# A Hot Take on the Intel Analytics Accelerator for Database Management Systems

**Christos Laspias**, Andy Pavlo, Jignesh M. Patel

# Overview

- Motivation
- Contribution
- IAA Overview
- Integration with DuckDB
- Evaluation
- Conclusion

# Motivation

# Why Hardware Acceleration for DBMS?

- Offload repetitive tasks to accelerator, free CPU for other work
- Accelerators often provide :
  - Lower Latency
  - Higher Bandwidth
- Problems:
  - Data movement
  - Interaction with accelerator

# Why Intel IAA?

- IAA resides on the CPU's die; no data movement required
- User-space API, no hardware programming (e.g., FPGAs)

# Why Intel IAA for OLAP?

- Columnar formats (e.g., Parquet) rely on compression
  - Less I/O, fewer data transfers
- Big % of cycles spent decompressing data
  - In TPC-H with Parquet & Snappy DuckDB  spends ~40% of cycles
- IAA improves latency for OLAP workloads through faster decompression

# Contribution

# Contribution

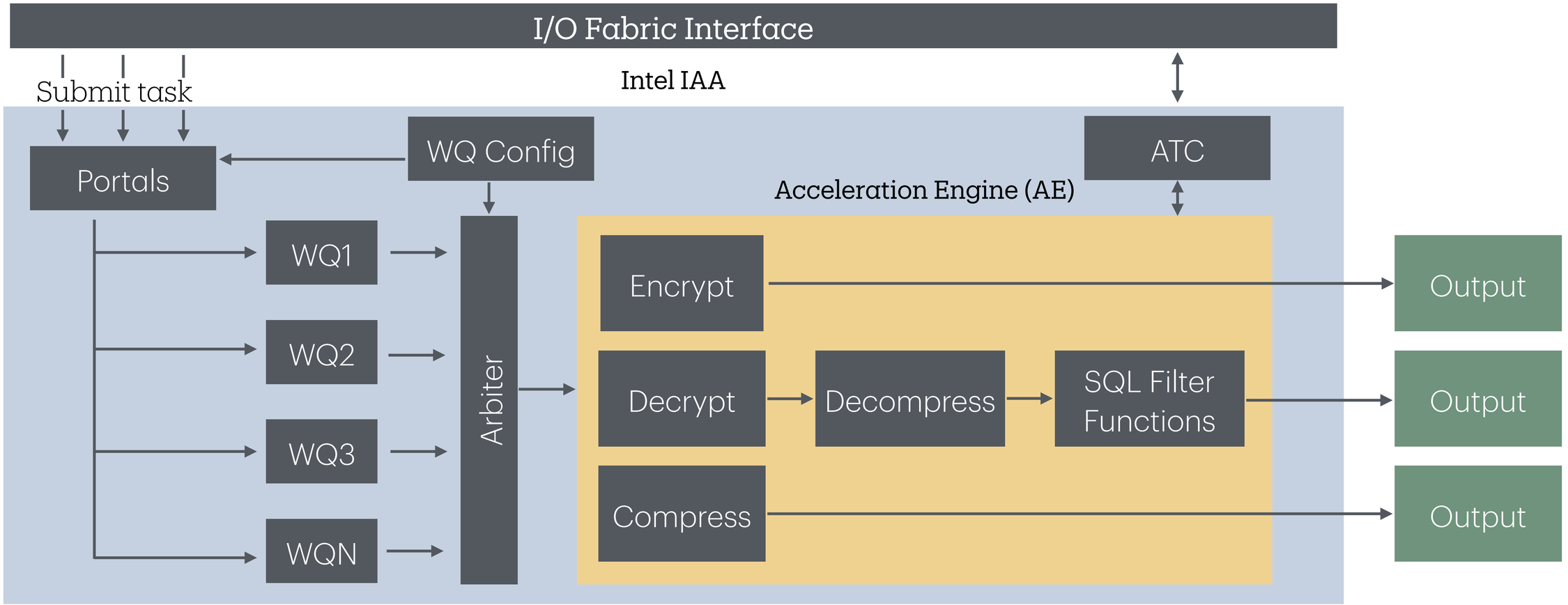
- Exploring IAA. Is IAA useful for database systems?
- Focus on IAA's (de)compression & Integrate IAA in DuckDB
- Up to 3.5x faster decompression in microbenchmarks
- Up to 38% faster TPC-H queries in DuckDB

# IAA Overview

# Intel IAA Overview

- On-chip accelerator designed to improve the performance of OLAP engines
- Shared Vitual Memory; no data movement over PCIe
- Cache coherent memory access
- Task submission using memory-mapped I/O (MOVDIRI, ENQCMD, etc.)
- Intel Query Processing Library (QPL)

# Intel IAA Hardware Structure






Intel Accelerators Ecosystem: An SoC-Oriented Perspective, ISCA 24'

