TU Delft

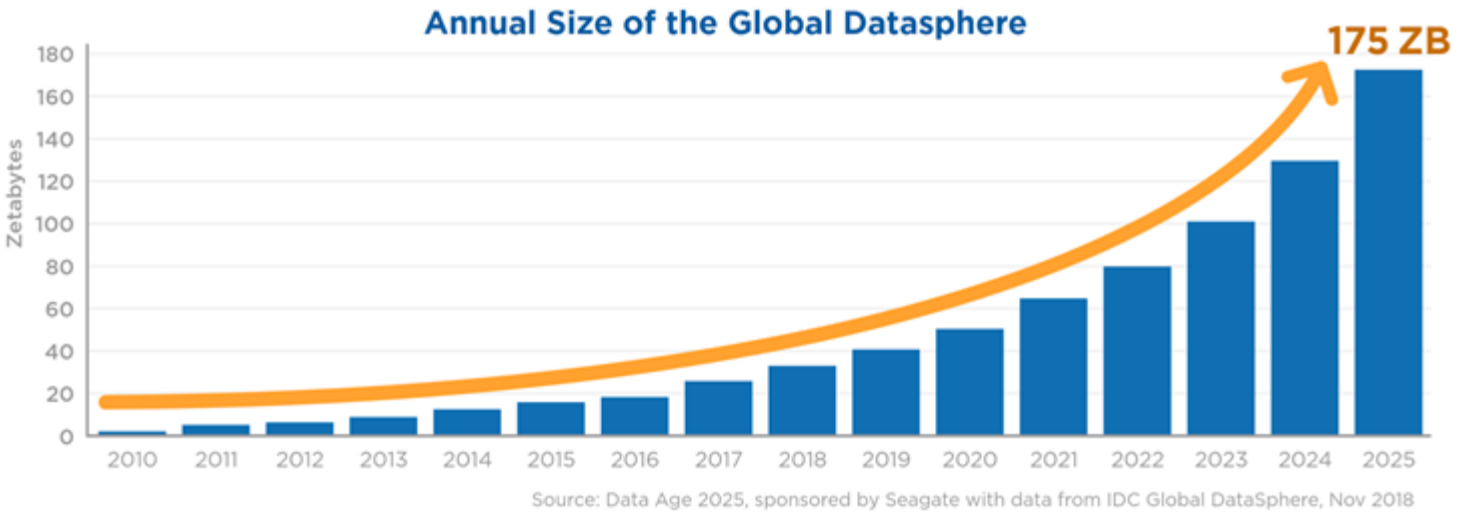
GSST

High Throughput Parallel String Decompression on GPU

Robin Vonk

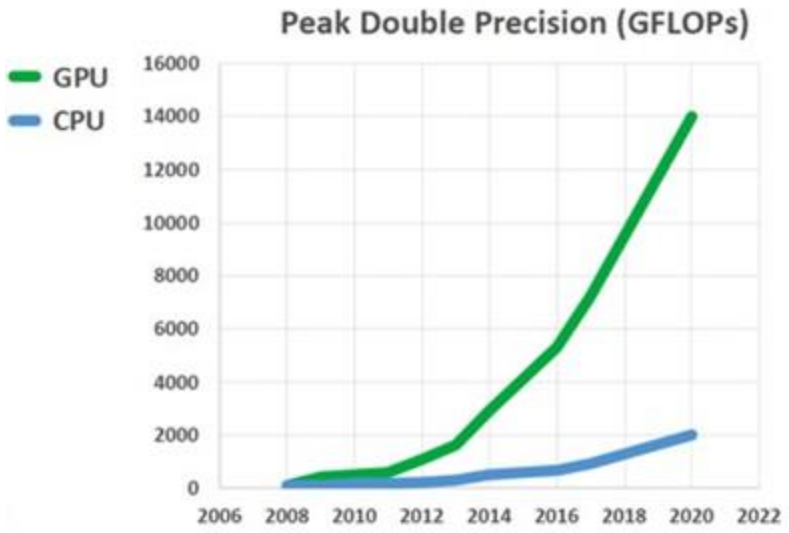
31-3-2025

Data Growth



- The volume of data doubles every 2 years
- The need for faster processing keeps growing

Graphics Processing Unit



Source: https://www.nextplatform.com/2019/07/10/a-decade-of-accelerated-computing-augurs-well-for-gpus/

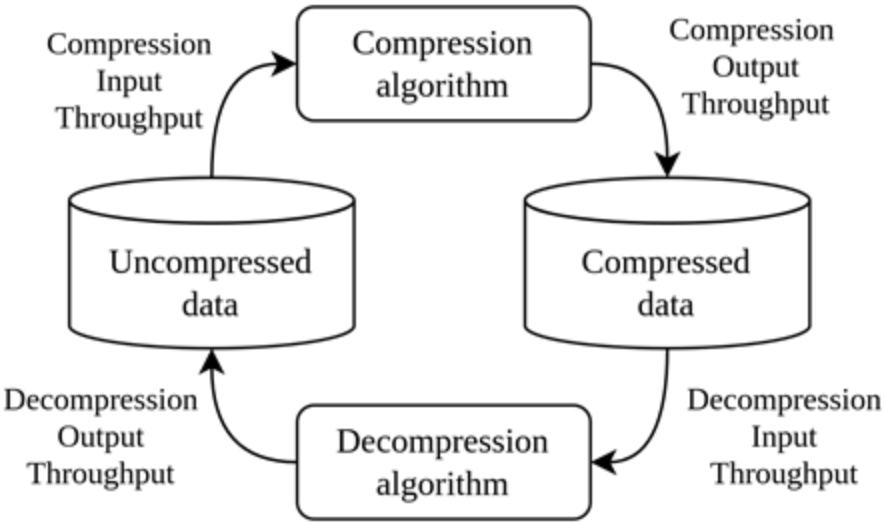
Source: https://lenovopress.lenovo.com/lp1734-thinksystem-nvidia-a100-pcie-40-gpu

