Optimizing ATLAS data storage: the impact of compression algorithms on ATLAS physics analysis data formats

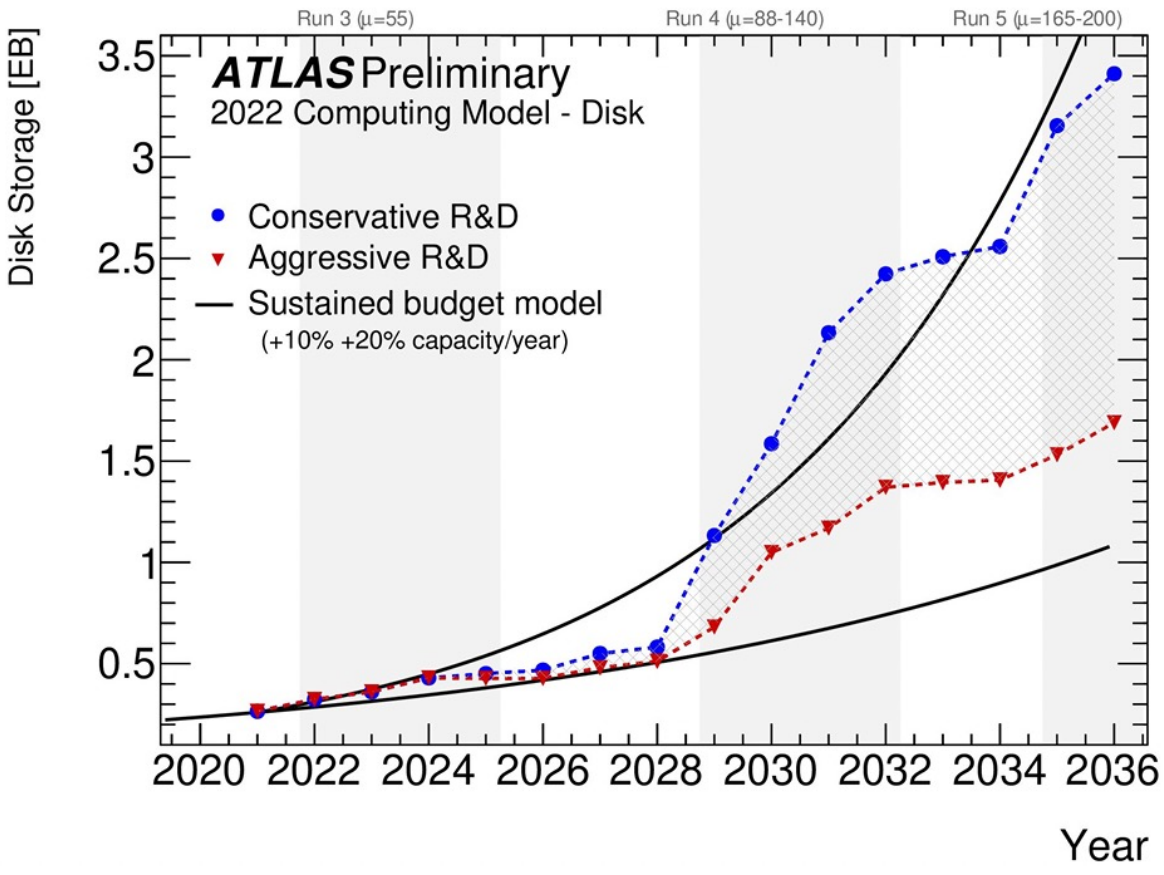
International Conference on Computing in High Energy and Nuclear Physics

May 8 – 12, 2023 - Norfolk Waterside Marriott

Caterina Marcon, Leonardo Carminati, Peter Van Gemmeren, Alaettin Serhan Mete

On behalf of the ATLAS Collaboration

Motivation



In the coming runs, the LHC accelerator will provide higher luminosity of particle collisions to the ATLAS experiment:

- more **simultaneous collisions** per event;
- higher demand of **disk space**;
- processing of a **larger event rate**;

⬇

- **storage** will present a **significant problem** for HL-LHC computing.

At the lowest level, LHC data is managed using **ROOT data framework;**

The need for efficient **lossless data compression** has grown significantly;

Interest in profiling the **compression algorithms provided by ROOT.**

ROOT Compression algorithms

- ROOT provides **four different compression algorithms**:
    - Zlib;
    - Lzma;
    - Lz4;
    - Zstd.

- All these algorithms can be tuned **via the compression level** option ranging **from 1 to 9**;

- Higher compression levels offer stronger compression;

- All the algorithms apply **lossless compressions** → no validation is needed;

- ROOT also provides different mechanisms to control how data are written to ROOT files (e.g. **AutoFlush** and **SplitLevel**).

Methods

- ATLAS events are stored in ROOT-based reconstruction output files (AOD) which are then processed within the derivation framework to produce Derived AOD files (DAOD);

- ATLAS **has changed its Analysis Model** which aims to reduce the disk footprint of centrally produced data products used for analysis;

- **Two new formats** have been proposed as a replacement for DAOD:
  - **(Run 3) DAOD_PHYS** (~50 kB/event) → containing all the variables needed to apply calibrations to reco objects;
  - **(Run 4) DAOD_PHYSLITE** (~10 kB/event) → containing precalibrated observables (see also [1]).

- Being ROOT-based formats, they **natively support the** aforementioned **lossless compression algorithms;**

- In ATLAS, **performance tests** of lossless compression algorithms are performed routinely when new ROOT features, new data products or major framework changes are available;

- This work is the first in-depth analysis on **DAOD_PHYS and DAOD_PHYSLITE formats**.

