UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

ziti

Bundesministerium
für Bildung
und Forschung

Assessing the Overhead of Offloading Compression Tasks

Laura Promberger[1] [2], Rainer Schwemmer[1] and Holger Fröning[2]

August 17, 2020

[1]CERN, Experimental Physics Department, Switzerland
[2]University Heidelberg, Institute for Computer Engineering, Germany

This work has been sponsored by the Wolfgang Gentner Programme of the German Federal Ministry of Education and Research (grant no. 05E15CHA)

Overview

In the context of compression accelerators, this work presents

- Part 1: Proposition and Implementation of multi-threaded benchmarks
- Part 2: Analysis of the systems performance while using the accelerator
- Part 3: Performance comparison of the accelerator and the CPU-only compression algorithms
- Part 4: Discussion
- Part 5: Conclusion and Outlook

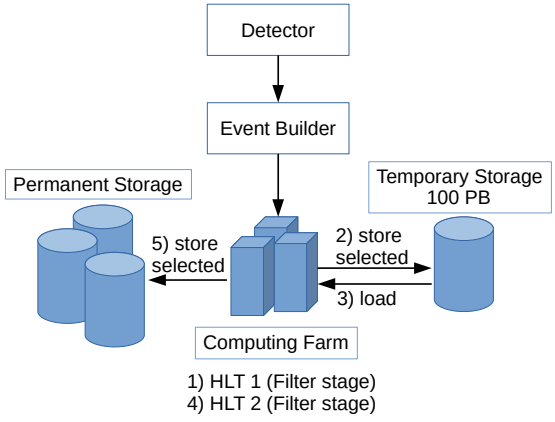
Motivation

CERN

- Upgrade from 2019 - 2021 to support higher luminosity

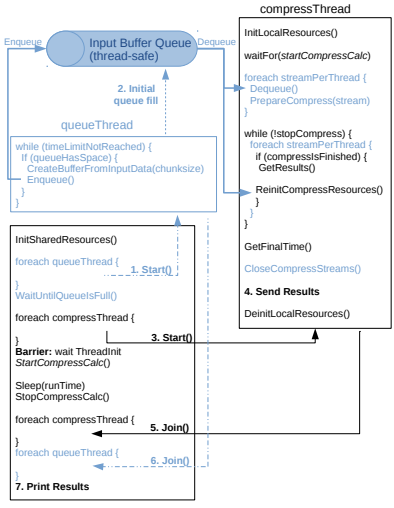
LHCb experiment upgrade

- Detector to compute farm bandwidth: from 60 GB/s to 5 TB/s
- 2 filter stages to filter interesting collisions (data transfer at 150 GB/s)
- 100 PB buffer in between the 2 filter stages



Part 1: Benchmarks

CPU Benchmark

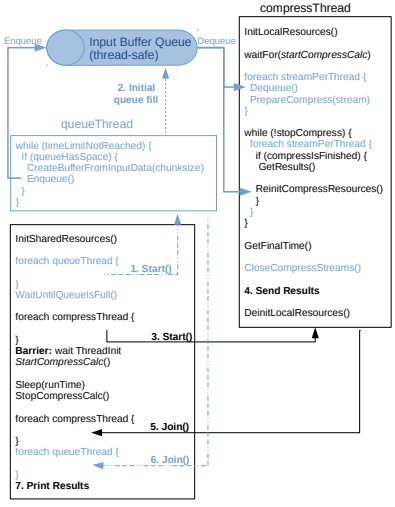


CPU-only compression benchmark

- `main` initializes and controls the threads
- `compressThread` executes compression

  Limitations due to RAM available
- Shared input data set for each NUMA-domain
- Private output buffer for each `compressThread`

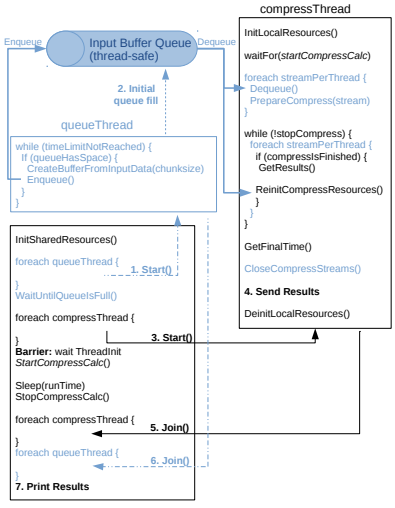
Accelerator Benchmark



Accelerator compression benchmark

- `queueThread` handels provisioning of data chunks
- One `compressThreads` coordinates multiple compression streams
- Uses busy-waiting to interact with the accelerator
- No internal NUMA-binding

Remarks about the Implementation



Measuring sustained throughput

- Time measured within the `compressThread`
- Coordinated by `main`
- Make sure: all threads are set-up before starting the compression and timing

