COM data pattern generator for Emerald

By   
Don Capps   
[capps@iozone.org](mailto:capps@iozone.org),  
[Don.Capps@netapp.com](mailto:Don.Capps@netapp.com)   
Date: 2/29/2016

Within the Emerald benchmark there is a section named the COM (Capacity Optimization Method). This is a simple test that produces a binary result. Either the system has a COM, or it does not. Technologies that will qualify as a COM include:

* Thin provisioning
* Compression
* Deduplication

The following document describes the data layout that is used to test the system and determine if it has a measurable technology that can achieve Compression, or Deduplication. The Thin provisioning mechanism is easily validated and does not require more complex testing methodoligies.

The Emerald COM (Capacity Optimization Method) pattern generator is described below.

Usage: ./comgen\_15 $Revision: 1.5 $

-d <dir>

[-s] <salt>]

[-n number of files] Defaults to 4.

[-b blocksize] (in KiB) Enables alternative pattern generation.

[-C] Selective alternative pattern generation for Compression no dedupe files.

[-D] Selective alternative pattern generation for Dedupe no compression files.

[-B] Selective alternative pattern generation for Dedupe and compression files.

[-I] Selective alternative pattern generation for irreducible files.

[-f filesize] (in GiB) Enables alternative pattern generation.

[-r devicename] Use this raw block device.

[-O] Use O\_DIRECT. ( If it works on your box )

The –d <dir> permits one to specify the directory where the file, or files, will be created.

The –s <salt> permits one to set the initial seed of the random number generator.

The –n number, permits one to specify the number of files to create of each type.

The –b blocksize permits one to specify the block size in KiB for the writes.

The –C permits one to specify to generate the pattern that is compressible and NOT dedupable.

The –D permits one to specify to generate the pattern that is dedupable and NOT compressible.

The –B permits one to specify to generate the pattern that is compressible AND dedupable.

The –I permits on to specify to generate the pattern that is irreducible by any COM.

The –f filesize permits one to specify the file size in GiB.

The –r devicename permits one to specify the name of a raw block device.

Example for Unix: -r /dev/sda

Example for Windows: -r [\\.\PHYSICALDRIVE1](file:///\\.\PHYSICALDRIVE1)

The –O enables the use of O\_DIRECT I/O, andsimiliar cache bypass under Windows.

# Compressible and non-dedupable

The “Compressible and NON-dedupable” data patterns are created using buffers that are filled with a random number, that is updated every 256 bytes. Also, the random number is incremented for every five 32 bit patterns within the buffer. This results on 256 bytes where 4 byte strings repeat 5 times for 20 bytes, followed by a new random 32 bit value which again is 4 byte strings that repeat 5 times for 20 bytes.   
The buffers are not reused for successive blocks as the pattern is written. This creates a data set that is highly compressible, but non-dedupable.

The use of the 256 byte count to alter the random number used is to prevent dedupe from triggering all the way down to 256 byte granule sizes. Below this size, the overhead of the dedupe meta-data would be excessive, and therefor very unlikely to engage.

Example: Hex dump

0000000 **7a62 9c9b 7a62 9c9b 7a62 9c9b 7a62 9c9b**

0000020 **7a62 9c9b** **7a62 9d9b 7a62 9d9b 7a62 9d9b**

0000040 **7a62 9d9b 7a62 9d9b** **7a62 9e9b 7a62 9e9b**

0000060 **7a62 9e9b 7a62 9e9b 7a62 9e9b** 7a62 9f9b

0000100 7a62 9f9b 7a62 9f9b 7a62 9f9b 7a62 9f9b

0000120 7a62 a09b 7a62 a09b 7a62 a09b 7a62 a09b

0000140 7a62 a09b 7a62 a19b 7a62 a19b 7a62 a19b

0000160 7a62 a19b 7a62 a19b 7a62 a29b 7a62 a29b

0000200 7a62 a29b 7a62 a29b 7a62 a29b 7a62 a39b

0000220 7a62 a39b 7a62 a39b 7a62 a39b 7a62 a39b

0000240 7a62 a49b 7a62 a49b 7a62 a49b 7a62 a49b

0000260 7a62 a49b 7a62 a59b 7a62 a59b 7a62 a59b

0000300 7a62 a59b 7a62 a59b 7a62 a69b 7a62 a69b

0000320 7a62 a69b 7a62 a69b 7a62 a69b 7a62 a79b

0000340 7a62 a79b 7a62 a79b 7a62 a79b 7a62 a79b

0000360 7a62 a89b 7a62 a89b 7a62 a89b 7a62 a89b

**0000400** 7e5f 4651 7e5f 4751 7e5f 4751 7e5f 4751 << 256 byte boundary

0000420 7e5f 4751 7e5f 4751 7e5f 4851 7e5f 4851

Note: The patten has 4 byte strings that repeat 5 times for 20 bytes, and changes to a new random number at 256 bytes. (Octal 400 on the left) The sequence continues for the file size requested.