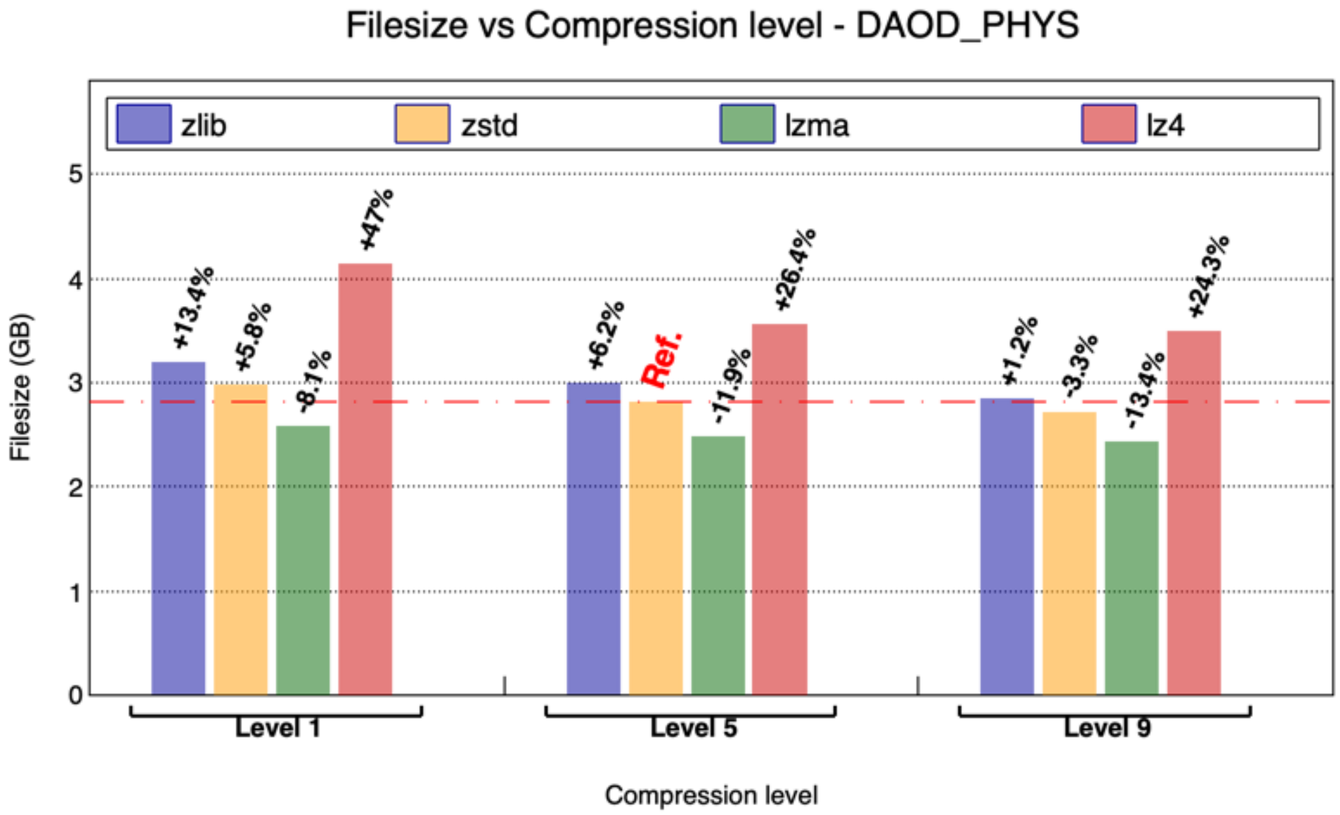
[1] https://indico.jlab.org/event/459/contributions/11586/

caterina.marcon@mi.infn.it

Methods

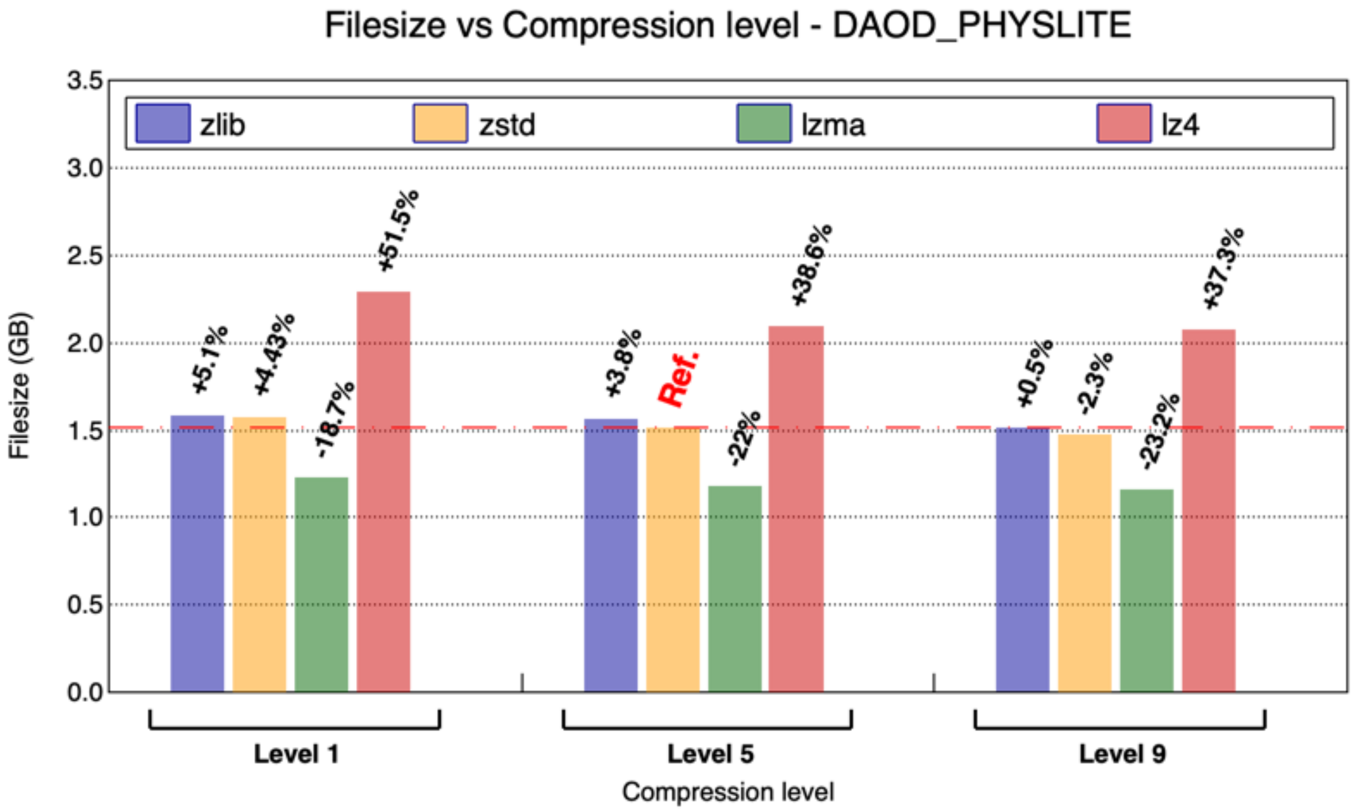
- Files compressed with a **minimal Athena tool**;
- Disk-based reading tests allow collection of I/O performance metrics;
- I/O performance metrics are collected via **PerfStats** (tool provided by ROOT → access to a range of performance statistics from within the process) and **dstat**;
- Reading tests emulate the typical ATLAS data access by reading events from the **TTree object** accounting for ~90% of the total file size;
- For each test, a subset of **20k events** has been read; and, for each event, **50% of the variables**;
- Each test was rerun **5 times** and **standard deviations are below 3%** in all cases;
- For file access, a lightweight analysis framework is used;
- All tests are carried out using **ROOT 6.24**, on a dedicated standalone machine.