3D Graphics → General Purpose

Lossless Compression

Compressed Data

Data with size reduced using compression techniques



Lossless

Data can be completely reconstructed

Compression Ratio

Compressed size compared to the uncompressed size

Throughput

Amount of data written per second by a compression algorithm

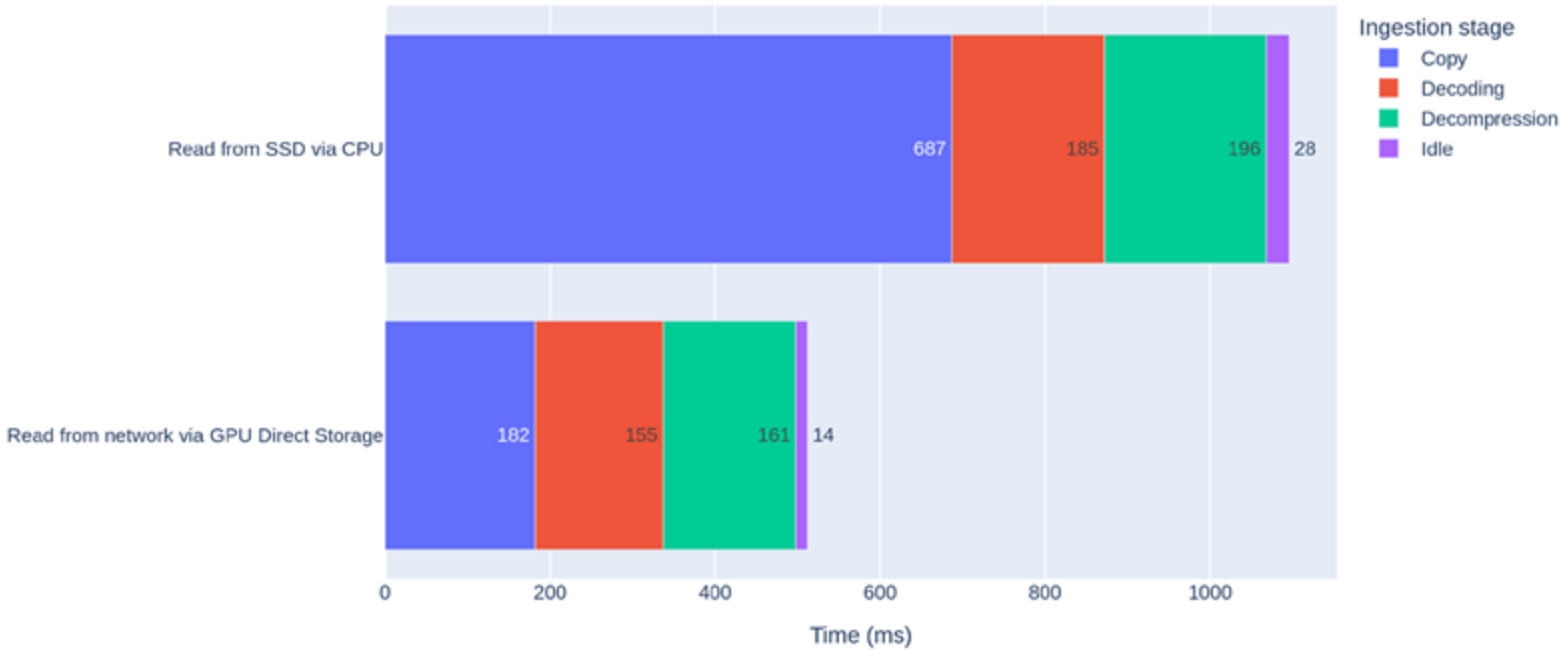
Data Ingestion

The process of preparing stored data for processing

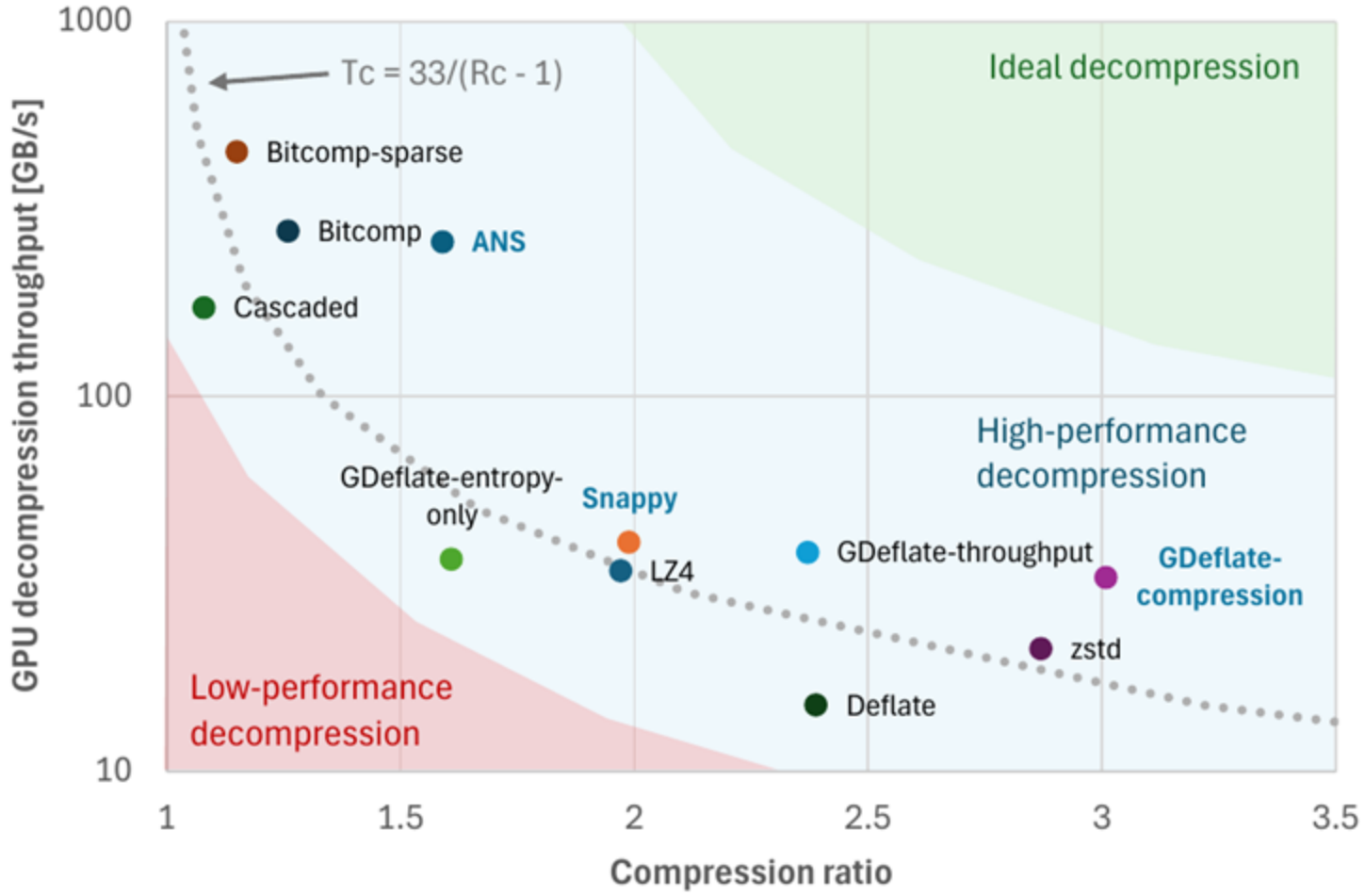
As data transfers are getting faster, decompression time becomes more significant

On fast network storages, decompression can take as long as the transfer itself

