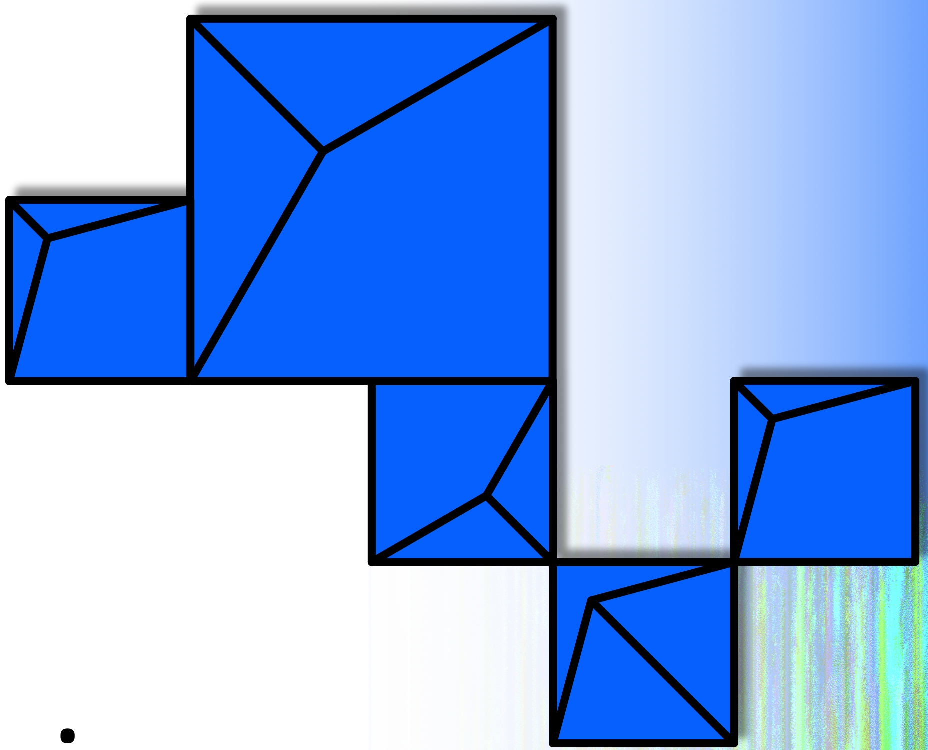


Leveraging z15 Hardware Compression

Daniel Gritter

Disclaimer

Agenda

Background

Compression Technology

Using Compression APIs

Compression Use Cases

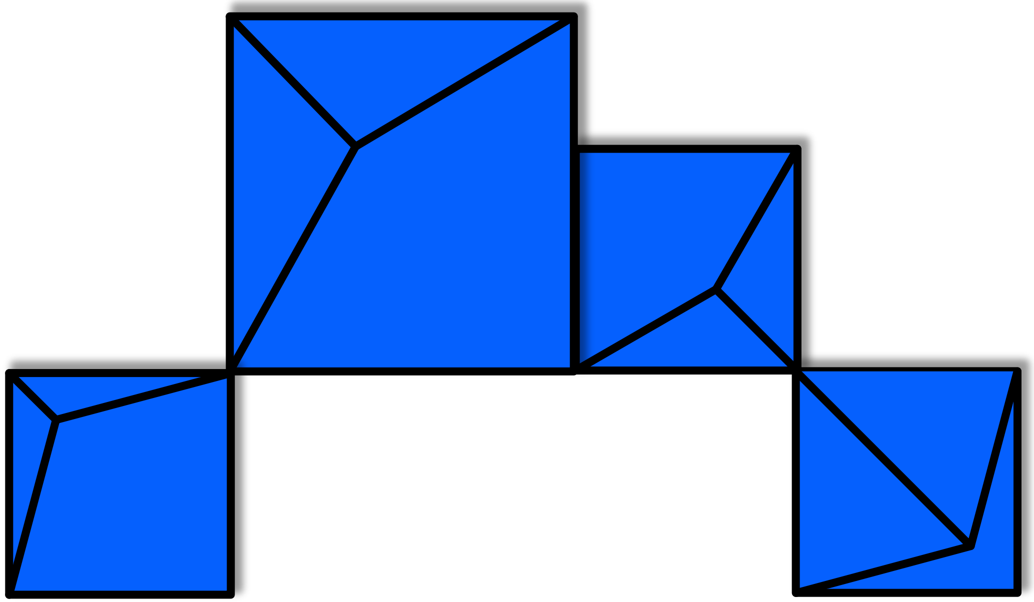
Background

On IBM z15™, with the Integrated Accelerator for zEnterpriseData Compression, the industry standard compression used by zEDC is now built into the z15 core

The new DFLTCC instruction allows z/TPF to easily leverage compression technology, just like CPACF did for encryption

Now you can have the best of both worlds with compression and encryption (in that order) right on the processor cores.

Encryption becomes even less expensive, since after compression, there is much less data to encrypt.



Compression Technology

Technical Details

- DFLTCC provides two modes of compression
  - Fixed Huffman
  - Dynamic Huffman
- **Fixed Huffman** excels for smaller data sizes and fastest compression time
  - Does not need to encode custom Huffman table in the output
- **Dynamic Huffman** excels at targeted optimization focused on optimizing compression ratio by analyzing data prior to compression
  - Must encode generated Huffman table as part of output.
  - Different Huffman tables may be used for different sections of the data.

Technical Details

- How does it compress data?
- Bit encoding:
  - Huffman encoding replaces sequences of common bytes with a bit specification, for example letter “E” may be expressed as a 3 bit sequence instead of a 1 byte letter
- String matching
  - Can be as simple as repetition of a space or 0 character
  - More complex string matching can also be achieved
  - Must occur within a 256 kb range currently
- Future hardware support may produce different encoding for more optimal results
  - Guaranteed backward compatible for inflate z15 will be able to inflate data deflated on a future processor
  - zlib before PJ45872 can also inflate data deflated with z15.