File size vs Compression Level DAOD_PHYS



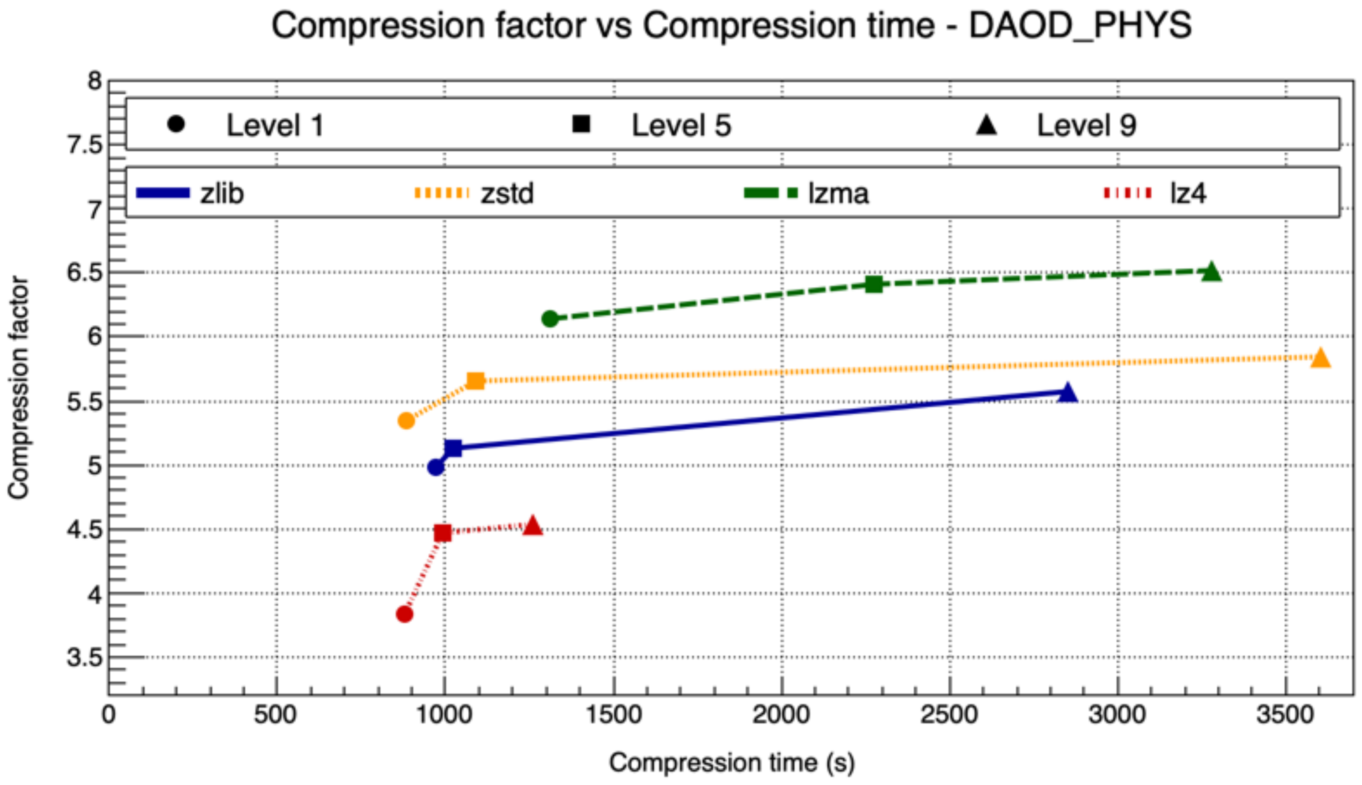
- The original file:
  - ttbar sample;
  - 15.92 GB;
  - AutoFlush: 500;
- The **zstd level 5** configuration has been considered as the **reference** performance;
- **Lzma provides the best compression** (with reductions of about 10%);
- **Lz4 results in the largest files** (with increases of up to ~ 45%);
- The file size depends primarily on the compression algorithm and not on the compression level.

File size vs Compression Level DAOD_PHYSLITE



- The original file:
  - ttbar sample;
  - 12.46 GB;
  - AutoFlush: 1000.
- The **zstd level 5** configuration has been considered as the **reference** performance;
- **Lzma provides the best compression** (with reductions of about 20%);
- **Lz4 results in the largest files** (with increases up to ~50%);
- The file size depends primarily on the compression algorithm and not on the compression level.