# Integration with DuckDB

# Integration with DuckDB

```
1. void ColumnReader::DecompressInternal(CompressionCodec::type codec,
2.                                       const_data_ptr_t src, idx_t src_size,
3.                                       data_ptr_t dst, idx_t dst_size) {
4.  switch (codec) {
5.          case CompressionCodec::UNCOMPRESSED:
6.              ...
7.          case CompressionCodec::SNAPPY:
8.              ...
9.          case CompressionCodec::IAA: {
10.             QplDecompress(src, src_size, dst, dst_size);
11.             break;
12.          }
13.     }
14.}
```

***New Code***

# Task Submission (Compression)

```
1. vector<uint8_t> source(uncompressed_size); //Uncompressed buffer
2. vector<uint8_t> destination(compressed_size); //Compressed buffer
3.
4. job->op = qpl_op_compress; // Specify the compression task
5. job->level = qpl_default_level; // Specify compression level
6.
7. job->next_in_ptr = source.data(); // Uncompressed buffer pointer
8. job->available_in = source.size(); // Uncompressed size
9.
10.job->next_out_ptr = destination.data(); // Compressed buffer pointer
11.job->available_out = destination.size(); // Compressed size
12.
13.job->flags = QPL_FLAG_FIRST | \ /* flags for compression*/
14.    QPL_FLAG_LAST | \
15.    QPL_FLAG_DYNAMIC_HUFFMAN | \
16.    QPL_FLAG_OMIT_VERIFY;
17.
18.//Execute the task
19.qpl_execute_job(job);
```