Technical Details

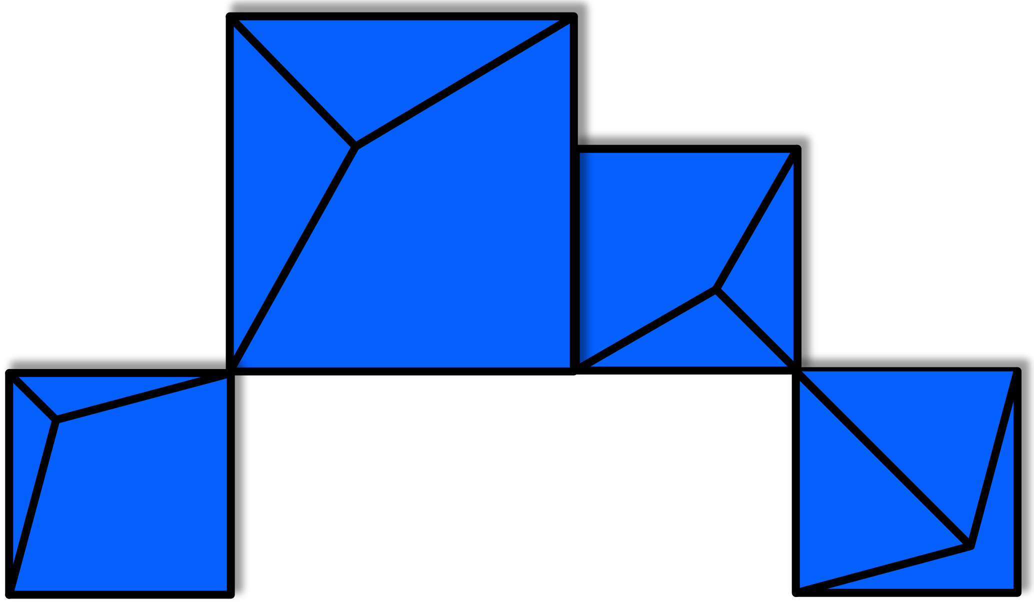
Huffman Table example

Symbol	A	B	C	D

¼ ½

ZIP, GZIP, ZLIB, DEFLATE

- DEFLATE
  - Raw compressed stream – no ID, error check
- ZLIB
  - 2 byte header, 4 byte trailer around a DEFLATE stream
  - ADLER32 error check
- GZIP
  - Can store original file information
  - 10 byte header
  - 8 byte footer
  - CRC32 error check
- ZIP
  - Can store multiple files
  - Support for various compression algorithms, including DEFLATE
  - Generally for file storage / archiving



Using Compression APIs

PJ45872, released 3Q2019 with z15 hardware launch

- Support for **zlib version 1.2.11** now included with the z/TPF product
- zlib 1.2.11 does not require z15 but can leverage it when available
- Both fixed and dynamic Huffman compression available
  - Z_FIXED specified as the compression strategy on deflateInit2() for fixed
  - Z_DEFAULT_STRATEGY or standard deflateInit() api.
