. So, please, ask if something is not clear! I am also counting on those who have been involved in research to keep me honest. Please remember that all this information is IBM proprietary and confidential. As far as I know, IBM has made none of it public, so none of it can be shared yet outside BMC.

WHY COMPRESSION

- **Everyday Life – pictures , zipping files, etc…**
- **Reduce amount of data read or written to disk**
- **Reduces IO costs**
- **Reduce storage needed**
  - Table size
  - Index size
  - Backup image size

Background

Past Compression in DB2

Past Compression in DB2

Tablespace compression was introduced in V3 (early 90's)

- Dictionary based, one row at a time
- Uses hardware instructions
- Typically good compression rate (~80%) with low CPU overhead

Index compression in V9

- Page level software compression of leaf pages
- 8K to 32K compressed to 4K on disk
- Compression occurs during I/O
- Not very popular because of the CPU overhead

Most customer tablespaces are compressed. It can actually speed up some applications by increasing the hit ratio in the buffer pool. The overhead truly is negligible.

Past Compression in DB2

"Auto compression" added in DB2 10

- Removed need for LOAD or REORG to compress data

LOB compression in V12

- Hardware compression using z/EDC, done during I/O
- Typically, 50% rate for CLOB's, with low CPU overhead
- "Dictionaryless"

We don't encounter many customers compressing indexes. The overhead is perceived as too high for the savings.

LOB compression is still very new. We don't know yet how widespread its use will be.

DATA COMPRESSION

Data Compression – Factors to Consider

- Data Row Size – shorter row has diminishing return
- Table Space Size – more rows the better
- Processing Costs – compression costs higher than decompression
- I/O Costs
- Data Patterns
- DSN1COMP – Compression Ratio < 10-20%

Data Compression – When It Happens

- **Compression/Expansion is at row level**
- **Based on observed data**
  - Reorg
    - End of Unload phase dictionary is build
  - Load/Replace
    - Looks at initial rows to build dictionary
  - Insert
    - Generally > 10 K rows
- **Sequences observed more frequently get an entry in the dictionary**

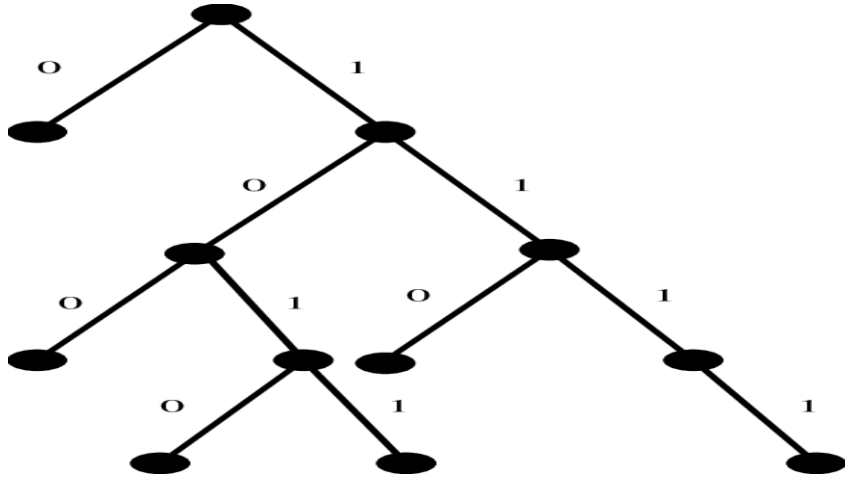
Data Compression – Techniques

- Fixed Length (Lempel-Ziv-Welch)
- Variable Length (Huffman)

Data Compression – Lempel-Ziv

- **Fixed Length**
  - IBM initial technique
  - Always 12 bit index
  - Hardware instruction support
    - CMPSC
    - Faster and less costly
  - Software support
    - Slower and more costly

Ziv Tree Example



Data Compression – Ziv Tree

- **Created when creating a dictionary**
- **Grows with amount of trained data**
- **Path from Root to Leaf represent repeated bytes**
- **Closed and formatted for use with hardware**
- **Most common size is 4096 nodes**
  - 12 bit fixed length symbols
- **Huffman uses Ziv Tree**
  - Symbol Translation Table
  - Maps fixed length Ziv tree node into variable length
  - 1-16 bit length symbols
  -

Huffman Technique

Optional extension to existing tablespace compression

- Maps 12 bit fixed dictionary entries to variable length Huffman codes
- More frequently found values are shorter

Hardware instruction supports STT (new mapping table)

Potential for significantly increased compression rate

We don't have figures yet for how much better typical compression rates will be, nor for how much the CPU overhead will be.