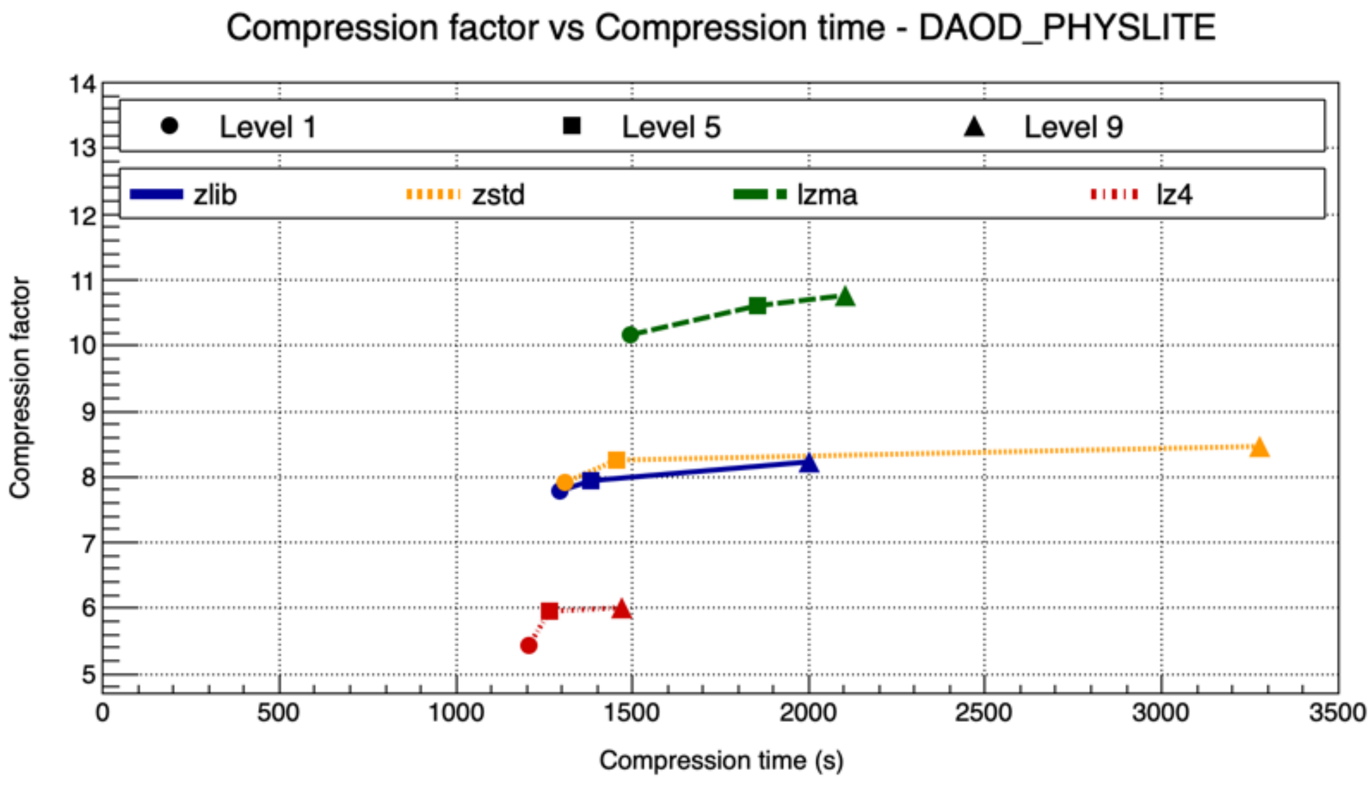
Compression Factor vs Compression time DAOD_PHYS



[1] Compression factor = uncompressed data/compressed data

- **Compression time** is the **total walltime** of the compression process**;**
- A **small compression time** with **a large compression factor** [1] **would be** the ideal configuration;
- **Lz4 provides fast compression** times but suffers from low compression factors;
- **Lzma achieves high compression** factors but compression times are slow;
- For Lzma, Lz4 and Zstd, the gain of compression level 9 flattens out → only relevant for cases where file size reduction is the most important metric.

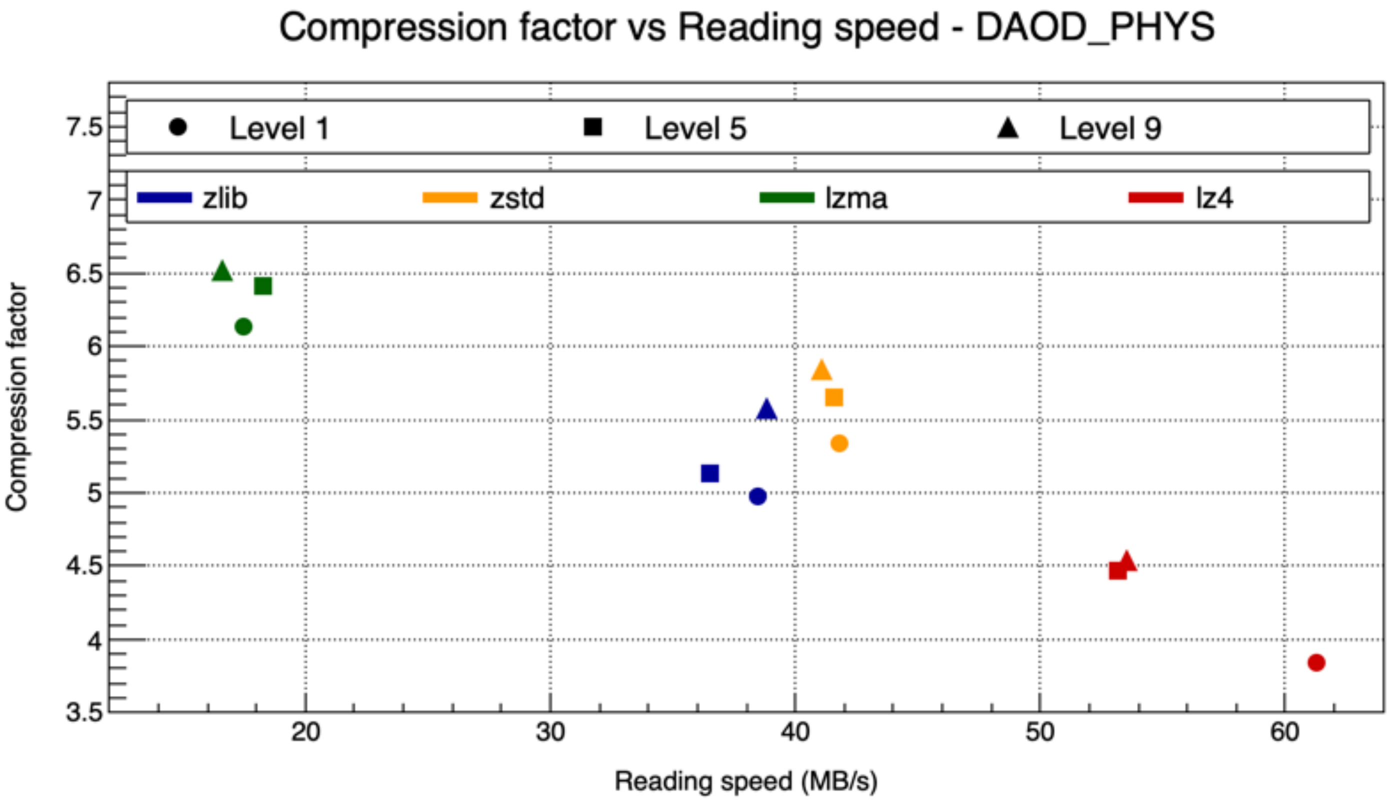
Compression Factor vs Compression time DAOD_PHYSLITYE



- **Compression time** is the **total walltime** of the compression process**;**
- A **small compression time** with **a large compression factor** [1] is the ideal configuration;
- **Lz4 provides fast compression times** but suffers from low compression factors;
- **Lzma achieves high compression factors** but compression times are slow;
- For Lzma, Lz4 and Zstd, the gain of compression level 9 flattens out → only relevant for cases where file size reduction is the most important metric.