Storing data uncompressed can be faster than using slow decompression



GPU Lossless String Decompression



- Data ingest throughput depends on
  - Compression ratio
  - Decompression throughput

How to increase data ingestion throughput of string data on GPUs?

Topics

01

FSST Compression

02

GSST Decompression Format

03

Memory Access Optimizations

04

Results and Conclusions

01

FSST Compression

FSST Compression

Fast Static Symbol Table

Unmatched single threaded decompression throughput

Compression

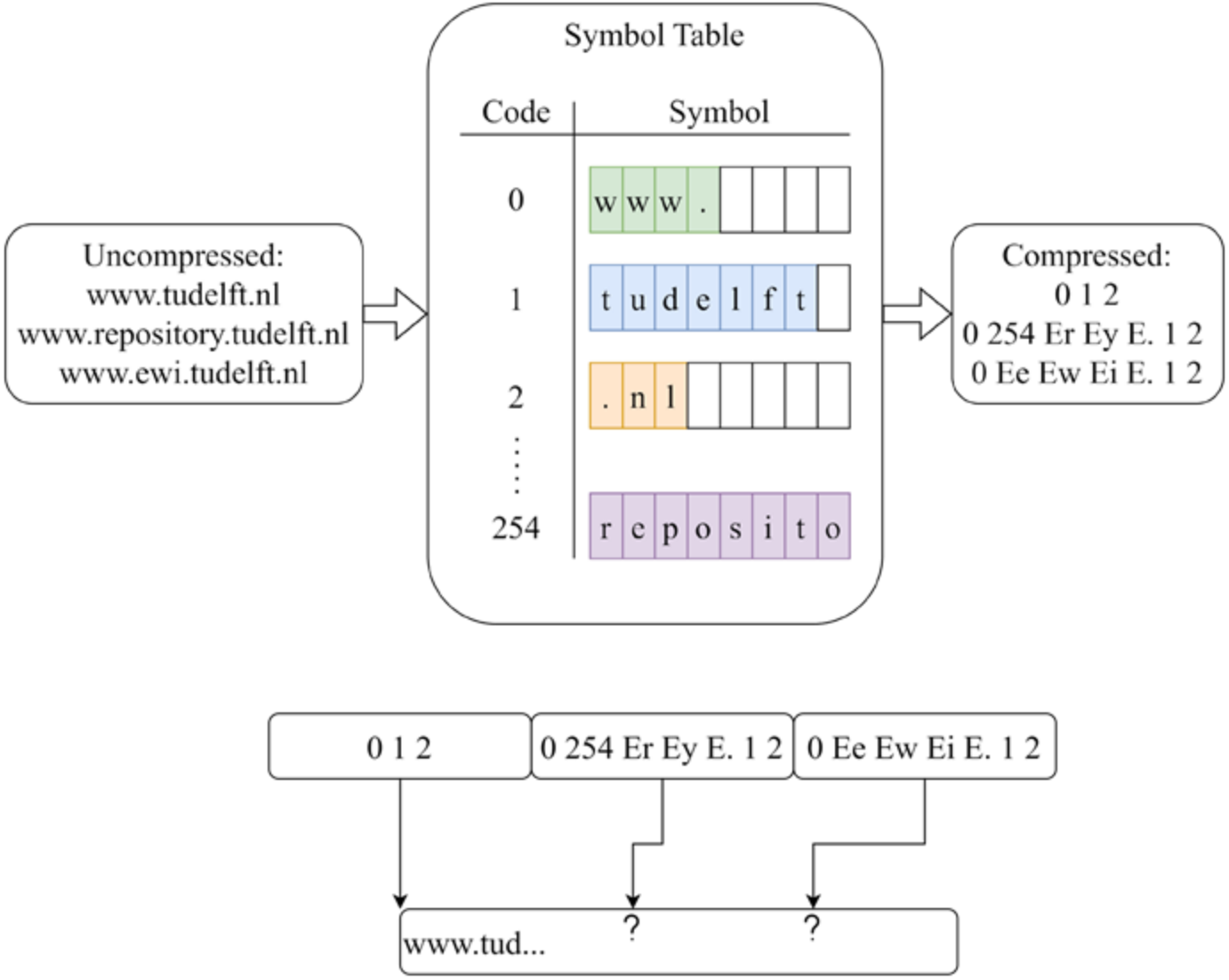
1. Scan input for repeated **symbols**
2. Replace symbols with **codes**