- Some zlib strategies not supported with hardware
  - Window bits != 15
  - Z_NO_COMPRESSION level
  - Z_FILTERED, Z_HUFFMAN_ONLY, and Z_RLE strategies
- New **tpf_compress()** and **tpf_expand()** APIs to more directly access DFLTCC instruction
  - Produces a raw deflate stream, no GZIP or ZLIB headers

Technical Details

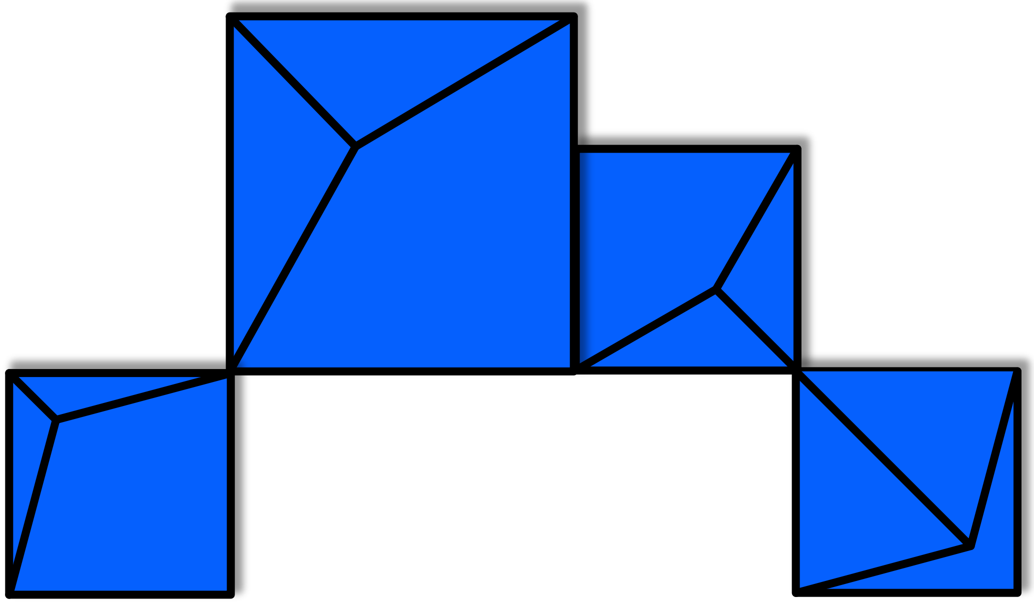
- Support for zlib version 1.2.11 now included with the z/TPF product
  - PJ45872, released 3Q2019 with z15 hardware launch
  - Previously optional component
- zlib 1.2.11 does not require z15 but can leverage it when available
- Both fixed and dynamic Huffman compression available
  - Z_FIXED specified as the compression strategy on deflateInit2 for fixed
  - Z_DEFAULT_STRATEGY or standard deflateInit api.
- Some zlib strategies not supported with hardware
  - Window bits ! 15
  - Z_NO_COMPRESSION level
  - Z_FILTERED, Z_HUFFMAN_ONLY, and Z_RLE strategies