[1] Compression factor = uncompressed data/compressed data

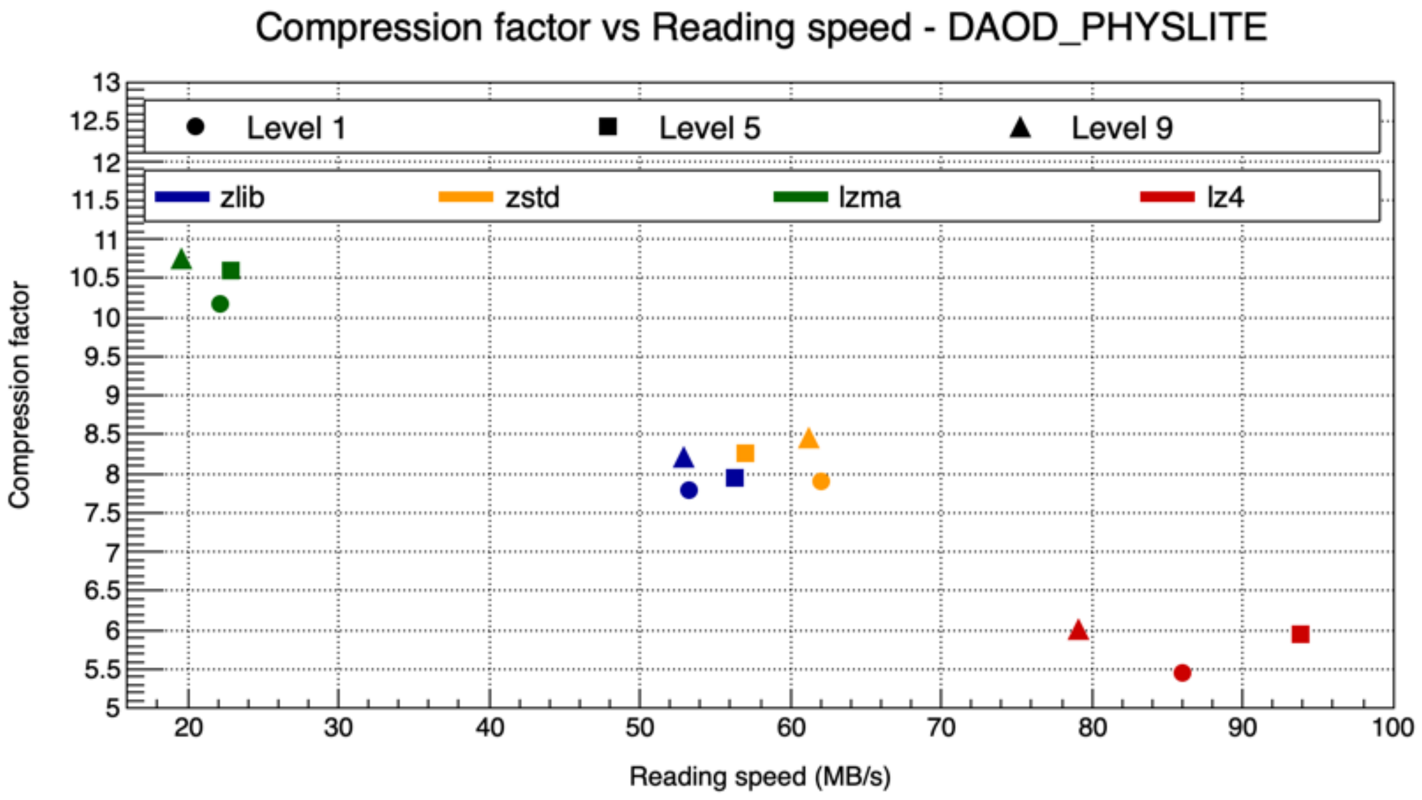
Compression Factor vs Reading speed DAOD_PHYS



- **Reading speed = (bytes read) / (process time)** where process time is the time spent processing events;
- **A large reading speed** with **a large compression factor would be** the ideal configuration;
- **Lzma has a low reading speed**;
- **Lz4 is the fastest in reading**;
- The reading speed depends primarily on the compression algorithm and not on the compression level.