Decompression

1. Replace codes with symbols

Only one sequential problem:

Where to write the output data?



GPU Hardware

Parallelism

6912 Cores over 108 streaming multiprocessors

High Bandwidth Memory

2 TB/s memory throughput

Source: https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf

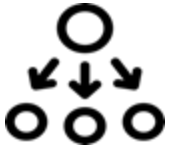
02

GSST Decompression Format

GSST Decompression

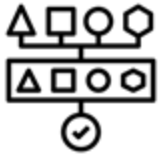
Block Parallelism Format

Decompressing blocks of data in parallel



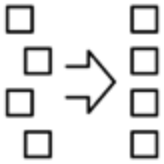
Split Parallelism Format

Decompressing a single block in parallel



Shared Memory

Buffering data close to the processor



Aligned Memory Transfers

Reducing data transfer overhead

Block Parallelism

Divide the input data over independent compressed blocks (tiles)

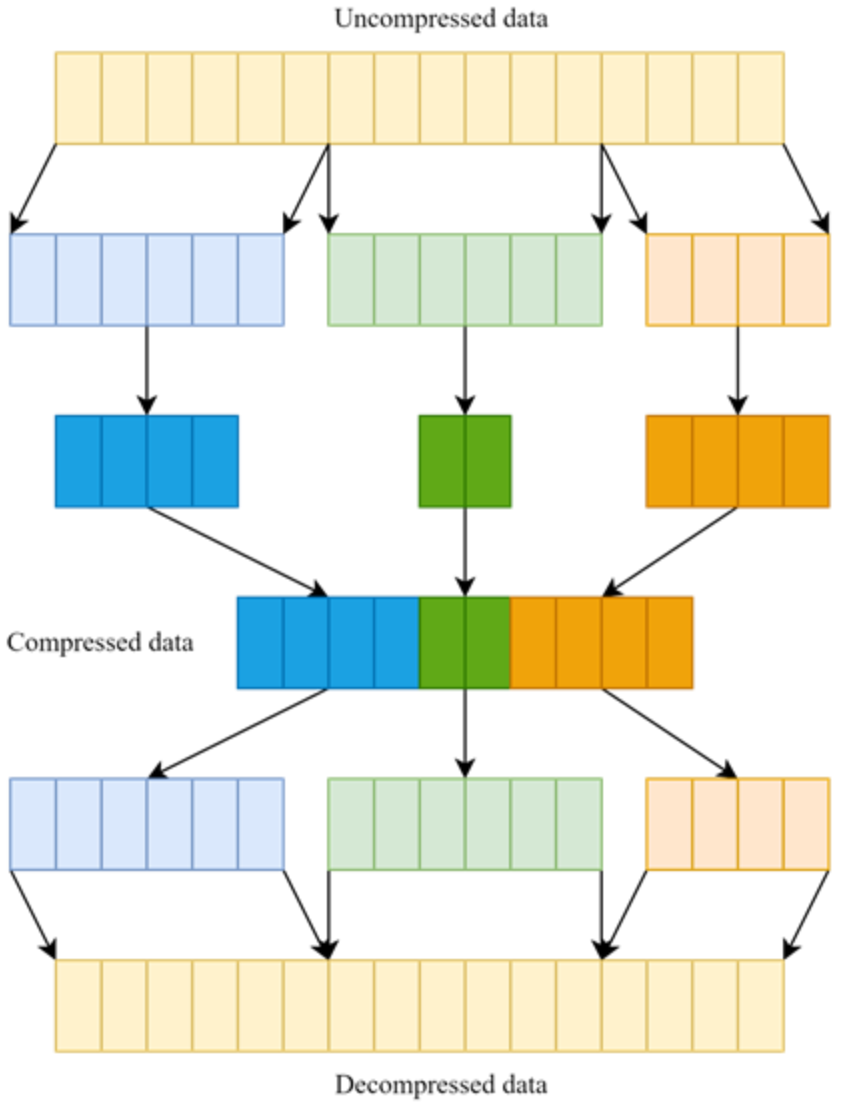
Stores the compressed and uncompressed size of each block

