PRESENTER STEPS:

- Ensure your backup presenter is available
- Present your slides or pre-recorded talk by:
  - selecting "Present now";
  - selecting “Chrome tab,” “A window,” or if needed, "Your entire screen;" and
  - selecting "Share"
  - ensure a Google participant starts recording the presentation.
- If applicable, remove participants by:
  - [Non-Googlers] asking a Google participant to remove participants
  - [Googlers] selecting "Remove from meeting" in a participant's thumbnail image

TECHNICAL CONSIDERATIONS:

- Turn on captions by:
  - selecting "Turn on captions"
- If you experience quality issues, change your send and receive resolutions to 360p by:
  - selecting "Settings"
  - selecting "Video" and then the setting you want to change

zlib: how fast can you go?

Adenilson Cavalcanti (cavalcantii@chromium.org and adenilson.cavalcanti@intel.com)

BSc. MSc.

Chromium/AOSP zlib maintainer

Notices & Disclaimers

Performance varies by use, configuration and other factors. Learn more on the Performance Index site.

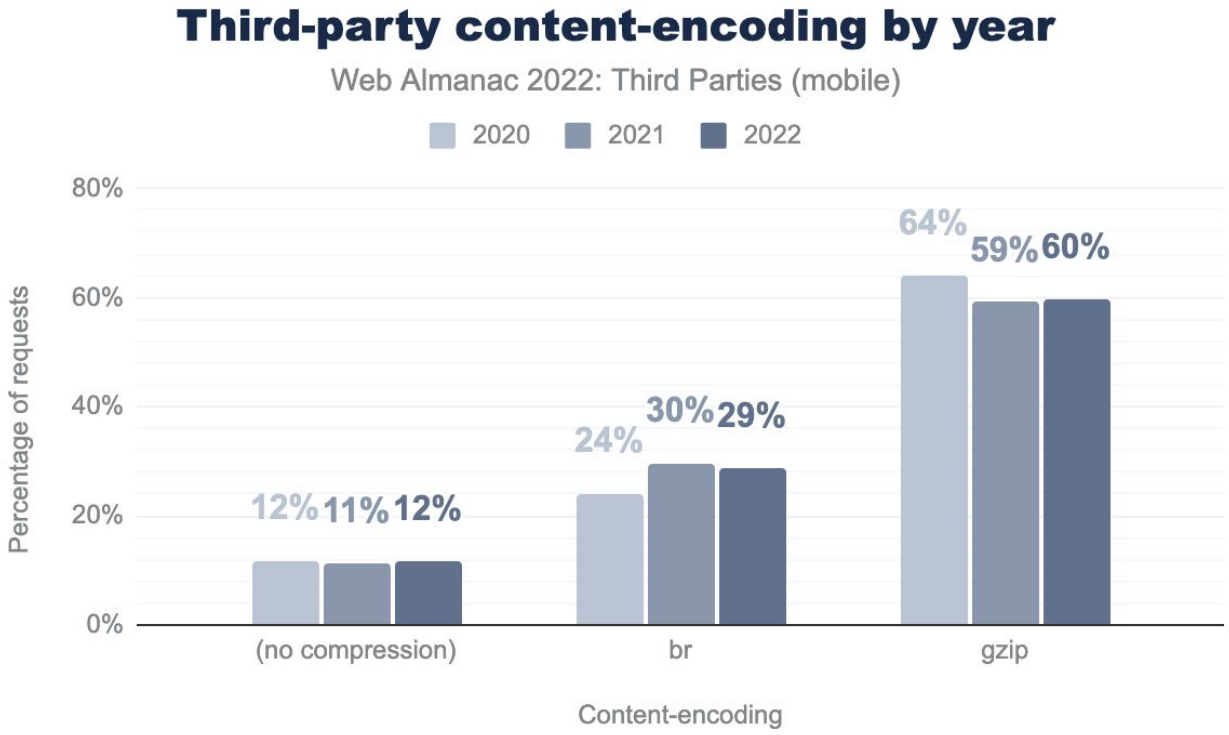
Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.

Your costs and results may vary.

Intel technologies may require enabled hardware, software or service activation.