[1] Compression factor = uncompressed data/compressed data

Compression Factor vs Reading speed DAOD_PHYSLITYE

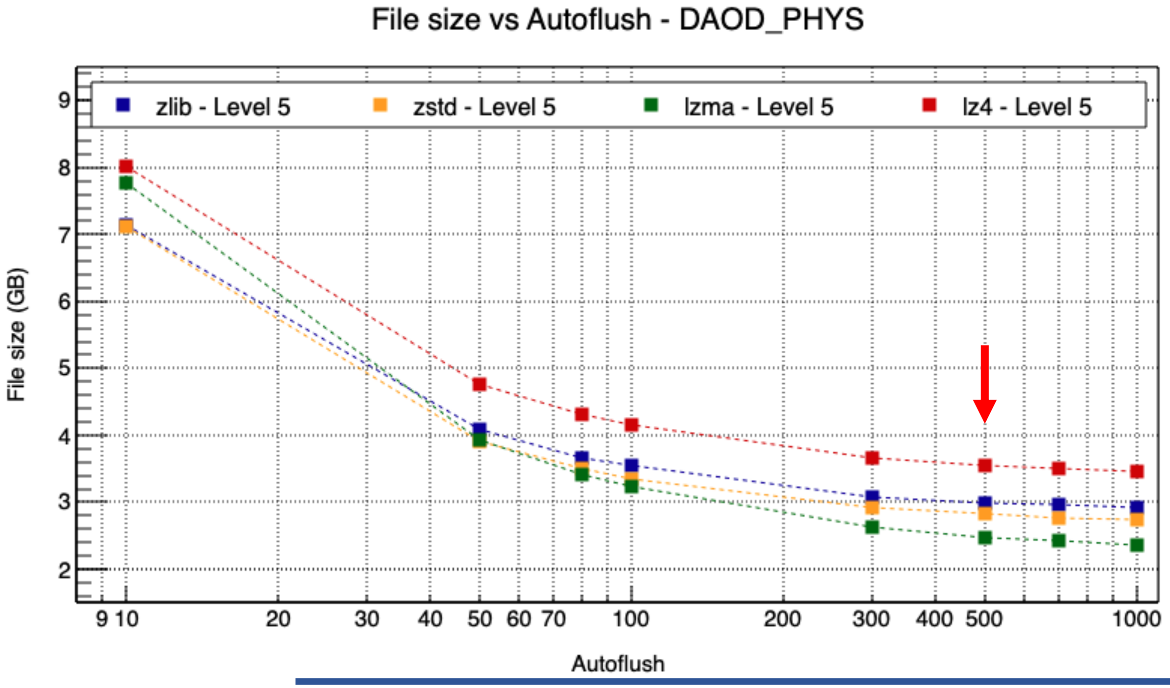


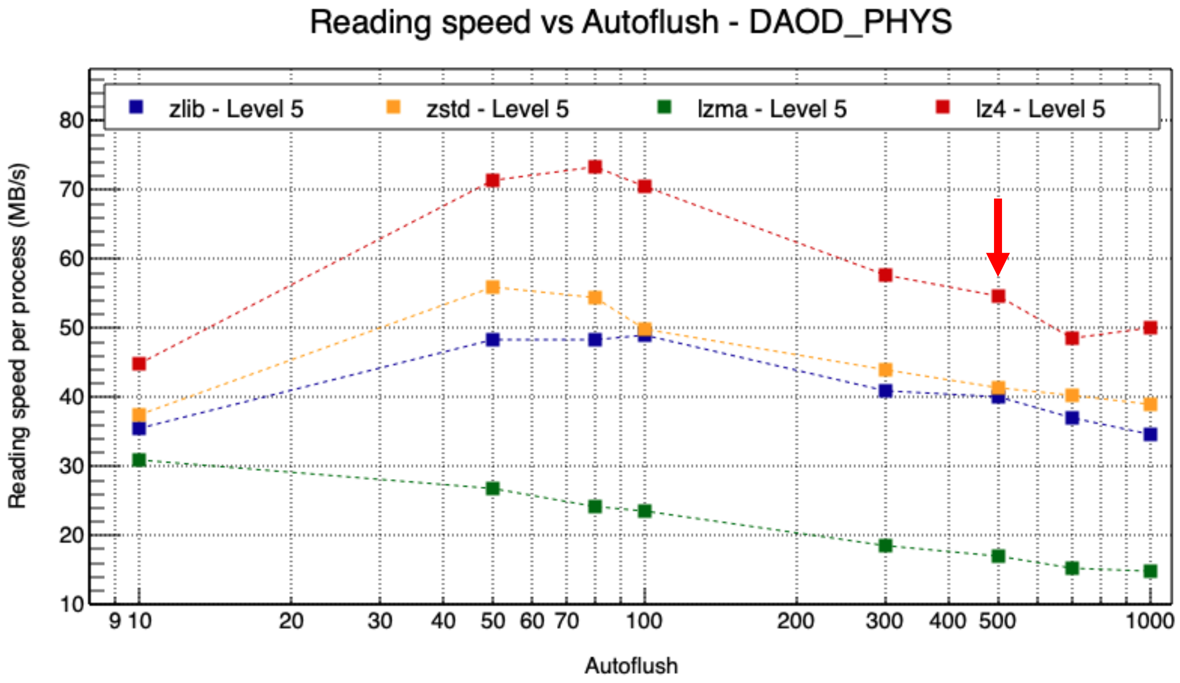
[1] Compression factor = uncompressed data/compressed data

- **Reading speed = (bytes read) / (process time)** where process time is the time spent processing events;
- **A large reading speed** with **a large compression factor would be** the ideal configuration;
- **Lzma has a low reading speed**;
- **Lz4 is the fastest in reading**;
- The reading speed depends primarily on the compression algorithm;
- For lz4 the impact of the compression level is more significant.

Autoflush impact on DAOD_PHYS

- AutoFlush specifies how large a single compression unit of a TTree is in terms of number of events;
- The **original AutoFlush value of the file is 500**;
- Tests are carried out for all the compression algorithms setting the compression level to 5.