Pros

- Parallelism equal to the number of blocks

Cons

- Metadata lowers compression ratio
- Doesn't utilize full processing power of the GPU



Splits Parallelism

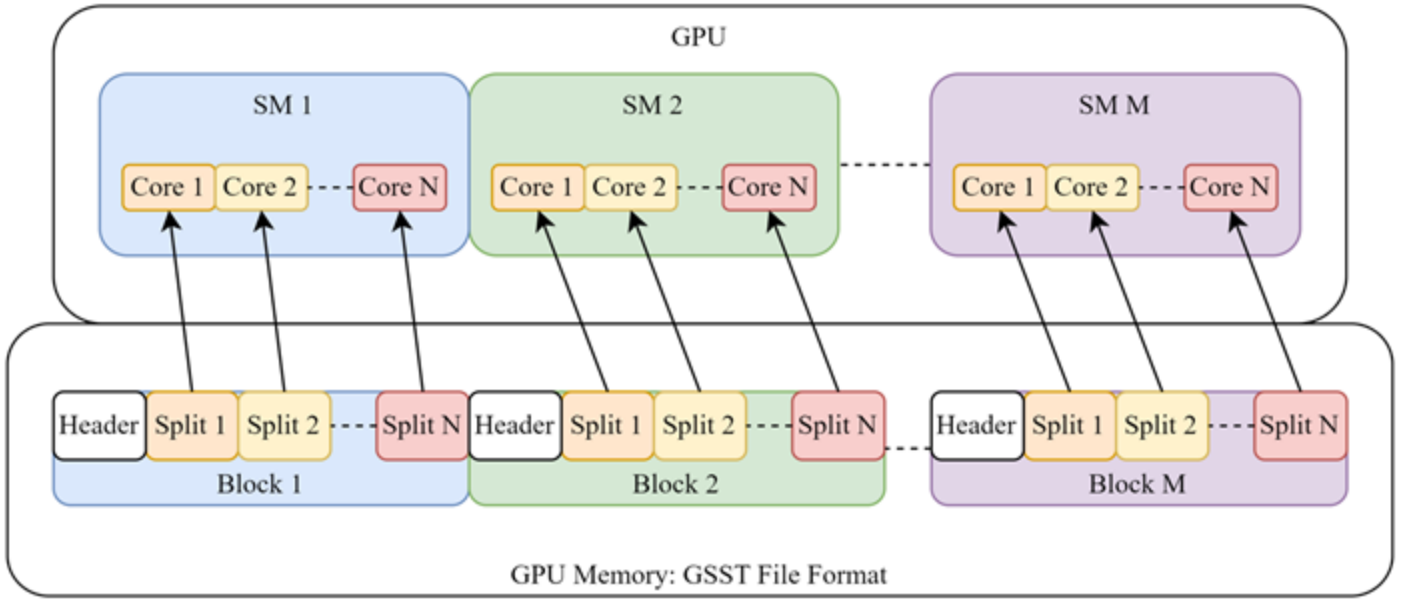
Divide a block into splits to allow parallelism within a block