# Dedupable and NON compressible

The “Dedupable and NON-compressible” data patterns are created using a buffer that is filled with a series of random numbers. This buffer is reused to fill the file or device. This creates patterns that are non-compressible, however very dedupable. I used a 4 KiB block size for this example.

0000000 793c 3a43 7006 23cc ac46 42ce 49ee 0e90

0000020 afba 22e5 ee3c 1573 b024 6e1d 43f6 5f17

0000040 fc3f 706e c5c4 0672 ca2a 5915 b917 258b

0000060 a842 7824 c2ad 2ea2 03c6 3f9b dcc9 5a14

0000100 3e53 0fab c32e 3388 1b70 57c6 8315 0f33

0000120 df7e 7ee2 0a46 60bf ab49 1e3b 5520 5f6a

0000140 d8d0 3dfa 5f7a 1fb0 52d6 744c 9b40 3ed2

0000160 a6f8 74ce 1fb1 2b34 b39f 6b0c c2a1 0ee4

0000200 dfaa 4ca5 3936 1dff 179c 5f05 3380 4751

0000220 3d14 240c fe89 1f7d e486 08d1 1df0 0bf2

0000240 7db0 448b d0f8 21e8 5b2c 3bef c471 5f4a

0000260 0797 25a7 5dd0 78e3 40b1 1724 3055 4d13

0000300 43fb 26ca 27a2 2935 11d0 60bd a04e 1cdc

0000320 6faf 58a0 779e 0d04 2cbf 2049 c028 2418

0000340 949b 54ef 77ad 3d05 1e27 31f4 ac10 11e7

0000360 579f 00d1 8824 2fcf 0c01 61f3 47e4 1b99

0000400 d7e3 6c3e af9a 11cf b098 5a11 f95a 3f60

0000420 a437 7792 5435 3c5b 8288 0ff5 b6e7 3f4c

….. continuing to the 4 KiB byte boundary (10000 Octal)

where the pattern will repeat. (dedupe trigger)

0010000 793c 3a43 7006 23cc ac46 42ce 49ee 0e90

0010020 afba 22e5 ee3c 1573 b024 6e1d 43f6 5f17

0010040 fc3f 706e c5c4 0672 ca2a 5915 b917 258b

0010060 a842 7824 c2ad 2ea2 03c6 3f9b dcc9 5a14

0010100 3e53 0fab c32e 3388 1b70 57c6 8315 0f33

0010120 df7e 7ee2 0a46 60bf ab49 1e3b 5520 5f6a

0010140 d8d0 3dfa 5f7a 1fb0 52d6 744c 9b40 3ed2

0010160 a6f8 74ce 1fb1 2b34 b39f 6b0c c2a1 0ee4

0010200 dfaa 4ca5 3936 1dff 179c 5f05 3380 4751

0010220 3d14 240c fe89 1f7d e486 08d1 1df0 0bf2

0010240 7db0 448b d0f8 21e8 5b2c 3bef c471 5f4a

0010260 0797 25a7 5dd0 78e3 40b1 1724 3055 4d13

0010300 43fb 26ca 27a2 2935 11d0 60bd a04e 1cdc

0010320 6faf 58a0 779e 0d04 2cbf 2049 c028 2418

0010340 949b 54ef 77ad 3d05 1e27 31f4 ac10 11e7

0010360 579f 00d1 8824 2fcf 0c01 61f3 47e4 1b99

0010400 d7e3 6c3e af9a 11cf b098 5a11 f95a 3f60

# Dedupable and Compressible

The “Dedupable and Compressible” data patterns are created using buffers that are filled with a random number that is updated every 256 bytes. Also, the random number is incremented for every five 32 bit patterns within the buffer. This buffer is reused to fill the file or device, thus creating a pattern that is both compressible and dedupable. Block size used was 4 KiB.