Part 2: Performance Analysis of the Accelerator

Hardware

Server: Intel Xeon E5-2630 v4

- Dual-socket
- Real cores 20, threads per core 2
- 64 GB RAM, Max Memory Bandwidth per Socket 68.3 GB/s
- CentOS 7

Compression Accelerator: AHA378

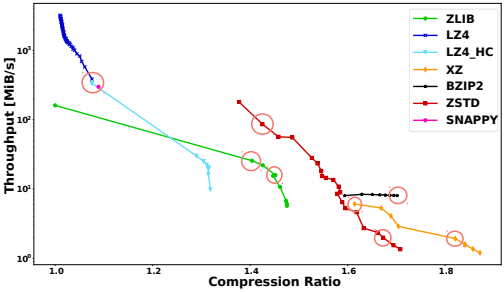
- Up to 80 Gb/s
- 4 FPGAs;
- Half-size, double-width PCI Gen3
- Compression algorithms
  - Lzs
  - Deflate, gzip, zlib

Data Sets

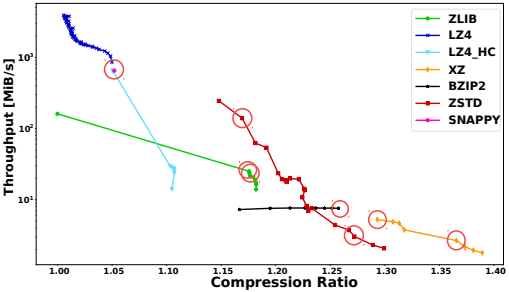
	Compression Corpora	
enwik9	1 GB	English Wikipedia from 2006
proteins	200 MB	Protein data of the Pizza & Chili corpus
silesia	203 MB	Multiple different files tarred into a single file
calgary	3.1 MB	Most famous compression corpora tarred into a single file
	150 MB	Calgary multiplied to conform with benchmark requirements
	High Energy Physics Data	
ATLAS	4.1 GB	Proton-Proton Collisions
ALICE	1 GB	Lead-Lead Collisions
LHCb	4.9 GB	Proton-Proton Collisions

Compression Algorithms

- bzip2 level 9
- lz4(_hc) level 2
- snappy
- xz level 5
- zlib level 2 and level 4
- zStandard (zstd) level 2 and level 21



ALICE

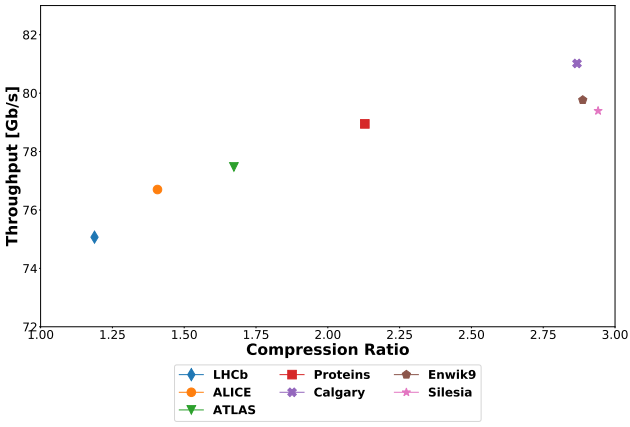


LHCb

Accelerator Results

Best Configuration: 2 MB chunk size, 500 queue elements, 3 `queueThread`, 4 `compressThreads`, 12 compression streams

- Memory bandwidth 22-30 Gb/s
- Power consumption: CPU 58 W, RAM 7 W

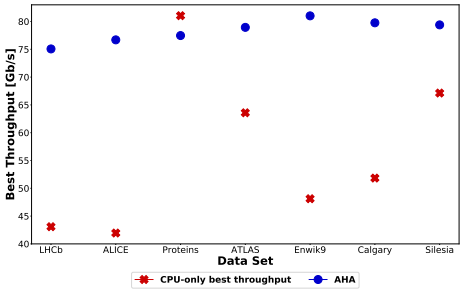


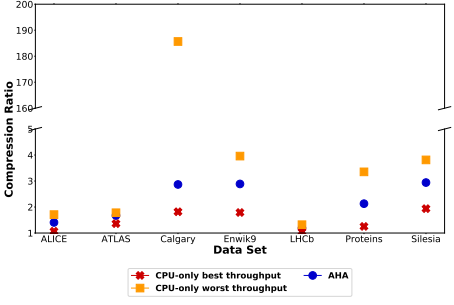
Part 3: Performance Comparison of accelerator vs CPU-only Algorithms