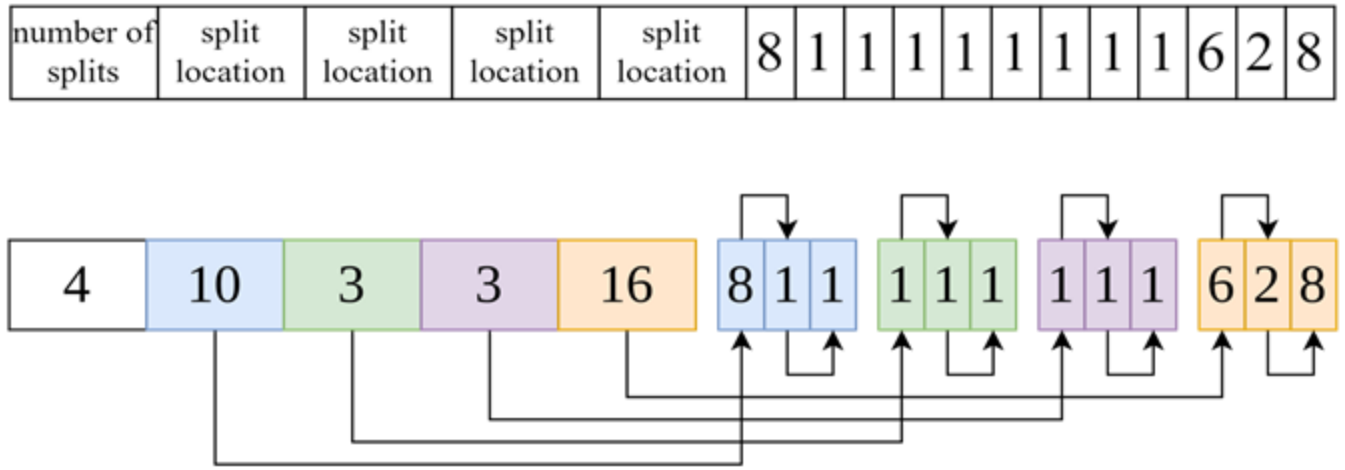
Pros

- Parallelism equal to the number of splits

Cons

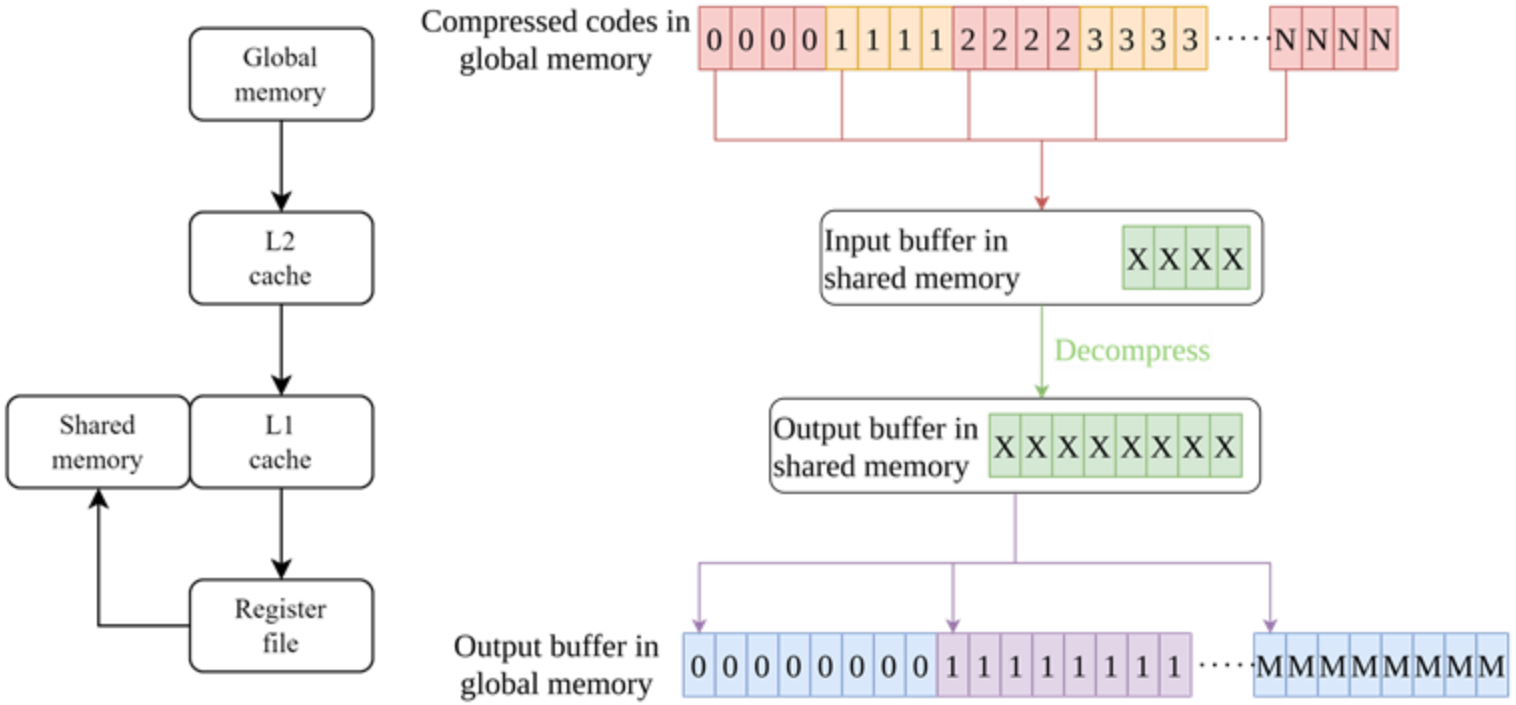
- Metadata lowers compression ratio





03

Memory Access Optimizations



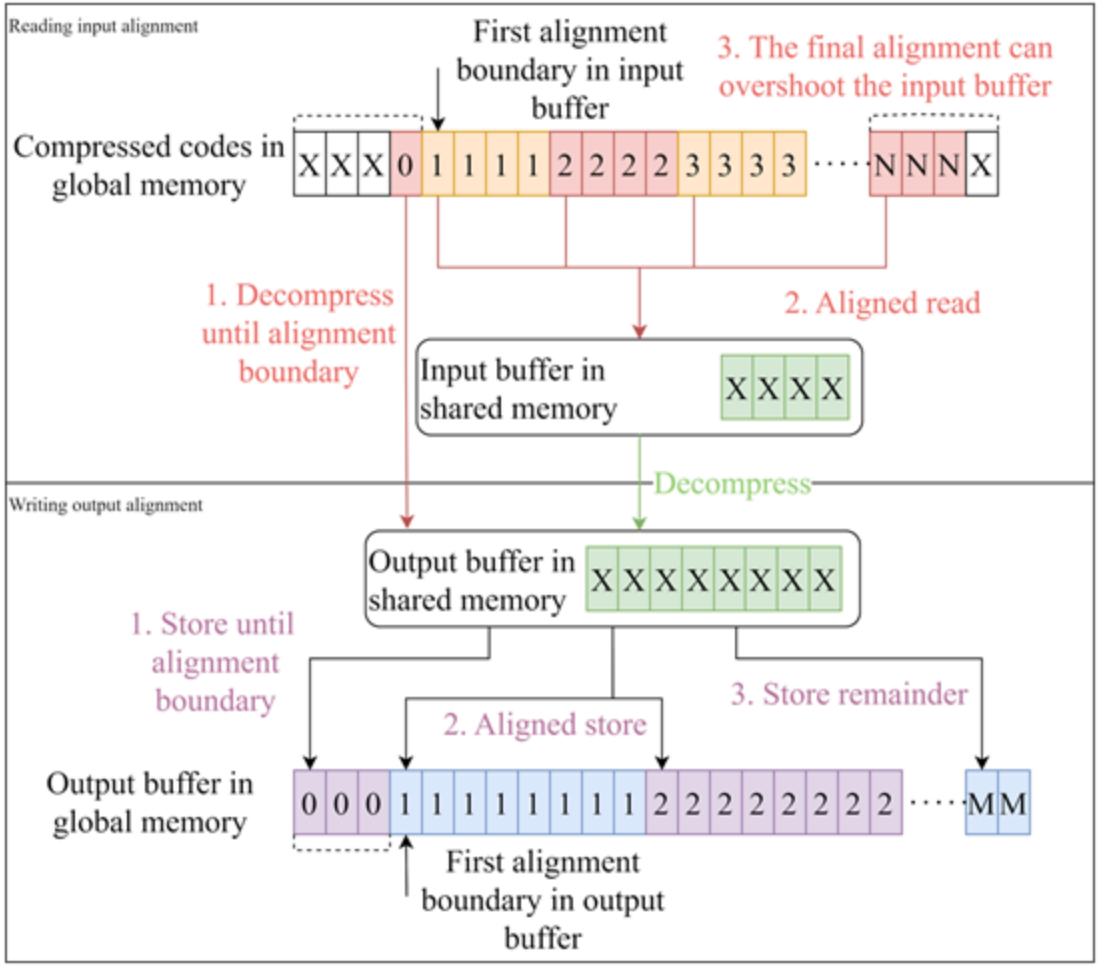
Shared Memory

Shared memory allows manually storing data in L1 cache

Shared allows efficiently sharing data between threads

- Store the symbol table in shared memory
- Buffer input and output data in shared memory

