COMPRESSION USING ON-CHIP HARDWARE

INTEL QUICKASSIST TECHNOLOGY (QAT) [MOSTLY] AND CAVIUM ZIP COPROCESSOR

DAVID ABDURACHMANOV
Crypto/Compression Coprocessor

- Crypto/Compression coprocessors are becoming commodity. Who has them?
  - AMD Opteron A1100 (aarch64), 2016, Crypto/Compression, in SOC
  - Intel QuickAssisst Technology (QAT) (x86_64), 2013, Crypto/Compression, big push with Xeon / Xeon D in 2017, not in CPU, external chip/SOC/chipset
  - Cavium ThunderX, OCTEON TX (aarch64), 2016, Compression, in SOC
  - More will show up later this year or next year
A GLIMPSE INSIDE THE INTEL® XEON® PROCESSOR SCALABLE FAMILY PLATFORM

Fabric
Intel® Omni-Path Architecture

Networking
Intel® Ethernet

Accelerators
Intel® QuickAssist
Intel® AVX-512

SSDs
Intel® Optane™ SSD DC P4800X
Intel® SSD DC P4600
Intel® VMD

Complementary
Intel® FPGA
Intel® Nervana™
Intel® Xeon Phi™
Intel® Silicon Photonics

Integrated options

Workload optimized frameworks & telemetry
(e.g. Caffe*, Intel® DAAL, Intel® MKL, DPDK, Intel® RDT, SNAP*, SPDK)

PERFORMANCE
SECURITY
AGILITY
EFFICIENCY

ADVANCING VIRTUALLY EVERY ASPECT: BRAND NEW CORE, CACHE, ON-DIE INTERCONNECTS, MEMORY CONTROLLER & MORE
Two high common algorithms LZO and DEFLATE (ZLIB)

Hardware vendors are implementing DEFLATE in hardware. Two possible interfaces:

- Common (Intel, Cavium): zlib-shim (but different strategies).
- OpenDataPlane (ODP) Compression API proposed in Budapest in April by Cavium. Works for networking and storage.
QAT HIGH LEVEL OVERVIEW

- Highly configurable, lots of knobs
- Not fully drop-in replacement for zlib
- Requires changes on user-side application
- Provides software fallback, but not triggered in all cases
- ROOT 6.10.00 works with QAT
#include "zlib.h"
#include "TFile.h"
#include "TTree.h"
#include "Compression.h"

int main(void) {
    zlibSetupEngine(Z_QAT_DEFAULT_POLLING_INTERVAL, Z_QAT_DEFAULT_MAX_NUM_RETRIES);
    zlibStartupEngine(Z_HW_COMP_HW_DECOMP);

    TFile f("tree1.root", "recreate");
    TTree t1("t1", "random data");
    f.SetCompressionAlgorithm(ROOT::ECompressionAlgorithm::kZLIB);
    f.SetCompressionLevel(9);
    Int_t ran;
    t1.Branch("ran", &ran, "ran/I");

    for (unsigned int i = 0; i < 100'000'000; i++) {
        ran = i;
        t1.Fill();
    }

    t1.Write();
}

zlibShutdownEngine();
return 0;
SIMPLE EXAMPLE RESULTS

- CPU: Intel(R) Xeon(R) CPU E5-2690 0 @ 2.90GHz, SandyBridge, TB up to 3.8GHz, intel_pstate, DH895XCC QAT card via PCIe
- Compression:
  - **SW:** 113.70s user 0.36s system 99% cpu 1:54.78 total
  - **HW:** 11.40s user 1.69s system 68% cpu 19.172 total
- Hardware delivered the speed, but we lost in compression ratio: from 200M (**SW**) compressed to 230M (**HW**)  
- Decompression:
  - **SW:** 3.86s user 0.31s system 88% cpu 4.729 total
  - **HW:** 4.35s user 0.67s system 55% cpu 9.091 total
- Currently I don't see benefits in decompression part.
RUNNING IN USER-MODE

- Everything works nice with root-permissions
  - But we don't run CMSSW in with root permissions
- There are no official documentation on how to use zlib-shim from user land, but we found minimal solution:
  - Creating special group, making /dev/* and device configuration owned by new user, changing permissions, and using Linux CAP_IPC_LOCK capability for locking the memory.
  - CMSSW relies heavily on LD_LIBRARY_PATH and Linux capabilities will ignore LD_* for security.
  - We developed two methods to get CMSSW compiled with RPATH.
  - QAT zlib-shim is available in CMSSW DEVEL builds, but requires user group build kernel part and do the configuration
CAVIUM ZIP COPROCESSOR

- Different strategy: a direct drop-in replacement for zlib (to be verified).
  - If it doesn't work as drop-in replacement then it's a bug (Cavium statement).
- No extra functions in zlib interface, no device configuration, a single kernel driver
- No software fallback (because it's part of SOC) [unless you remove device from PCIe device list in kernel]
  - Compression results for simple example:
    - **SW**: 1042.69s user 0.77s system 99% cpu 17:24.42 total
    - **HW**: 25.18s user 2.00s system 95% cpu 28.534 total
  - Decompression (HW shows 0 request received for decompression):
    - **SW**: 16.96s user 1.02s system 93% cpu 19.159 total
    - **HW**: 17.38s user 0.60s system 96% cpu 18.633 total
  - Lost in compression ratio: 192M (**SW**) to 236M (**HW**)  
  - We received zlib-shim less then a week ago: number of issues/suggestions reported, meetings with the Cavium teams, hardware issues with 2S machines (will provide access to CERN)
HARDWARE LIMITS: TIME

- The current setup needs to go via PCIe to external QAT card. What are the limits?

QAT does not perform well for small buffers

Fat zlib in CMSSW (SSE4.2, PCLMUL, AVX, AVX2) does better

- This is based on dumps ROOT IO using LLTng + custom while running CMSSW Phase 2 job. Not random data.
Smallest 2906 buffers replayed for compression, logarithmic scale used QAT start flatting out at around 10K buffers
What about compression ratio for small buffers?

QAT compression is worse or negative (i.e. our compressed buffer is up to 20% bigger.)