Results Comparison I

CPU-only benchmark best configuration: 40 `compressThreads`, 150 MB chunk size

- Accelerator was compared to zlib level 2. zlib level 4 was not selected because
  - Only 7% better compression ratio
  - But 13 - 37% decrease in throughput
- zstd level 2 compression ratio was close to zlib level 2, but 2 - 3x higher throughput





best: snappy or lz4

worst: zstd level 21 and xz

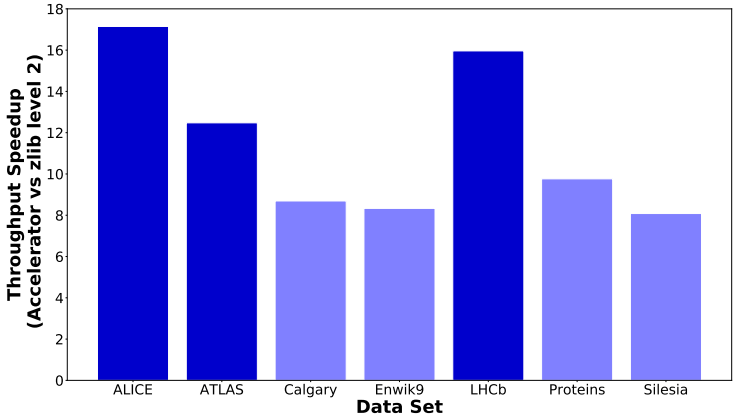
Result Comparison II

- CPU power consumption: 72 - 103 W (TDP 105 W)
- RAM power consumption: 4 W (zlib level 2) - 11 W (xz level 5)

- **Highest CPU power consumption**: zlib level 2 and zstd level 2 (99 - 103 W)
  - This is 40-45 W more than the accelerator
- **RAM power consumption**
  - Accelerator had 2x the memory bandwidth for 7 W compared to CPU-only algorithms
  - For CPU-only algorithms: to get same memory bandwidth 9 W are needed

Result Comparison III

The accelerator had a 8-17x higher throughput than zlib level 2



Part 4: Discussion

Discussion I

- Throughput variation between data sets: 7% accelerator vs 42% CPU-only
  - Higher accelerator throughput possible if PCI Gen4?
  - NUMA binding very important for stable performance
- Related Works (Calgary Corpus, Harmonic mean compression ratio)
  - FPGAs
    - Abdelfattah et al.[1]: 2.17 with 24 Gb/s
    - Qiao et al.[2]: 2.03 with 80 Gb/s
    - AHA378: 2.81 with 80 Gb/s
  - CPU-only algorithms
    - snappy: 1.85 with 53 Gb/s
    - zlib level 2: 2.65 with 9.3 Gb/s
    - zstd level 2: 2.76 with 22 Gb/s
    - zstd level 21: 3.26 with 13 Gb/s
    - xz level 5: 3.38 with 0.58 Gb/s
- Limitations
  - The server did not provide enough reliable performance counter to do further analyses

[1] https://doi.org/10.1145/2664666.2664670; [2] https://doi.org/10.1109/FCCM.2018.00015

Discussion II

For system integration it is important to understand underlying requirements to not degrade the accelerator's performance

- Accelerator needed only 10% of the CPU compute resources (4 `compressThreads`)
- Accelerator needed 80% of memory bandwidth of one socket
  - Note: In general, software utilizing two sockets will not have twice the amount of memory bandwidth available compare to a single socket
- Most compatible tasks for collocation
  - NUMA binding important to prevent unnecessary cross-NUMA-domain communication
  - Best choice: Tasks which have a big computation, but small memory footprint
  - Preferable, utilizing compression as a post-processing step, so that the data already resides in memory

Discussion III

- Benchmark used busy waiting
  - → Interrupt-based implementation might reduce further resource requirements

Accelerators and API Documentation

- Other accelerator tested: Intel QuickAssist 8970
  - Up to 100 Gb/s; ASIC; half-size, double-width PCI Gen3
  - Does encryption and compression
  - Compression algorithms: deflate, gzip, zlib
  - ↯ Not further pursued because of significant performance and stability issues
- AHA378
  - Good documentation, but must be thoroughly read to get good performance
    - → Accelerator access memory via DMA!
  - Examples allow to get a good performance in reasonable time

Part 5: Conclusion and Outlook

Conclusion

- Insights of integration costs of accelerators
  - NUMA binding is important
  - Measuring time is not so straight-forward in multi-threaded environments
- Accelerator vs zlib level 2
  - 80% less CPU resources
  - 8-17x higher throughput
- Commercially available compression accelerators are a realistic option
- The work on CPU-only compression algorithms can be used to select the right algorithm for the right task

Outlook

- Implement interrupt-based communication with the accelerator
- Measuring collocation performance with different CPU workloads

→ Hypothesis: Is it possible to have a generic *optimal performance* policy for co-scheduling accelerator and CPU workloads with little parameterization effort?