# Evaluation

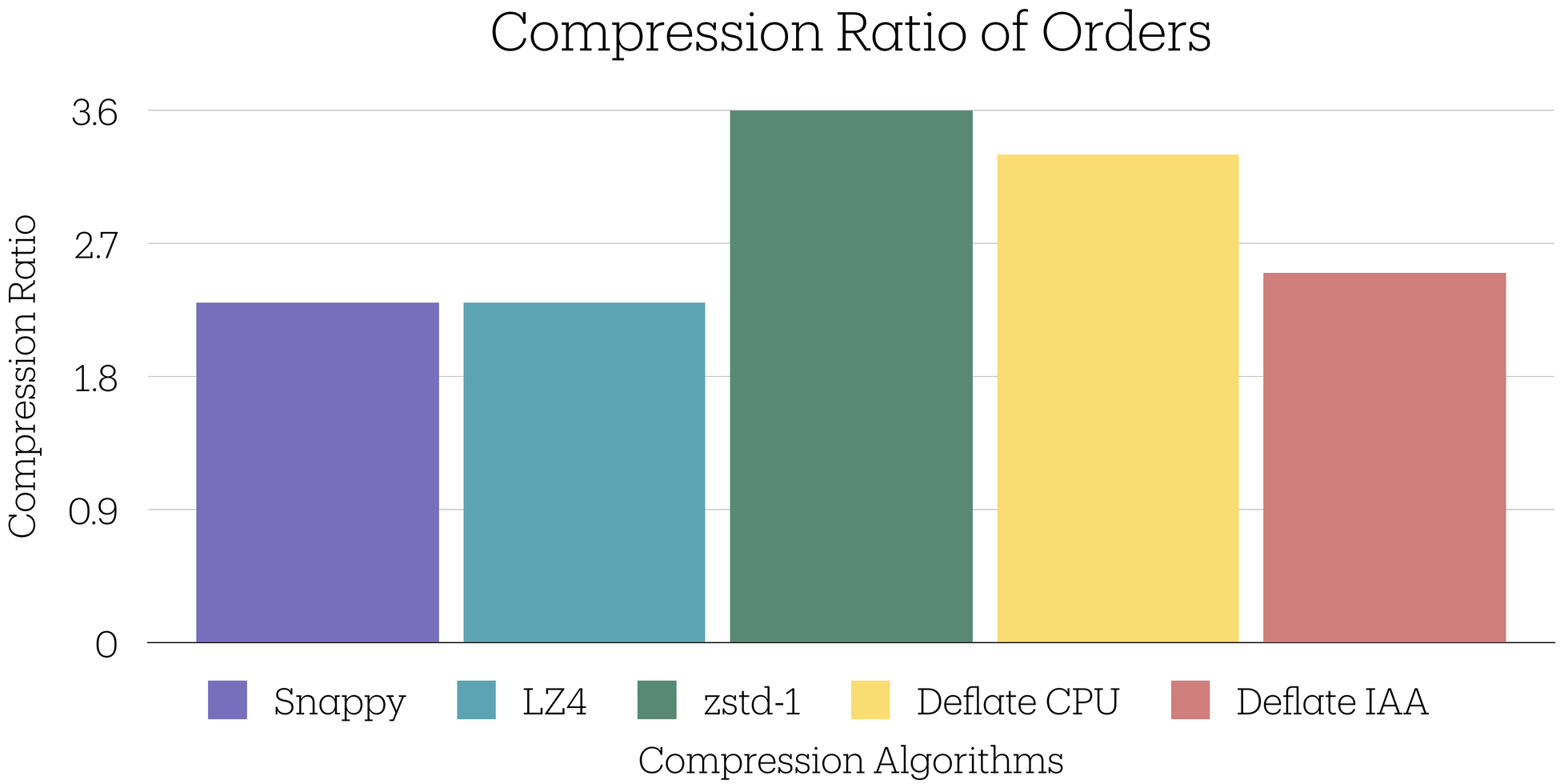
# Experimental Setup

- AWS m7i.metal-24xl
- 4th Gen Intel Xeon Sapphire Rapids Platinum 8488C (48 Cores, 96 Threads)
- 4 IAA Devices, 8 AEs each, 32 AEs in total
- DuckDB v1.2.2 compiled with GCC 13.3
- Microbenchmarks in C++

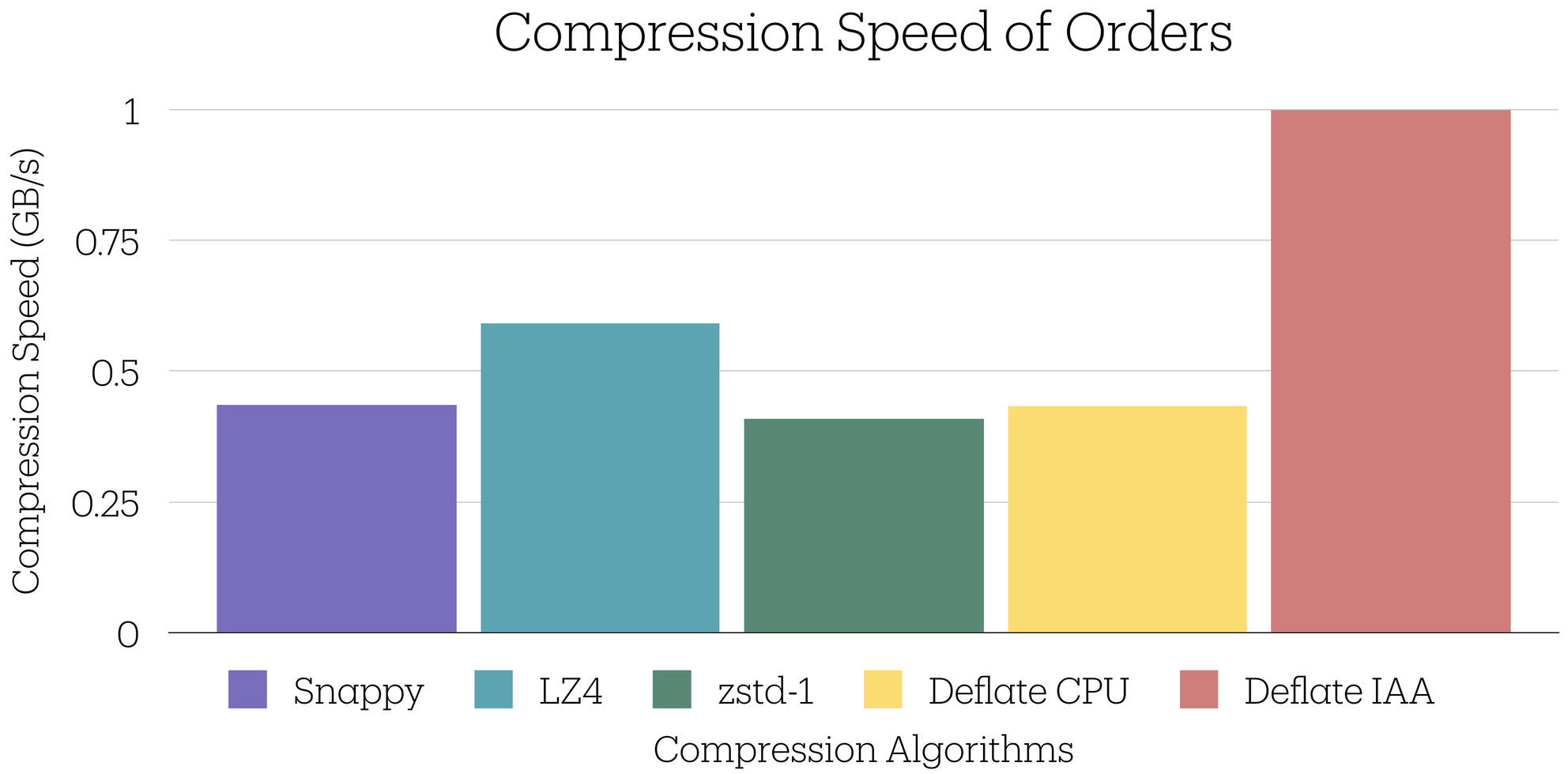
# Microbenchmarks

- Generate TPC-H tables with scale factor 10
- Write each table uncompressed in a Parquet file
- (De)/Compress the file