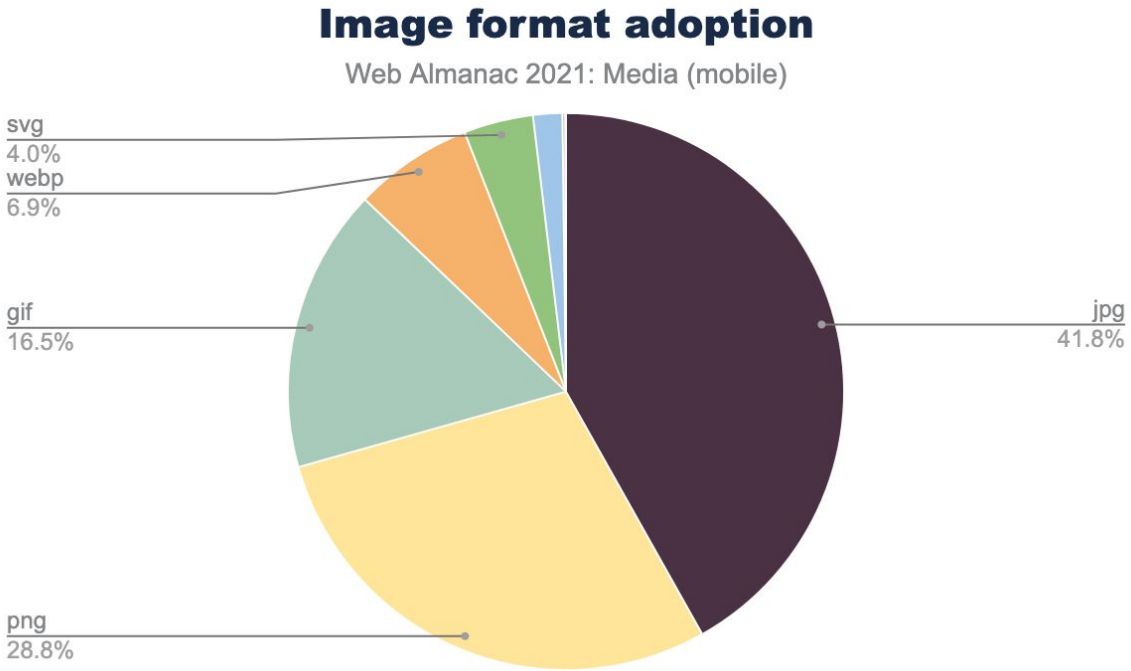
Why care about zlib

“Gzip is still the most popular compression type, i.e. 59% of scripts and 68% of CSS are compressed with Gzip.”



https://almanac.httparchive.org/en/2022/third-parties

“The old formats still reign: JPEG dominates, with PNG and GIF rounding out the podium. Together, they account for almost 90% of the images on the web”



https://almanac.httparchive.org/en/2021/media#format-adoption

Objectives

- Share the maintenance work performed in the last 6 months.
- Report performance improvements.
- Show original research in leveraging hardware acceleration on 4th Gen Intel® Xeon® processors.

Chromium zlib: context

- Heavily patched on top of canonical zlib.
- Leverages SIMD optimizations for both x86 and Arm architectures.
- Up to 3x faster decompression and +30% compression.
- In active development since 2017 (near 6 years already!).
- Used everywhere: Chromium, AOSP, node.js, V8, Skia, Google*.

*since late september.

Security and maintenance

Security: CVE-2022-37434

- Quote from NIST: “buffer overflow in inflate via a large gzip header extra field”.
- Severity score: 9.8 Critical.
- Fortunately Chromium doesn’t use the impacted API (i.e. inflateGetHeader()).
- Reported on 21th Aug, patch ready and tested on 23th Aug, landed on 25th Aug.
- Backported fix to M104 through M106.