Technical Details

- Java 3Q2019 refresh (PJ45880 prereq) will automatically use z15 compression technology if available
- Expanding JAR files uses native zlib – faster startup time with lower CPU consumption
- GZIPOutputStream / InputStream automatically use native zlib
- DeflaterOutputStream / InputStream automatically use native zlib
  - Care should be used about buffer sizes - hardware will do a better job of optimizing results for a larger buffer size.
  - Benefits of using hardware minimal for the default 512 byte buffer size

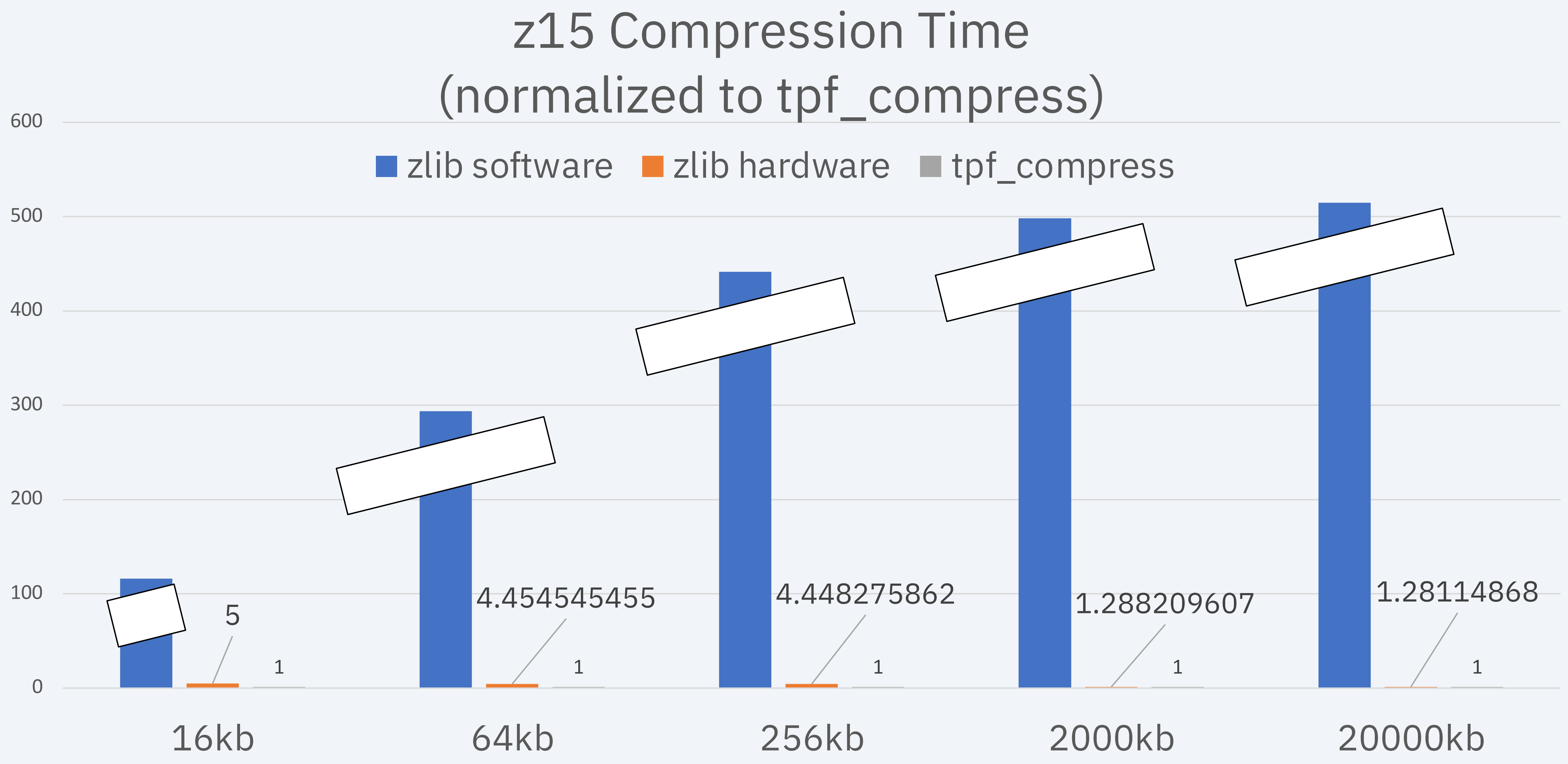
Technical Details

- Software vs Hardware
  - Hardware compression ratio generally does not compress data quite as well as the lowest software level currently but still relatively close
  - Can override at a process level using _HZC_COMPRESSION_METHOD environment variable
  - Hardware threshold value – “don’t bother trying to use hardware if the buffer is too small”
    - _HZC_DEFLATE_THRESHOLD environment variable sets custom compression threshold (default is 256 bytes for zlib)
    - _HZC_INFLATE_THRESHOLD environment variable sets custom expansion threshold (default is 256 bytes for zlib)