System Requirements

This is a z14 exploitation feature

Hardware instruction with STT is available only on z14

Data will not be compressed on z13 or earlier

Software decompression is available on z13 (for data sharing coexistence)

IBM is recommending customers upgrade to z14 across their environments

IBM is continuing to add software features to push sales of the z14. They didn't give relative performance figures, but it's safe to bet that software decompression is painfully slow, so their model is to couple exploitation of this feature to deploying z14's across customer environments.
The STT is the table linking Huffman codes to the corresponding dictionary entries.

DB2 Implementation

Function Level 504

System Requirements Met

Create Database/Tablespace with Compression On as usual

Must be a UTS

KEEPDICTIONARY Not Used

DB2 Implementation

New ZPARM TS_COMPRESSION_TYPE

- FIXED_LENGTH (default) or HUFFMAN
- In data sharing, all members must have same value

No plans for external specification of fixed or Huffman

- DSN1PRNT is the tool they suggest
- DSN1COMP does NOT support Huffman

DB2 Implementation

- Expansion Preprocessor Routine or EPROC
- Executable Code generated when dictionary is closed
- Included in the LZDH
- Software expansion using Huffman dictionary when Huffman is not available eg..migration
- Certain GPR's usage to call EPROC to expand a compressed record

DB2 Implementation

- **Entropy Descriptor or ED**
- Histogram of the count of Huffman nodes at each depth in the Huffman tree
- The Huffman tree is limited (by the CMPSC instruction) to depth 16, the ED is an array of 16 halfword values
- DSN1PRINT will format the ED under **LZDH ED STATS:**

Possible DDL Support

IBM future syntax to CREATE or ALTER speculated

- COMPRESS YES FIXEDLENGTH
- COMPRESS YES HUFFMAN

Would allow a customer to set a default with the ZPARM

Supported with auto-compression

Interesting Catalog Fields

- **SYSTABLESPACE**
  - COMPRESSRATIO

  Average percentage of bytes saved by compression on each compressed data record in the table space when the table space is defined with the COMPRESS YES attribute.
- **SYSTABLEPART**
  - COMPRESS

  For a table space partition, whether the COMPRESS attribute for the partition is YES.

  For a nonpartitioned table space, whether the COMPRESS attribute is YES for the table space.

Interesting Catalog Fields

- COMPRESSRATIO

Average percentage of bytes saved by compression on each compressed data record in the table space when the table space is defined with the COMPRESS YES attribute.

For example, a value of 25 indicates that the average compressed record size is approximately 75% the size of the uncompressed record.

New Page Formats

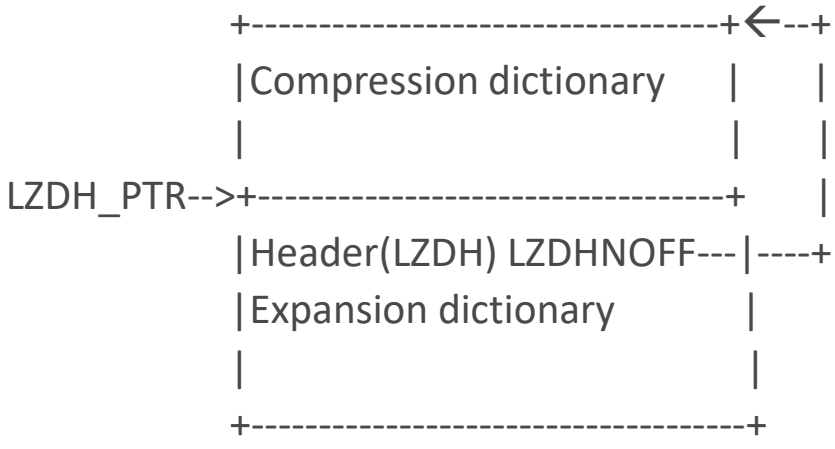
Data Compression – Header Page

- **HPGZLD**
  - ‘L’ or ‘F’ for the fixed-length dictionary
  - ‘H’ for Huffman dictionary
  - Blank or zero for “no dictionary”
- **HPGZ43NO**
  - Page number of first dictionary page.