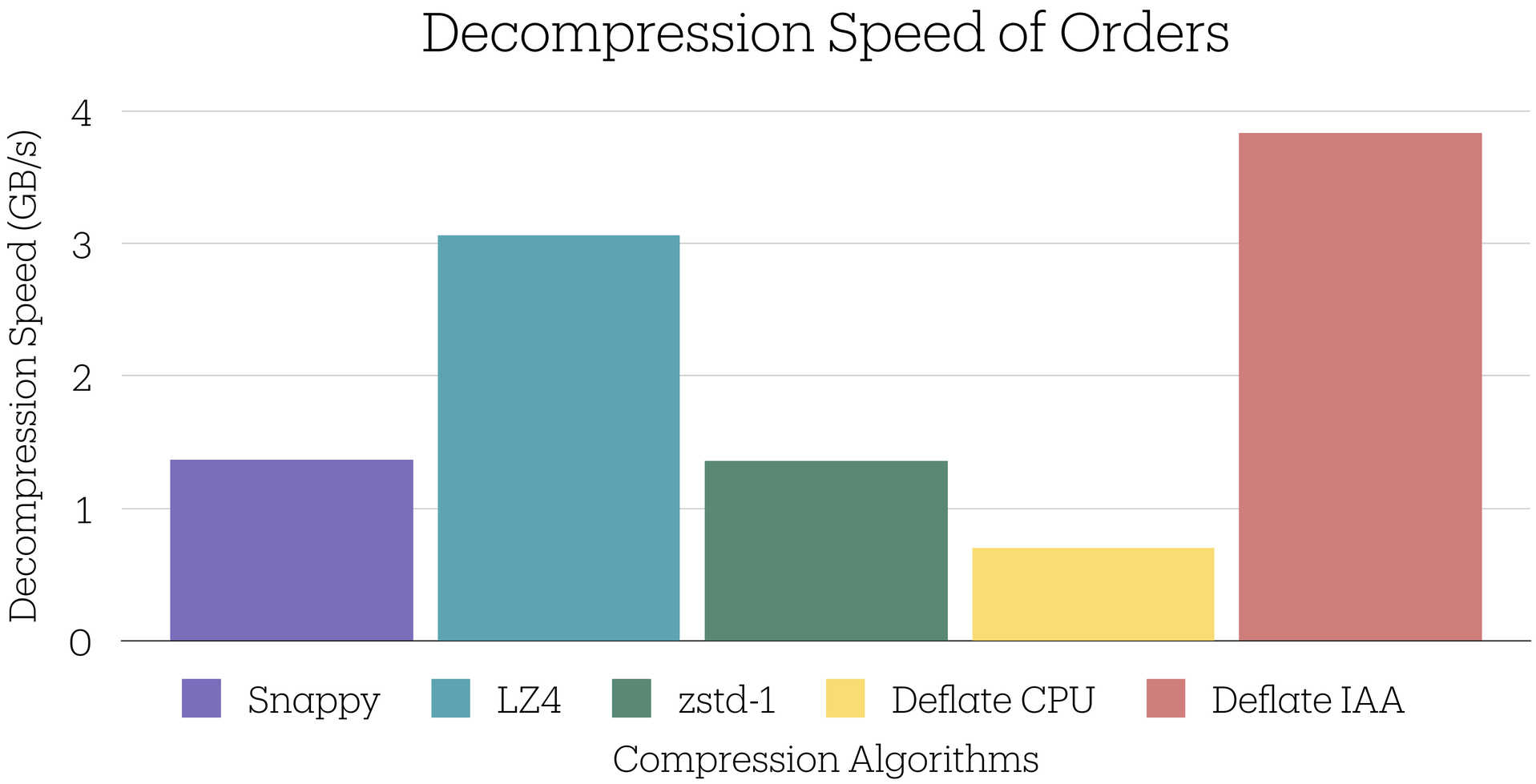
# Microbenchmarks



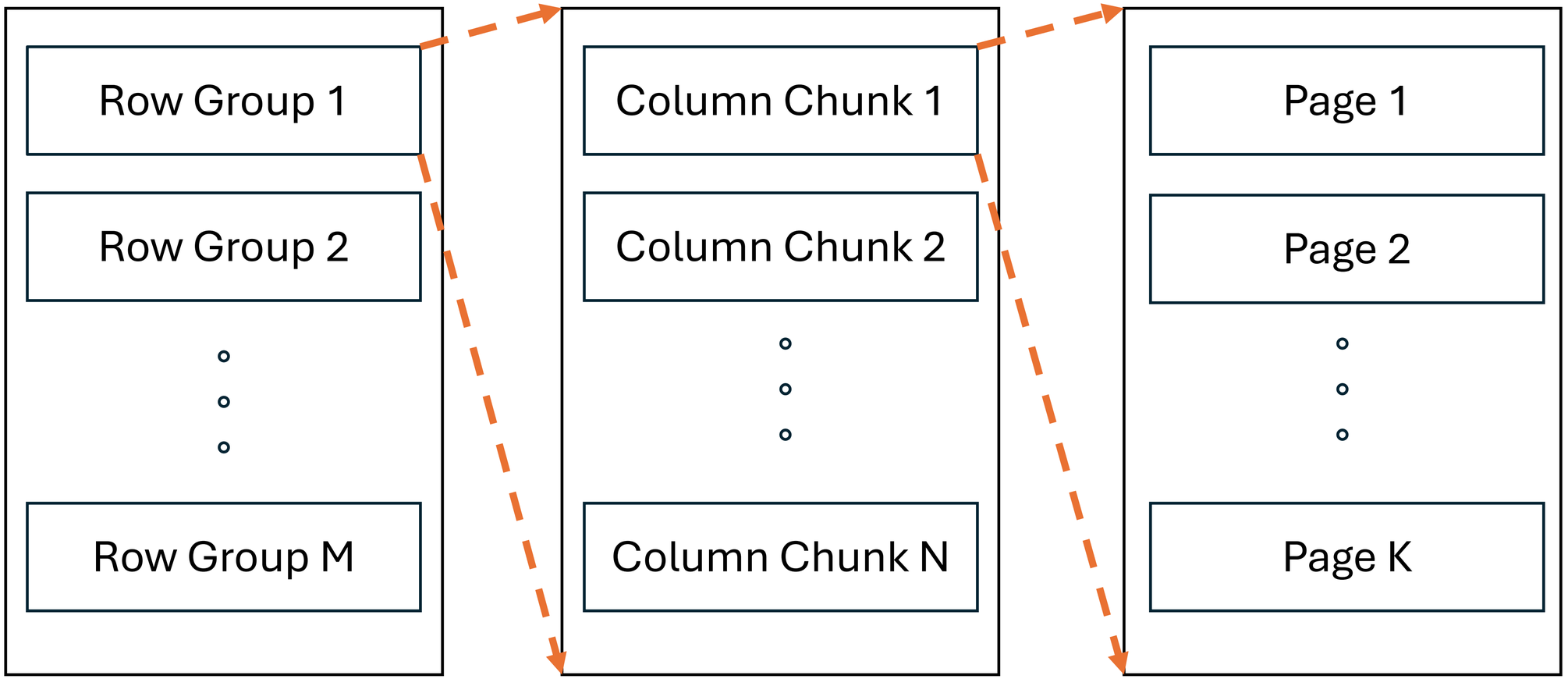