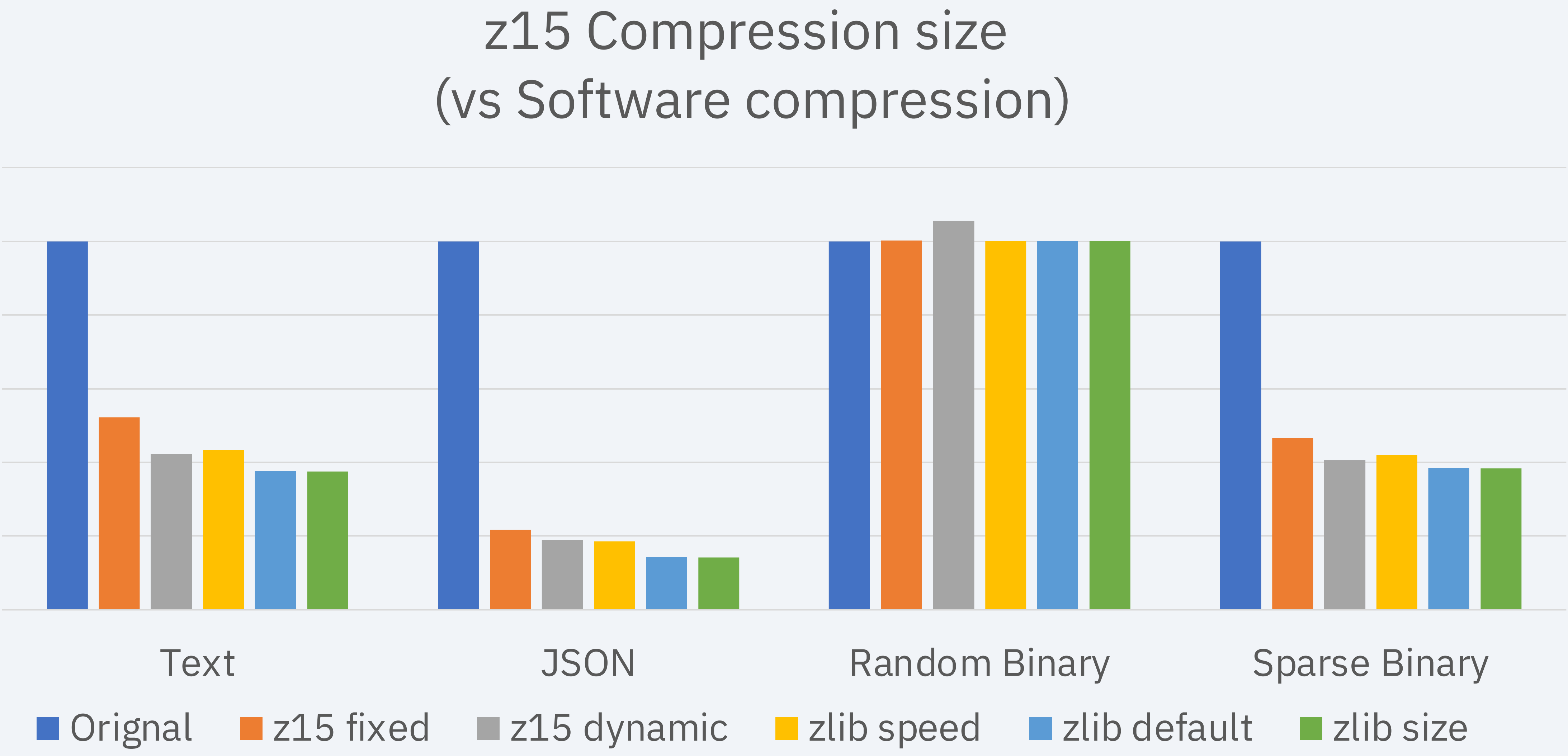


How good is it?