Security: CVE-2022-37434

- Patched and added unit test.
- Fuzzer by Hans Wennborg@google.

```
SUMMARY: AddressSanitizer: heap-buffer-overflow /b/s/w/ir/cache/builder/sr
r-rt/lib/asan/asan_interceptors_memintrinsics.cpp:22:3 in __asan_memcpy
Shadow bytes around the buggy address:
  0x0c087fff9210: fa fa fd fd fd fd fd fd fa fa fd fd fd fd fd fd
  0x0c087fff9220: fa fa fd fd fd fd fd fd fa fa fd fd fd fd fd fd
  0x0c087fff9230: fa fa fd fd fd fd fd fd fa fa fd fd fd fd fd fd
  0x0c087fff9240: fa fa 00 00 00 00 00 00 fa fa fd fd fd fd fd fd
  0x0c087fff9250: fa fa 00 00 00 00 00 fa fa fa 00 00 00 00 00 00
=>0x0c087fff9260:[fa]fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c087fff9270: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c087fff9280: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c087fff9290: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c087fff92a0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c087fff92b0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack after return:      f5
  Stack use after scope:   f8
  Global redzone:          f9
  Global init order:       f6
  Poisoned by user:        f7
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
==29051==ABORTING
[87/92] ZlibTest.InflateCVE (CRASHED)
[88/92] ZlibTest.DeflateStored (0 ms)
```

Security: sync with zlib 1.2.13

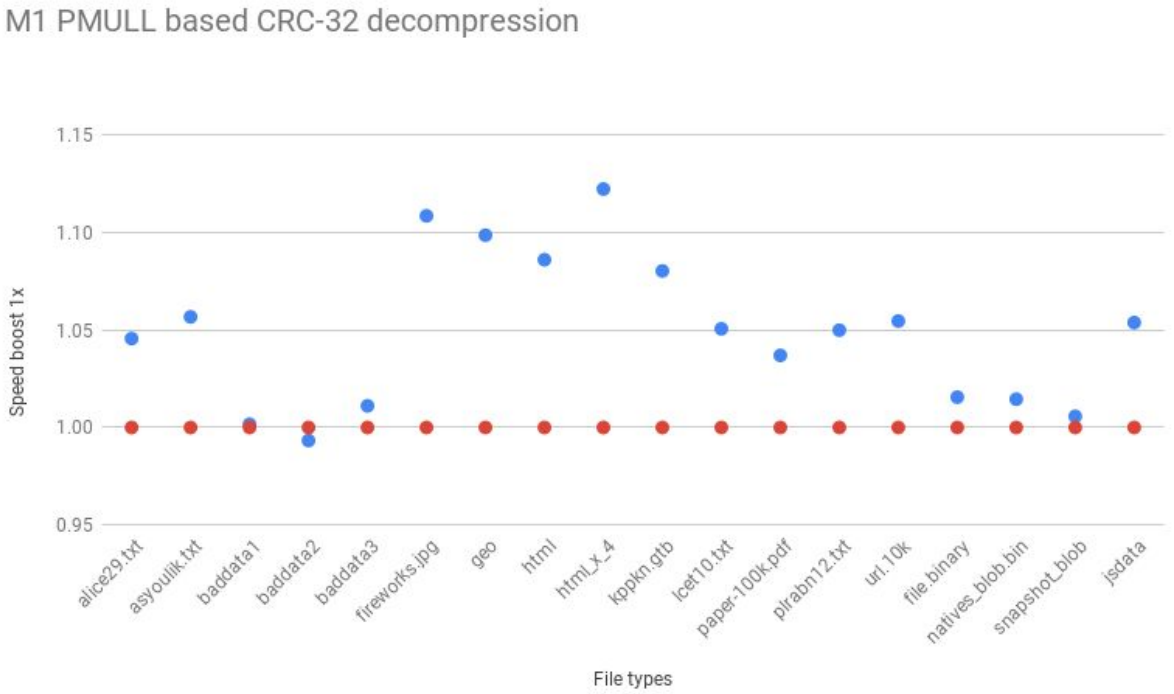
- Canonical zlib was released on October 13th, 2022.
- We were partially in sync with ‘devel’ branch due the aforementioned CVE.
- Completed with two patchsets on November 7th.
- Tremendous effort in previous sync paid dividends.
- Special thanks to Chris Blume (xoogler/retired), Noel Gordon@google and Hans Wenborg@google for reviewing patches.

Performance improvements

Four years in the making: PMULL based CRC-32

- Used polynomial multiplication and Barret transform to calculate CRC-32.
- Started in 2018, sadly Arm processors back then didn’t show improvements.
- In 2020, Apple released the DTK (Developer Transition Toolkit) with A12Z processor.
- Plus new Arm designs improved handling of PMULL instruction.
- Landed in September this year.