# Microbenchmarks



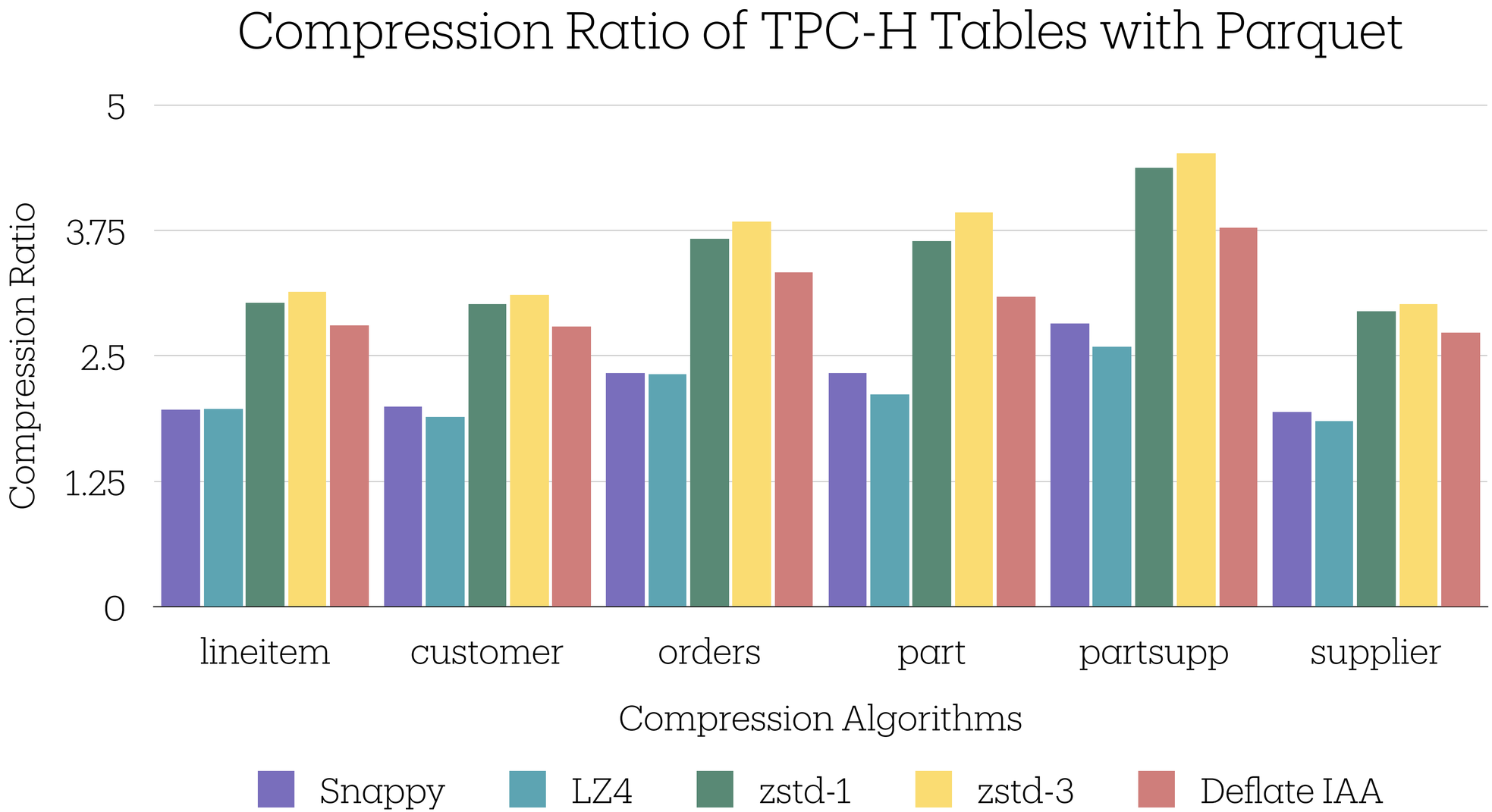
# Microbenchmarks



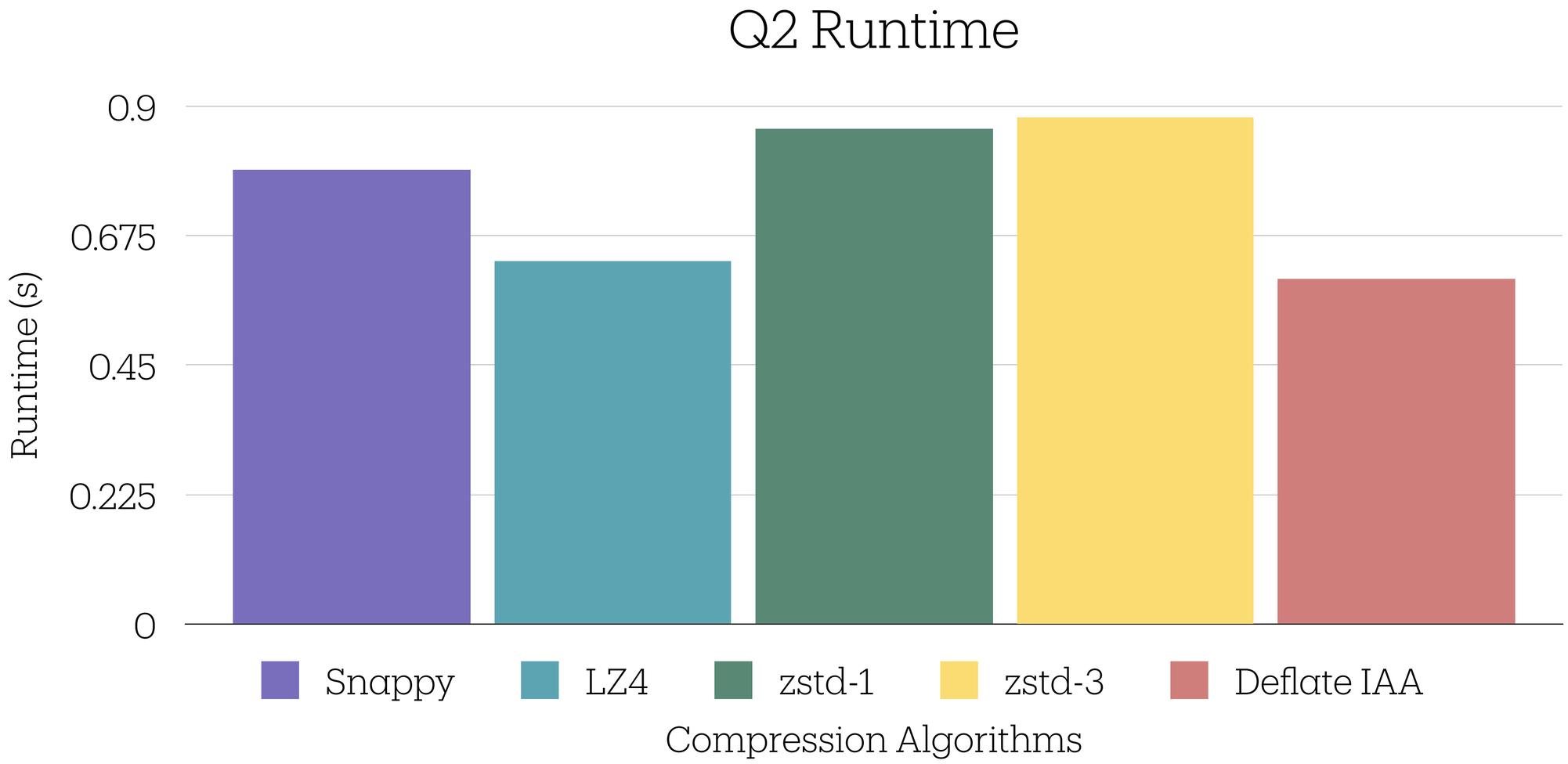
# Parquet


Row Group 1
Row Group 2
Row Group M
Column Chunk 1
Column Chunk 2
Column Chunk N
Page 1
Page 2
Page K