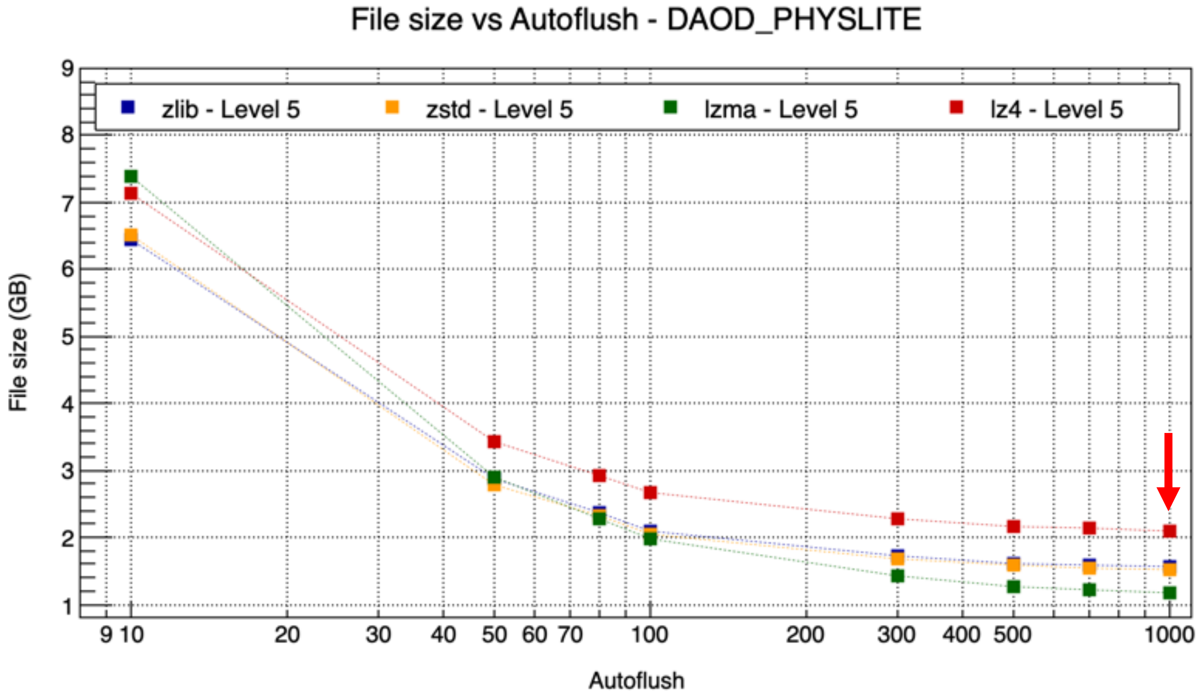


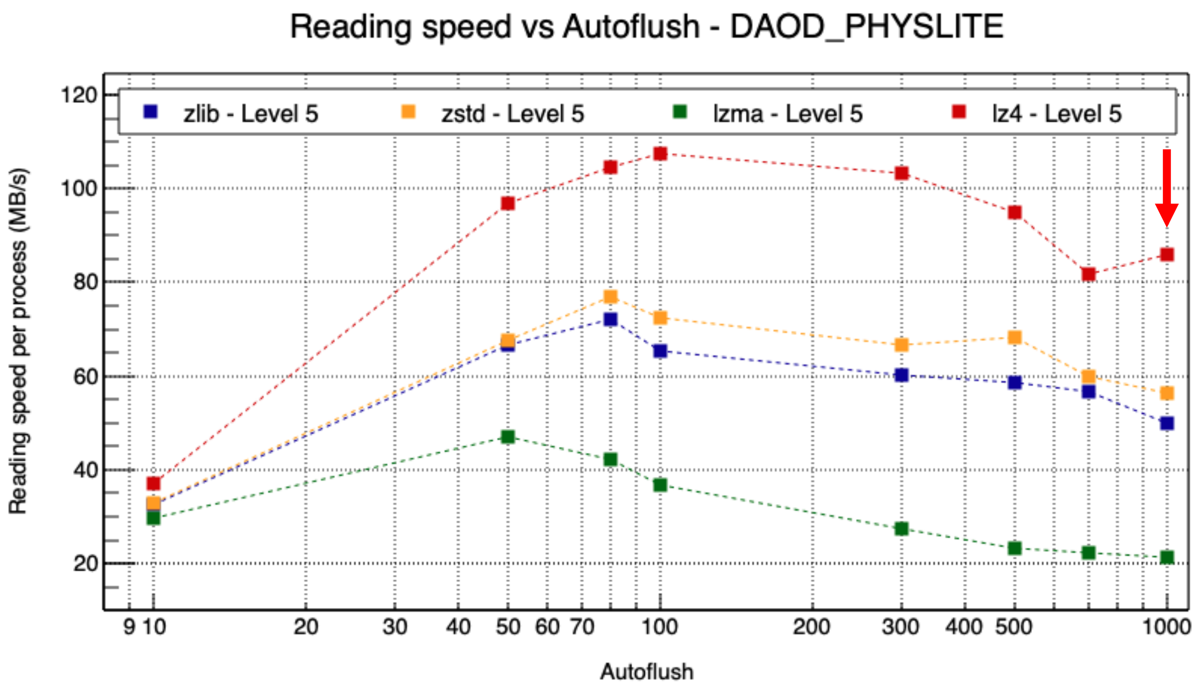


- Compression algorithms are more efficient with more data to compress;
- **The original AutoFlush value (500) is reasonable**: it shows a good performance both in terms of file size and reading speed.

Autoflush impact on DAOD_PHYSLITE

- AutoFlush specifies how large a single compression unit of a TTree is in terms of number of events;
- **The original AutoFlush value of the file is 1000;**
- Tests are carried out for all the compression algorithms setting the compression level to 5.





- Compression algorithms are more efficient with more data to compress;
- The original AutoFlush value (1000) is reasonable; although **AutoFlush = 500 shows a slightly better performance** in terms of reading speed.

Future steps & Conclusions

- Rerun **partial event reading tests** for different event and variable ratios (ongoing);
- Add **memory profiling** to the test suite (ongoing).

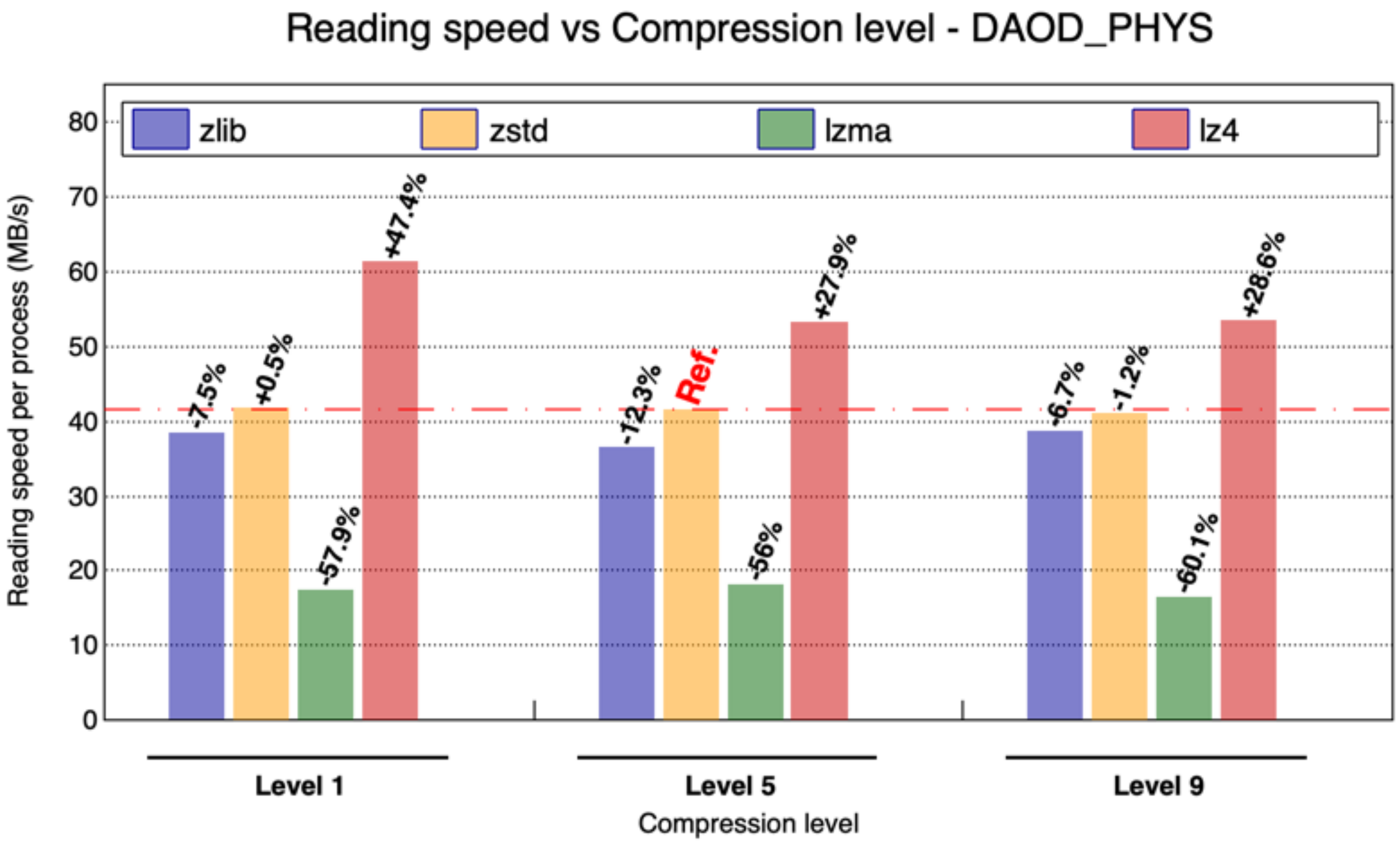
- For both types of derived files, **Lz4 is the fastest in reading but results in the largest files**: it should be considered when fast reading is more important than file size reduction;
- In both cases, **Lzma provides higher compressions at the cost of significantly slower reading speeds**: it should be considered when file size reduction is the key parameter;
- For both types of derived files, **AutoFlush = 500 could be considered a good compromise** considering both file size and reading performances.