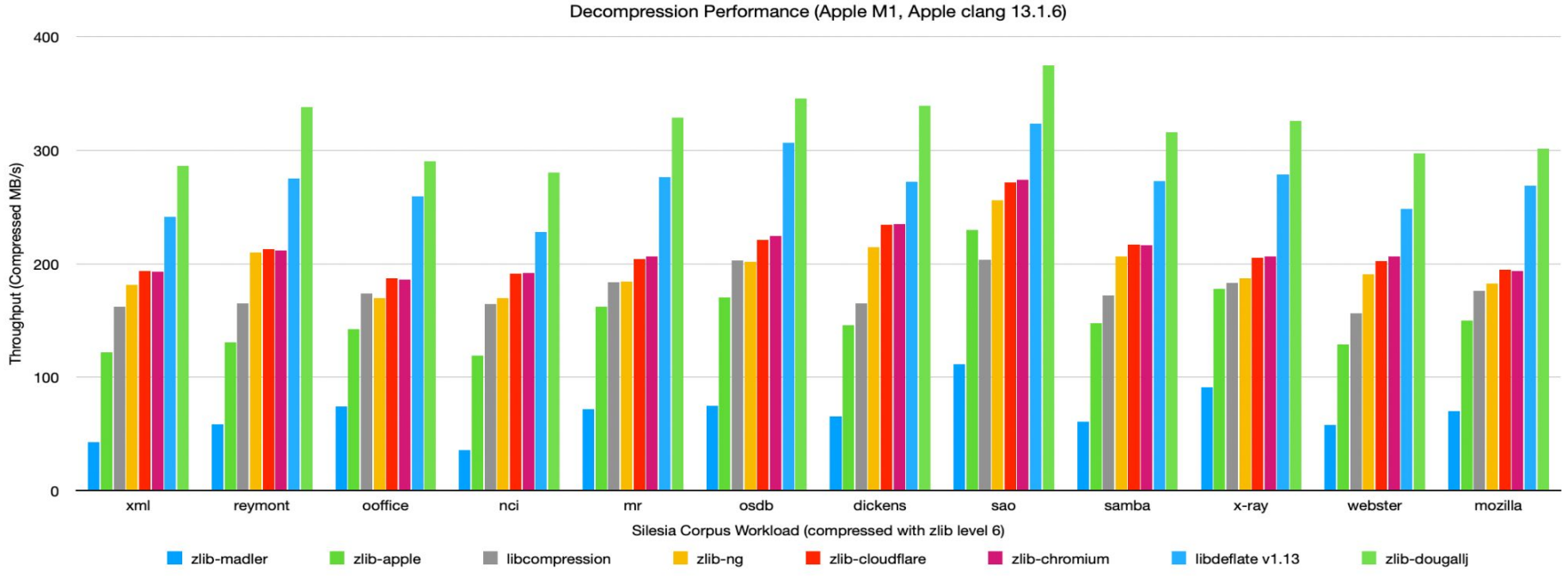
M1: up to +13% faster decompression for HTML



Results may vary.
System configuration: Mac Air late 2020, M1 processor, 16GB RAM.

A new player enters the stage

- Contacted Dougall Johnson in July about performance counters code.
- Dougall started some pretty innovative research on zlib (covered in his blog).
  - Source: https://dougallj.wordpress.com/2022/08/20/faster-zlib-deflate-decompression-on-the-apple-m1-and-x86



First patch: flatter Huffman tables

- Basic idea is to use a bigger root table.
- Avoids branch mispredictions in lookups for Huffman codes.
- Memory price: 1920 bytes* per inflate_state
- Average gains of +5% (Haswell) to +8.3% (Cortex X1@64bit).
- Landed in September.

*Even lower spec boards feature CPUs with 32KB L1 cache size.

Results may vary.
M1 system configuration: Mac Air late 2020, M1 processor, 16GB RAM.
Haswell system configuration: i7-4870HQ, 16GB, late 2015.