0000000 **7a62 9c9b 7a62 9c9b 7a62 9c9b 7a62 9c9b**

0000020 **7a62 9c9b** **7a62 9d9b 7a62 9d9b 7a62 9d9b**

0000040 **7a62 9d9b 7a62 9d9b** **7a62 9e9b 7a62 9e9b**

0000060 **7a62 9e9b 7a62 9e9b 7a62 9e9b** 7a62 9f9b

0000100 7a62 9f9b 7a62 9f9b 7a62 9f9b 7a62 9f9b

0000120 7a62 a09b 7a62 a09b 7a62 a09b 7a62 a09b

0000140 7a62 a09b 7a62 a19b 7a62 a19b 7a62 a19b

0000160 7a62 a19b 7a62 a19b 7a62 a29b 7a62 a29b

0000200 7a62 a29b 7a62 a29b 7a62 a29b 7a62 a39b

0000220 7a62 a39b 7a62 a39b 7a62 a39b 7a62 a39b

0000240 7a62 a49b 7a62 a49b 7a62 a49b 7a62 a49b

0000260 7a62 a49b 7a62 a59b 7a62 a59b 7a62 a59b

0000300 7a62 a59b 7a62 a59b 7a62 a69b 7a62 a69b

0000320 7a62 a69b 7a62 a69b 7a62 a69b 7a62 a79b

0000340 7a62 a79b 7a62 a79b 7a62 a79b 7a62 a79b

0000360 7a62 a89b 7a62 a89b 7a62 a89b 7a62 a89b

0000400 7e5f 4651 7e5f 4751 7e5f 4751 7e5f 4751 << 256 byte boundary

………

0010000 **7a62 9c9b 7a62 9c9b 7a62 9c9b 7a62 9c9b** << 4KiB byte boundary

0010020 **7a62 9c9b** **7a62 9d9b 7a62 9d9b 7a62 9d9b**

0010040 **7a62 9d9b 7a62 9d9b** **7a62 9e9b 7a62 9e9b**

0010060 **7a62 9e9b 7a62 9e9b 7a62 9e9b** 7a62 9f9b

0010100 7a62 9f9b 7a62 9f9b 7a62 9f9b 7a62 9f9b

0010120 7a62 a09b 7a62 a09b 7a62 a09b 7a62 a09b

0010140 7a62 a09b 7a62 a19b 7a62 a19b 7a62 a19b

0010160 7a62 a19b 7a62 a19b 7a62 a29b 7a62 a29b

0010200 7a62 a29b 7a62 a29b 7a62 a29b 7a62 a39b

0010220 7a62 a39b 7a62 a39b 7a62 a39b 7a62 a39b

0010240 7a62 a49b 7a62 a49b 7a62 a49b 7a62 a49b

0010260 7a62 a49b 7a62 a59b 7a62 a59b 7a62 a59b

0010300 7a62 a59b 7a62 a59b 7a62 a69b 7a62 a69b

0010320 7a62 a69b 7a62 a69b 7a62 a69b 7a62 a79b

0010340 7a62 a79b 7a62 a79b 7a62 a79b 7a62 a79b

0010360 7a62 a89b 7a62 a89b 7a62 a89b 7a62 a89b

0010400 7e5f 4651 7e5f 4751 7e5f 4751 7e5f 4751

# Irreducible

The “Irreducible” data patterns are created using buffers that are filled with random numbers, the buffer is not reused. Every 32 bits is a new random number for all of the buffer. Each time a buffer is written, the buffer is then re-filled with more random numbers. The random number generator returns a value that covers 2^31 values. This in combination with a block size of 1k would mean that the total random space is 2^(31+10) values. ~2TB. If the block size is 4 KiB, then this would mean that the total random space is 2((31+13) values. ~16TB.   
As long as the files being written are smaller than these sizes, then the random number fill should be quite reliable for producing a high quality measurement.

0000000 536f 6b29 d35f 6624 365c 7bb1 1175 2f68

0000020 30a3 5952 4b25 22f7 7c12 1966 8ec5 175b

0000040 337d 7bfa 9321 674a 9081 5101 2eb3 3db6

0000060 096c 03b4 935a 20d6 0a8d 6764 e402 58f0

0000100 6cec 2f09 1214 13ce e934 3e04 74bb 346a

0000120 bbdf 3915 4ac0 41e1 a50a 2df2 4b17 1531

0000140 dddf 5738 8835 566e 7be7 72b0 b584 2096

0000160 fdd3 0666 2aed 53a2 5880 3bb0 5a1d 3682

0000200 41e0 27ea ed99 048a da23 3cf4 4f18 6f1d

0000220 e9a5 6d33 87a5 6b30 989d 3f19 8bc8 2972

0000240 0cba 1f37 9758 5724 3f1a 234c d610 62c2

0000260 e717 6573 cb0a 1c48 0c07 6f0a d884 7c9b

0000300 0205 5ba4 c543 6c80 e95b 7a16 93fb 7632

0000320 805c 7291 52c7 2e82 9eab 6e08 24fc 77e9

0000340 59e4 6b7b bedc 6f4a 9699 1438 212e 0f27

0000360 5ac7 51f3 e8d9 3fcc 1f48 25cb c35a 3a72

0000400 67b0 4477 76ed 7b38 00ce 3808 f11f 156c

0000420 3536 234a 7e53 5ceb aacd 68b2 ae6e 29eb

Non-repeating for the entire file.