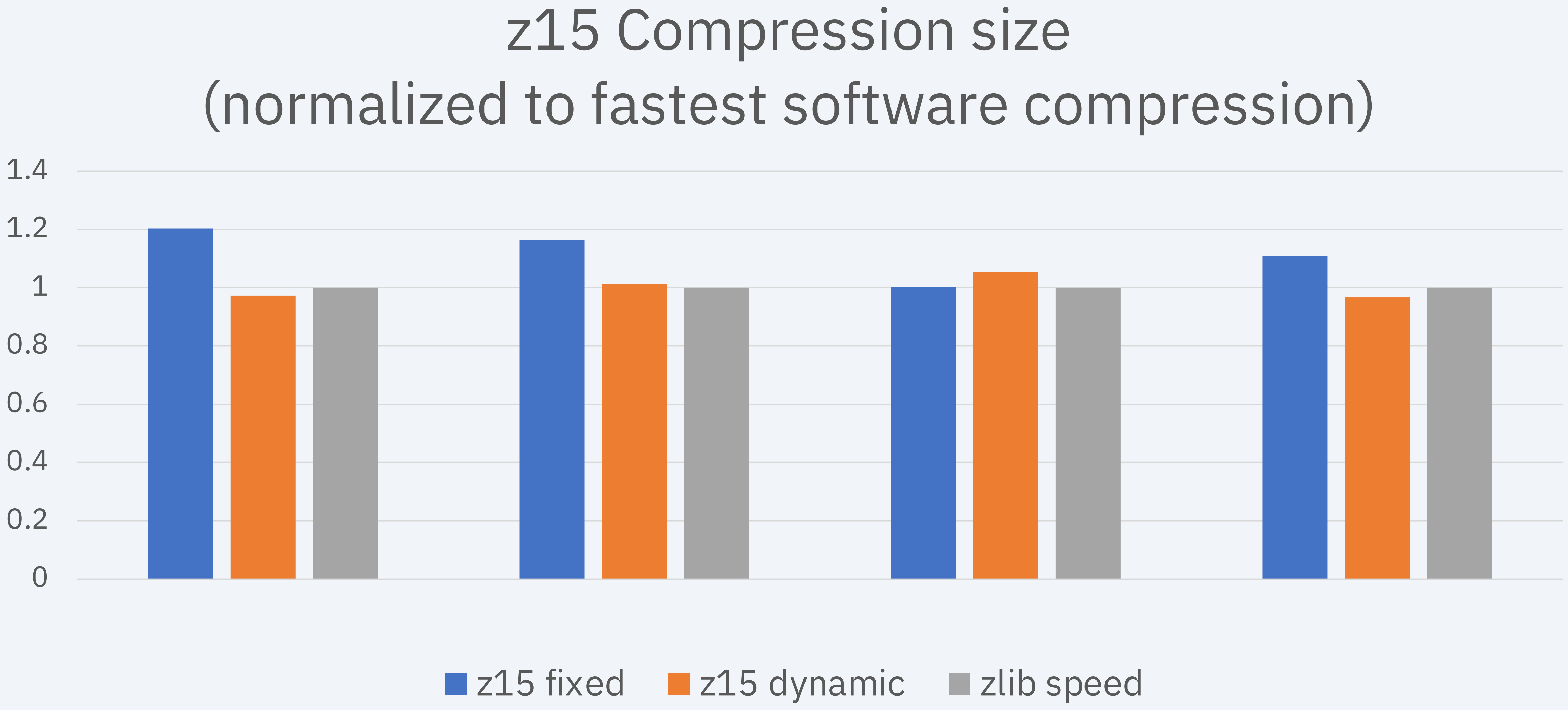
Technical Details

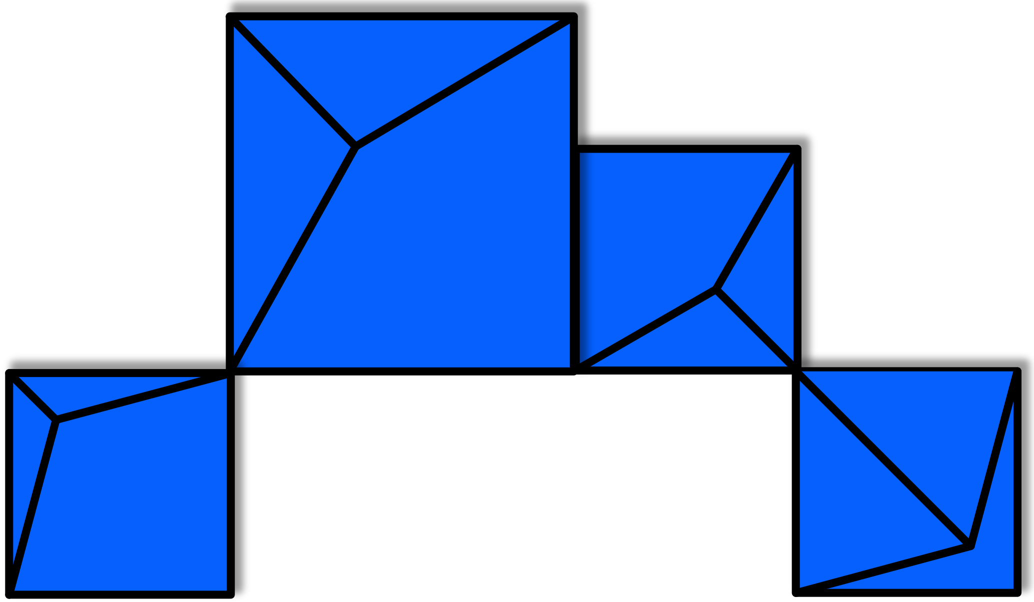


Technical Details



Technical Details





Compression Use Cases

What’s next?

zlib compression is **ubiquitous** in data transmission

MongoDB

Kafka

GRPC

HTTP

OpenSSL

MQ

....

All transports leveraging / supporting zlib / z15 compression –

z/TPF is ready to support your enterprise connectivity.

AND

With the power of z15 anytime you’re encrypting data, it’s usually faster/cheaper to compress it first.

What’s next?

zlib compression is **perfect** for data storage

Filesystem

Cache

Initialization templates

Large database records

Passing large data sets in memory

....