Compression work: WIP

- Faster/better multiplicative hash (ANZAC+) suggested by Noel Gordon@google.
- Portable (i.e. consistent results between x86 vs Arm vs others).
- Average gains: +29.2% (Tigerlake), +20% (Cortex X1), +18.6% (M1).
- Got solve failing pixel tests that expect matching results in generated PNGs.

Results may vary.
M1 system configuration: Mac Air late 2020, M1 processor, 16GB RAM.
TigerLake: i7-1185G7 launched on Q3@2020, 16GB RAM.
Cortex X1 system configuration: Pixel 6a.

New frontiers: hardware accelerated data compression

4th Gen Intel® Xeon® new tech

- IAA (In-Memory Analytics Accelerator)
- QAT (Quick Assist Technology)
- And many more.
- To learn more about: https://youtu.be/3zyQNiJw82U?t=362

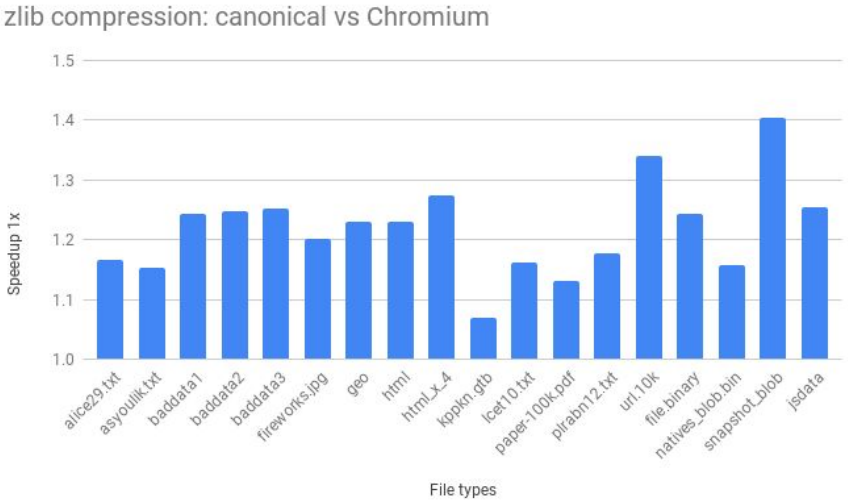
Could be deployed to speed up some operations in zlib?

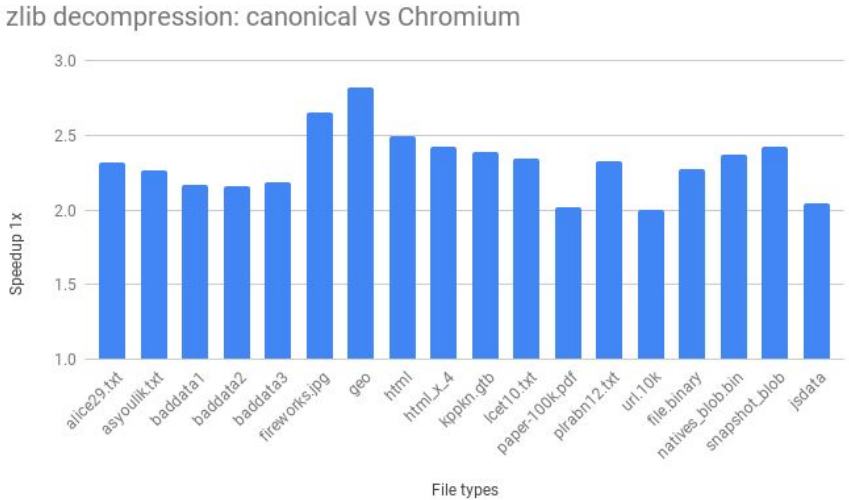
Initial PoC (Proof of Concept)

Disclaimers:

- Pre-production hardware (reported values are useful only for comparing zlib vs chromium vs hardware accelerated).
- Lots of work is still needed to achieve production-ready code.
- Xeon® (i.e. server) specific optimizations.
- Special thanks to Chengfei Zhu@intel and Joel Schuetze@intel for sharing code and knowledge on Xeon 4th gen accelerators (You guys rock!).