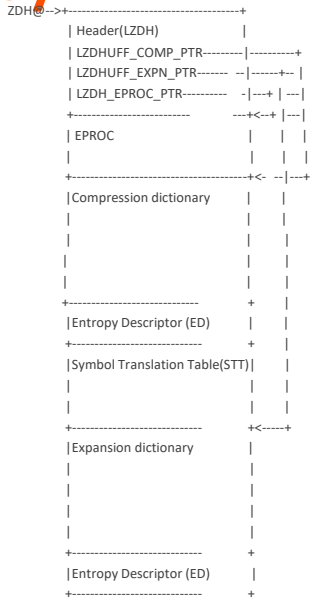
  (Never changed/used for Huffman dictionaries.)
- **HPGZ4PNO**
  - Page number of first dictionary page.
- **HPGZNUMP**
  - Number of 4K memory pages required to store the dictionary.

  This value is larger for Huffman dictionaries (compared to fixed-length) because we need to keep additional information like STT or EPROC.

Data Compression – Lempel-Ziv-Welch



Huffman Dictionary



We don’t have figures yet for how much better typical compression rates will be, nor for how much the CPU overhead will be.

DSN1PRINT

LZDH INFO LZDHIDFR='2370'X LZDHLENG=1024 LZDHVID='LZDH' LZDHTYPE='H' LZDHPART=1

LZDHFLGS='D00000'X LZDHSTOR=74752 LZDHESTG=0 LZDHNOFF=0 LZDHNODE=3872 LZDHSIZE=4096

LZDHNUMP=19 LZDHNUDP=19 LZDHEGR0='000A0100'X LZDHCGR0='000B0000'X

LZDHVERS='0000000000000000'X LZDHDBNM='TSTDBB ' LZDHTSNM='TSTTS'

LZDH HUFFMAN LZDH_HUFF_EYE='HUFF' LZDH_PRODUCER='IBM1802A' LZDHEDOFFSET='8000'X

LZDHGR1='0000000000000800'X LZDH_HUFF_COMPSTOR=40992 LZDH_HUFF_EXPNSTOR=32800

LZDH_EPROC_STOR=1024 LZDH_STT_USED=3872 LZDH_ED_TOTAL=4097

LZDH – Dictionary Header

```
0000  00000000 00000000 00000220 00000003  *.............*
0010  0FE02370 0400D3E9 C4C8C800 000001D0  *.\....LZDHH....}*
0020  00000001 2FA00000 00000000 00000000  *.............*
0030  0F2C0000 0FF00000 00130000 0014000A  *.....0.........*
0040  0100000B 00000000 00000000 0000E2C3  *..............SC*
0050  C3C9C2D4 C4C2E2C3 C3C9C2D4 E3E20000  *CIBMDBSCCIBMTS..*
0060  00000000 001502DE 72520000 00000000  *.............*
0070  00000000 00000000 00000000 00000000  *.............*
0080  00000000 00000000 0000C8E4 C6C67F80  *.........HUFF".*
0090  00000000 00000000 07F80000 9F800000  *.........8.....*
00A0  7FA00000 04000F2C 00000000 00000000  *".............*
00B0  00000000 00000000 00000000 01D20340  *............K. *

01D0  00000000 10000000 B0000000 04000000  *.............*
```

EPROC

```
0410  0000C5D7 D9C3B904 0016B908 0076B908 *..EPRC.........*
0420  0098C284 00000002 E3008000 0004C00A  *.qBd....T.....{.*
0430  0000FFFF C28A0000 0008A729 0030B920  *....B.....x.....*
0440  0089A7D4 0013B904 00B8B909 00B9EBBB  *.ixM............*
0450  0003000D A7A90030 1BA21ABA EB00B000  *....xz...s......*
```

CMPSC INSTRUCTION ENHANCEMENT

- **HARDWARE INSTRUCTION**
- **SETUP**
  - SOURCE – even/odd register pair
  - Target – even/odd register pair
  - R0 – flags for compression/expansion found in LZDH
  - R1 – pointer to the compression/expansion dictionary
    - OFFSET to STT found in LZDH header Boolean Or'd into R1 for Huffman

What to Expect

- Huffman compression to provide better compression ratio then fixed length compression
- Better compression ratio will mean performance gains by reducing number of getpages
- Similar costs
- High performance costs with S/W expansion
- Compression ignored if running with Huffman dictionary on Z13 machine. Expansion will work using EPROC

Is Huffman Better

- **Find a UTS object that is of significant size and is compressed and has a fairly high compression ratio with fixed-length compression**
- **Either fetch compression stats from last Reorg or do an new Reorg on object**
- Change the online zPARM TS_COMPRESSION_TYPE to HUFFMAN and executes the SET SYSPARM command to load it or setup object on a separate Huffman system and Reorg it
- Compare Reorg utility compression stats


Thank You
Bring IT to Life
bmc