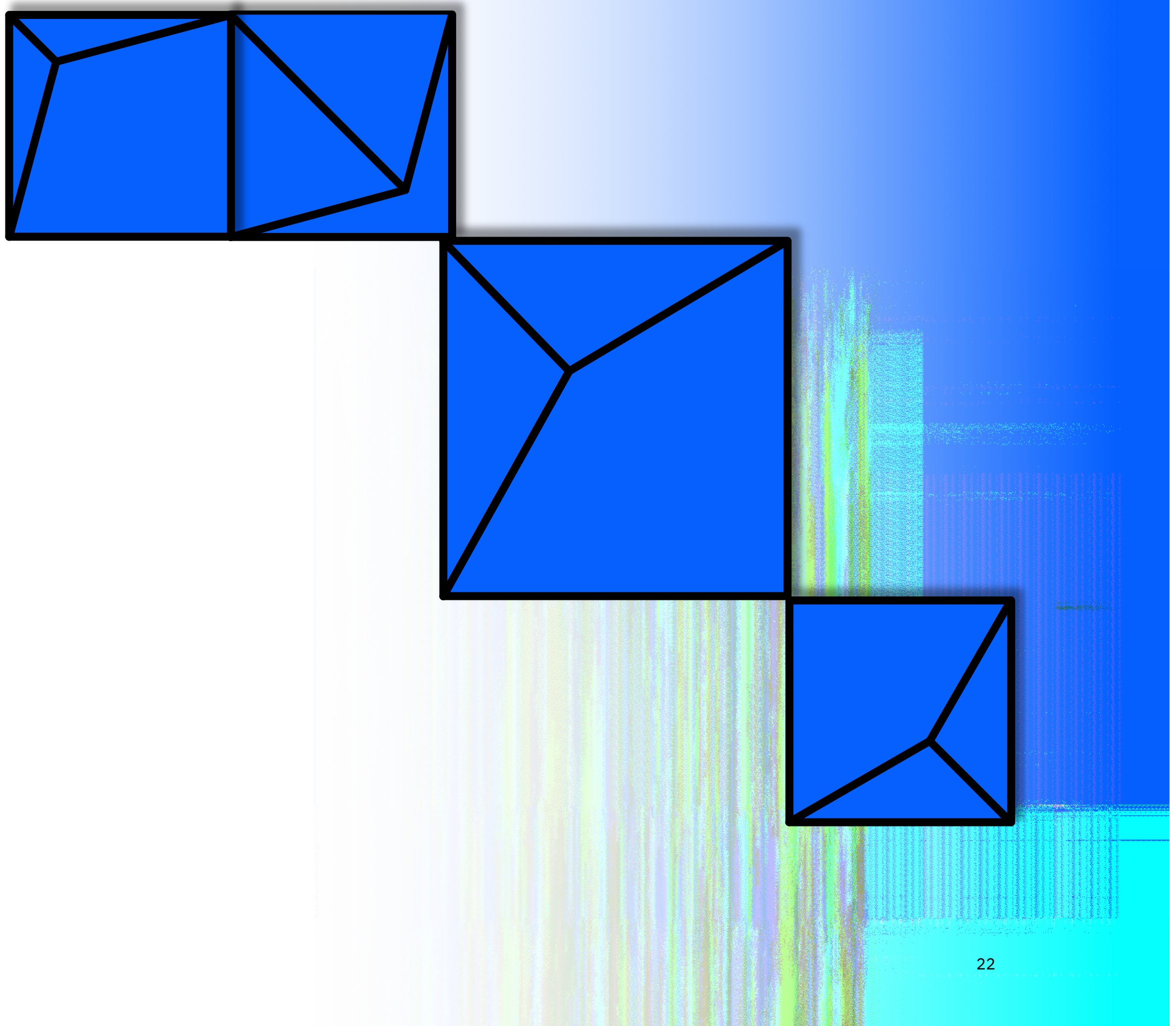
If you’re writing to disk and can reduce it by even 1 IO

Or maybe you’re writing encrypted data now when you were counting on storage-level compression before

For large data, it can actually be faster than moving the data **once.** If you’re moving the data multiple times, compress it.

Thank You

Questions   Comments



Trademarks

IBM, the IBM logo, ibm.com and Rational are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

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.

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 presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.