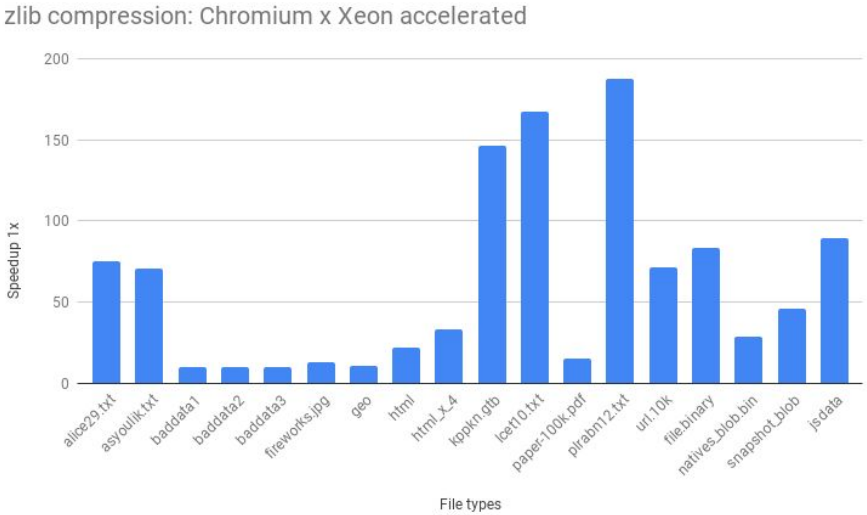
Vanilla vs Chromium: avg +21% comp. and 2.3x faster decomp.

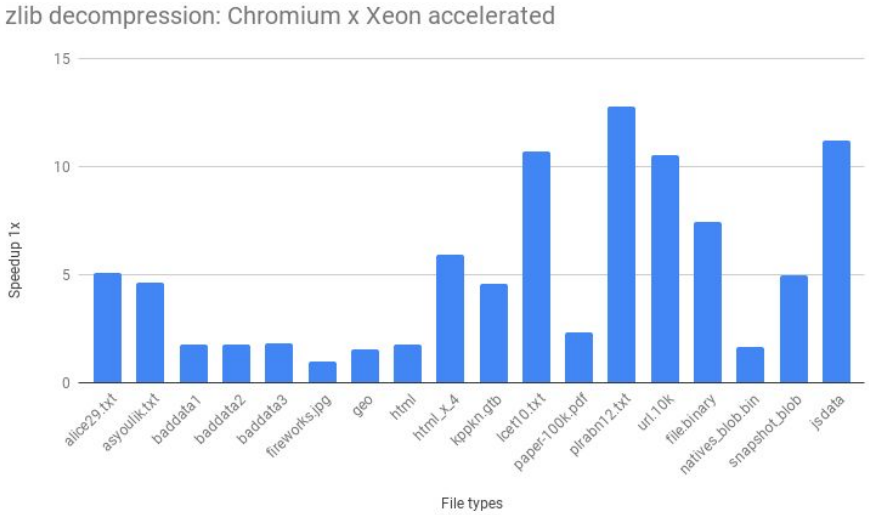




Results may vary.
SPR system configuration: two sockets, using only 1 accelerator, 100GB DDR5 ram.

Chromium x hardware: avg 60x compression and 5x decomp.!





Results may vary.
SPR system configuration: two sockets, using only 1 accelerator, 100GB DDR5 ram.

What is next in hardware acceleration

- Handle streaming cases.
- Testing.
- Option for opt in to stable data stream (i.e. compression).
- Issue: [zlib][x86] Explore new Xeon features to accelerate zlib
  https://bugs.chromium.org/p/chromium/issues/detail?id=1381000

Chromium zlib: next steps

Chromium zlib: a never ending journey

- Software: decompression + compression work to be completed.
- Hardware: early beginnings, so much potential.
- Patches are welcome!

Special thanks

- Chromium zlib team (Z-team): Chris Blume (xoogler/retired), Noel Gordon (google), Hans Wennborg (google).
- Intel team mates: Chengfei Zhu, Joel Schuetze, Tomasz Paszkowski, Gustavo Espinoza Quintero, Daniel Byrne.
- Jean-loup Gailly and Mark Adler (canonical zlib maintainer) who created and maintained canonical zlib in the last +27 years.
- My family (wife + daughter) being so understanding of my zlib obsession. :-)