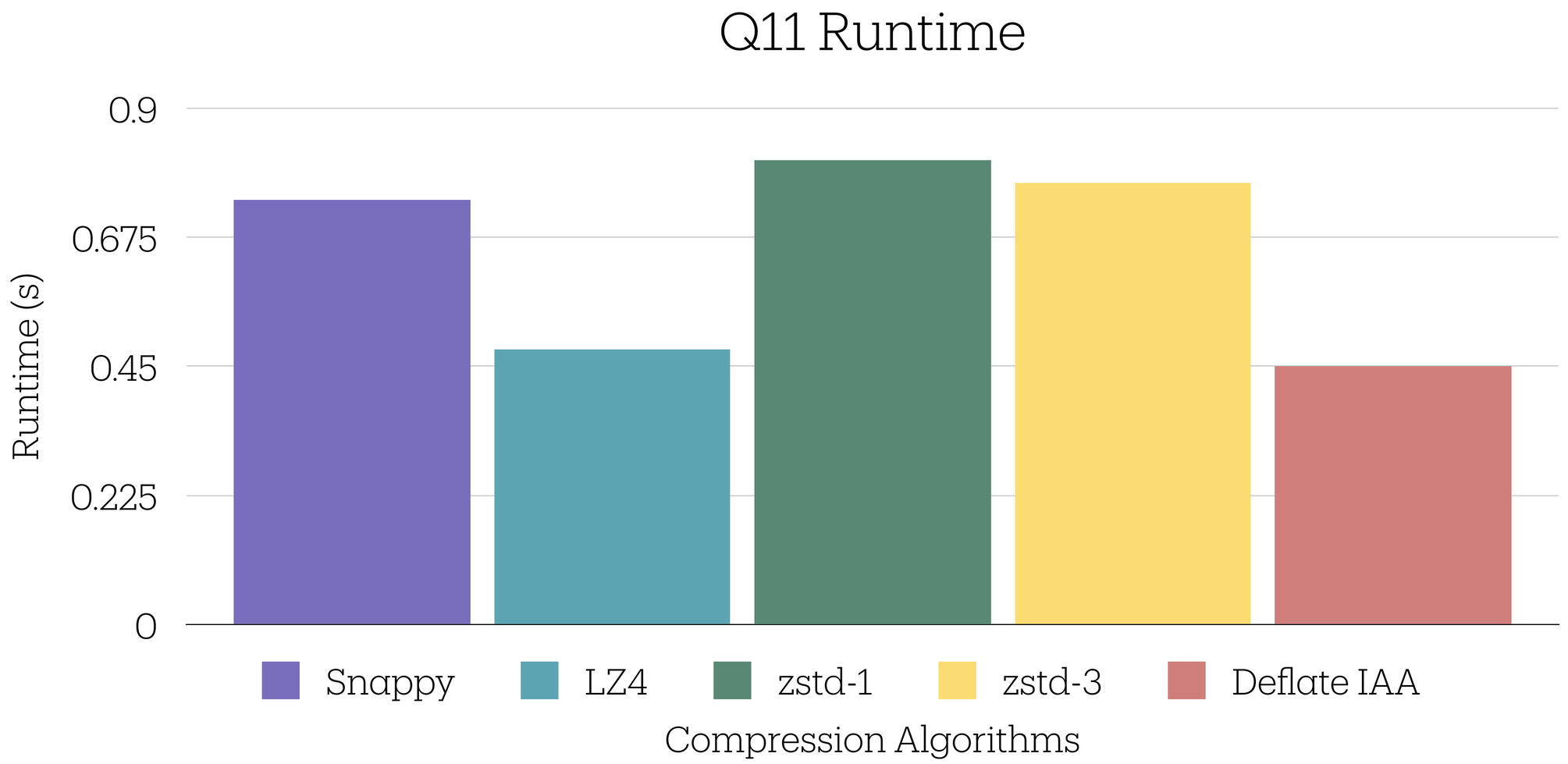
# Parquet Compression



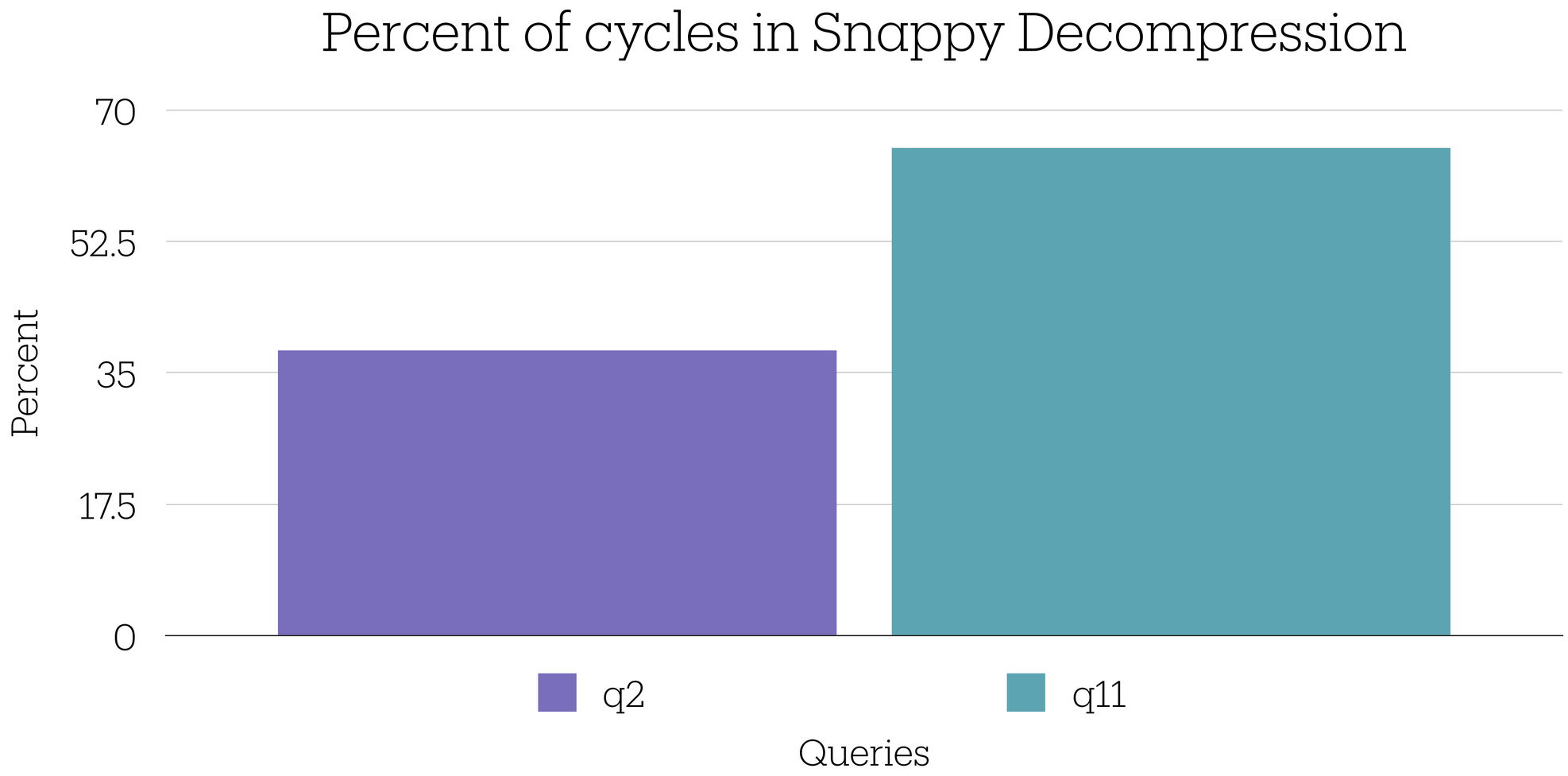
# End-to-End TPC-H



# End-to-End TPC-H



# Profiling TPC-H Queries



# Limitations of IAA

# Limitations of IAA

- Only supports a few compression algorithms
- Lack of programmability:
  - Compression algorithms
  - Hash functions
- Max Input Buffer 4GB

# Conclusion

# Conclusion

- Compression is promising
  - High decompression speed
  - Competitive compression ratios
  - Up to 38% faster TPC-H queries in DuckDB
- Big adoption potential if aligned with software standards

Thank you!

https://www.cs.cmu.edu/~claspias