Backup

Computing resources

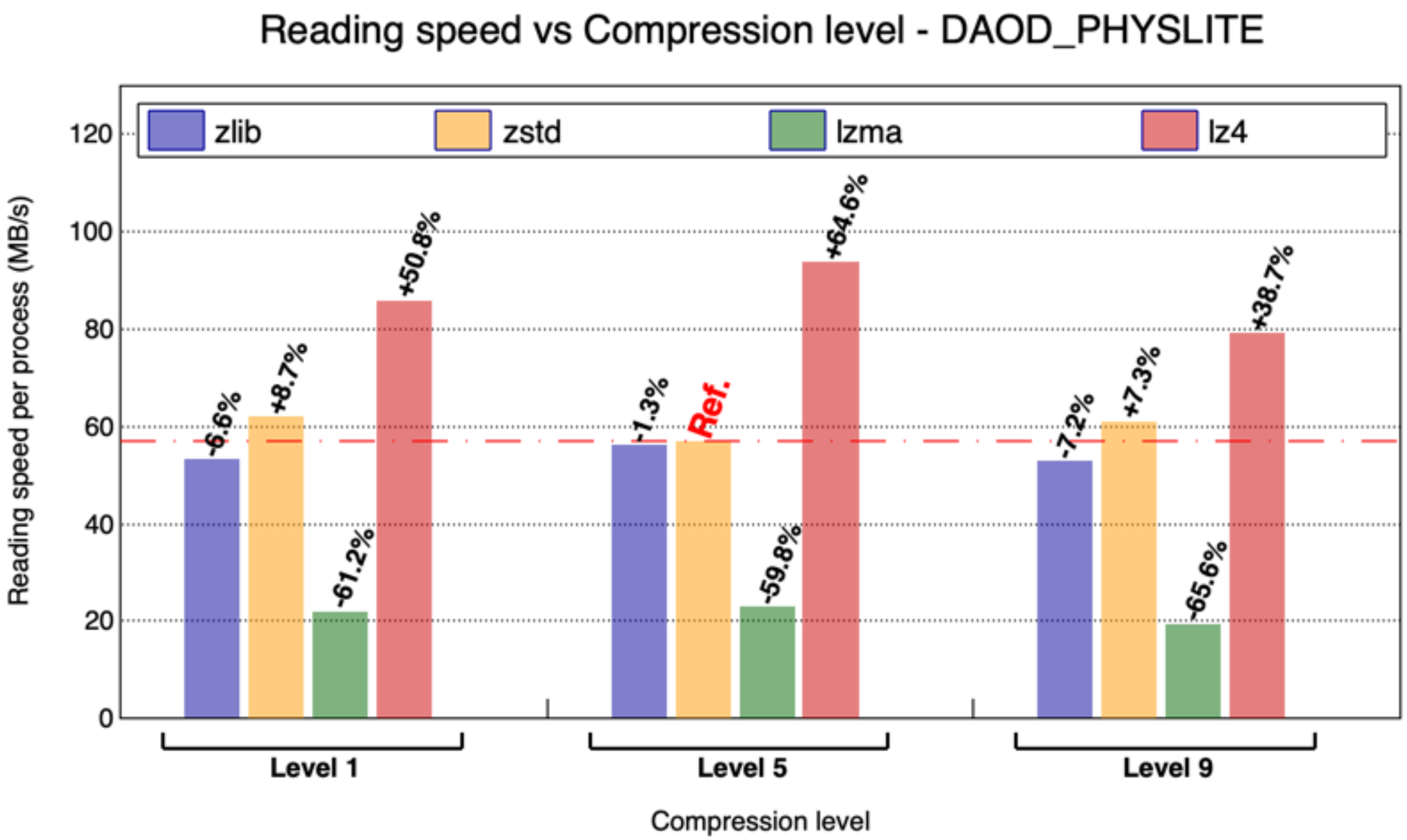
- CPU: 2x AMD EPYC 7302 (16 Core, 32 Thread)
- 256 GB RAM
- 1.92 TB NVMe SSD (Read: 3000 MB/s, Write: 1500 MB/s)
- CentOS 7

Reading speed VS Compression Level DAOD_PHYS



- The **zstd level 5** configuration has been taken as **reference**;
- **Lzma has a low reading speed** (with degradations of more than 55%);
- **Lz4 is the fastest in reading** (with a ~40% improvement);
- The reading speed depends primarily on the compression algorithm and not on the compression level.

Reading speed VS Compression Level DAOD_PHYSLITE



- The **zstd level 5** configuration has been taken as reference.
- **Lzma has low reading speed** (with degradations of more than 60%);
- **Lz4 is fastest in reading** (with ~ 40% improvements);
- The reading speed depends primarily on the compression algorithm and not on the compression level.