Memory Alignment



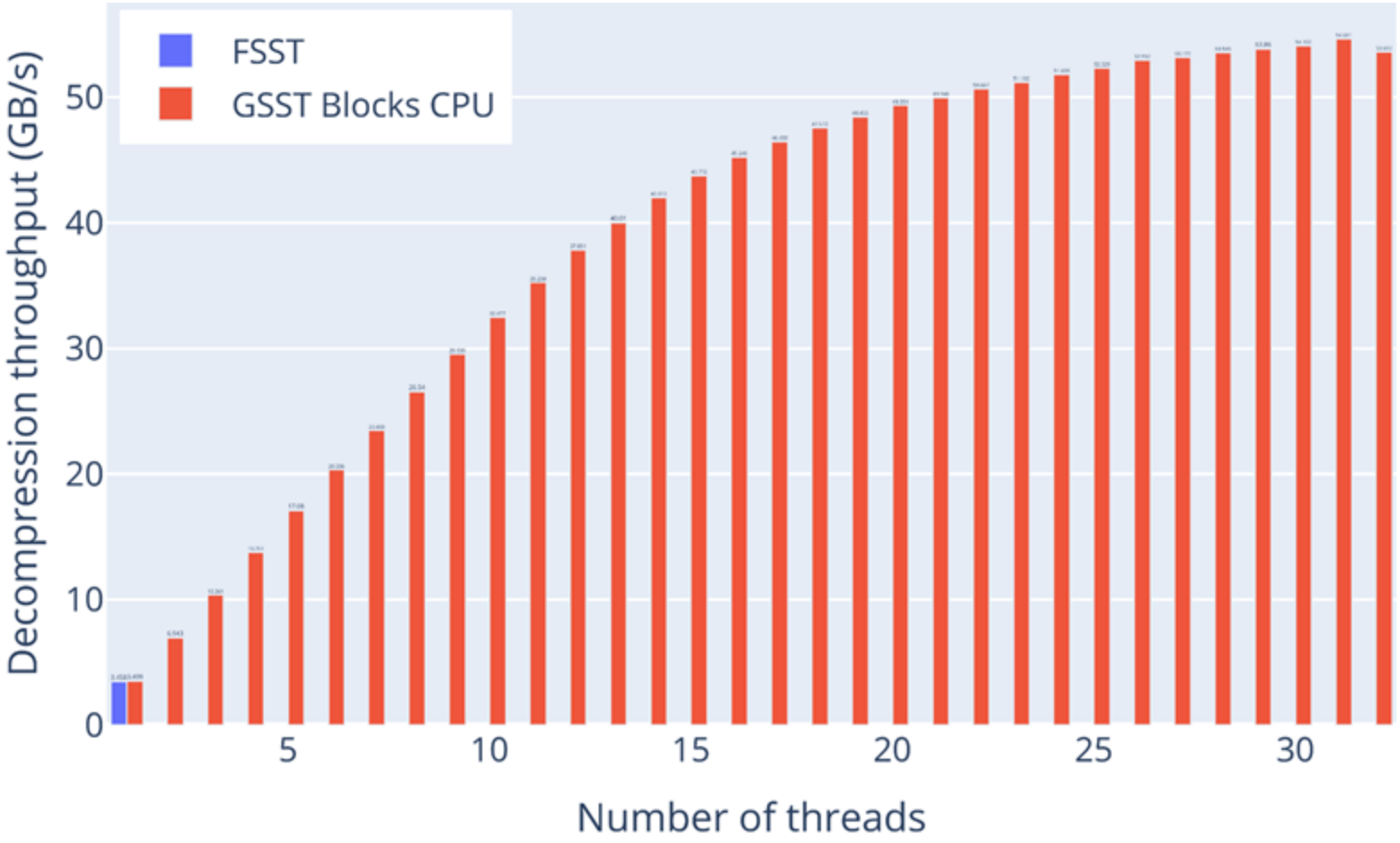
Using aligned transfers

1. Find the first aligned location in the buffer
2. Use slow data transfers until the aligned location
3. From the aligned location, use fast aligned transfers
4. Handle the remainders

04

Results and Conclusions

GSST CPU Performance

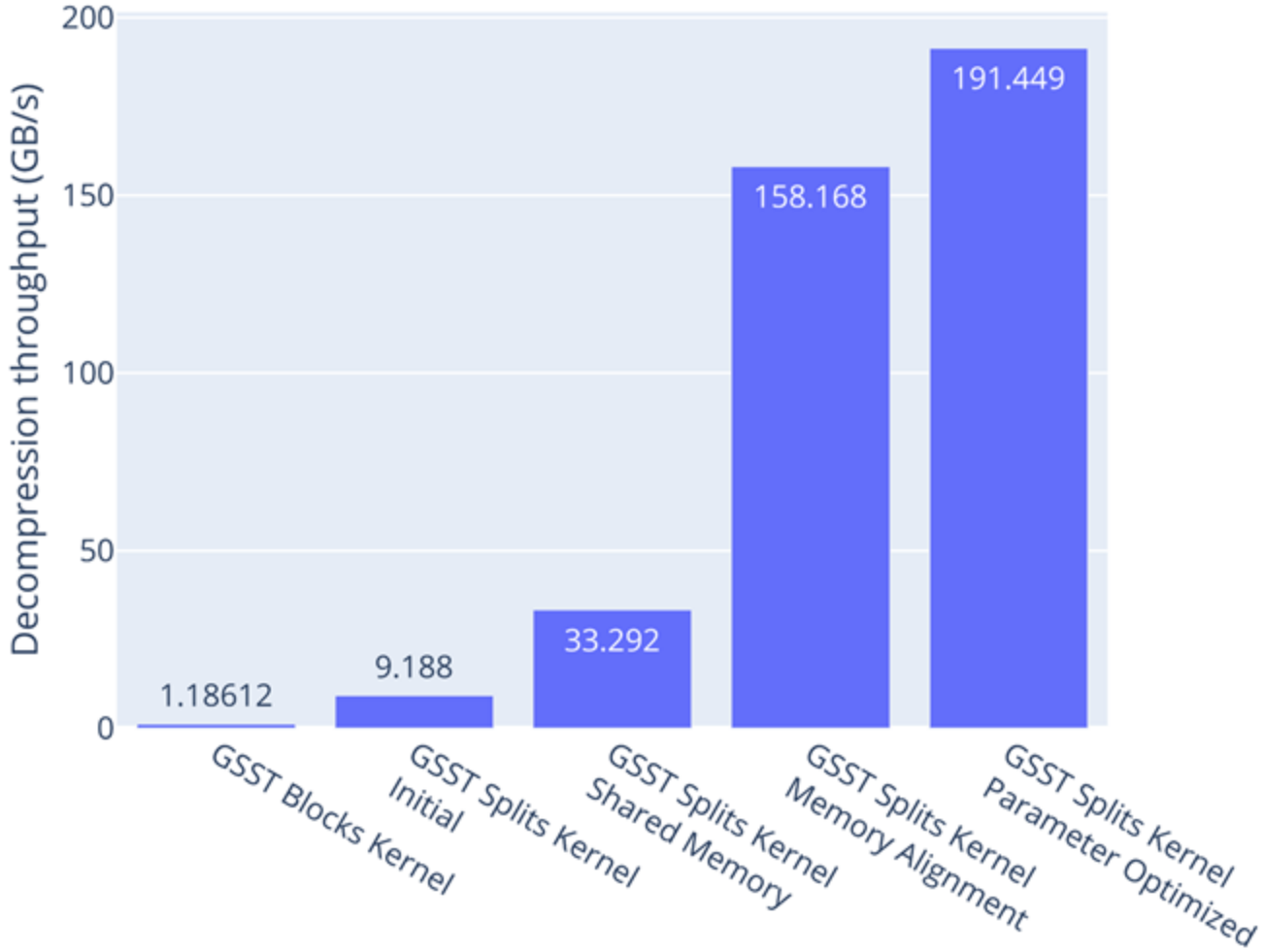


GSST format tests implementation in C++

Benchmarks are run on a platform with two Intel Xeon Platinum 8380 CPUs (40 cores each)

Throughput is measured during the decompression of 1.5 GB of generated TPC-H text data

GSST Implementations Comparison



All optimizations are implemented as CUDA kernels

Benchmarks are run on NVIDIA A100 80GB GPU

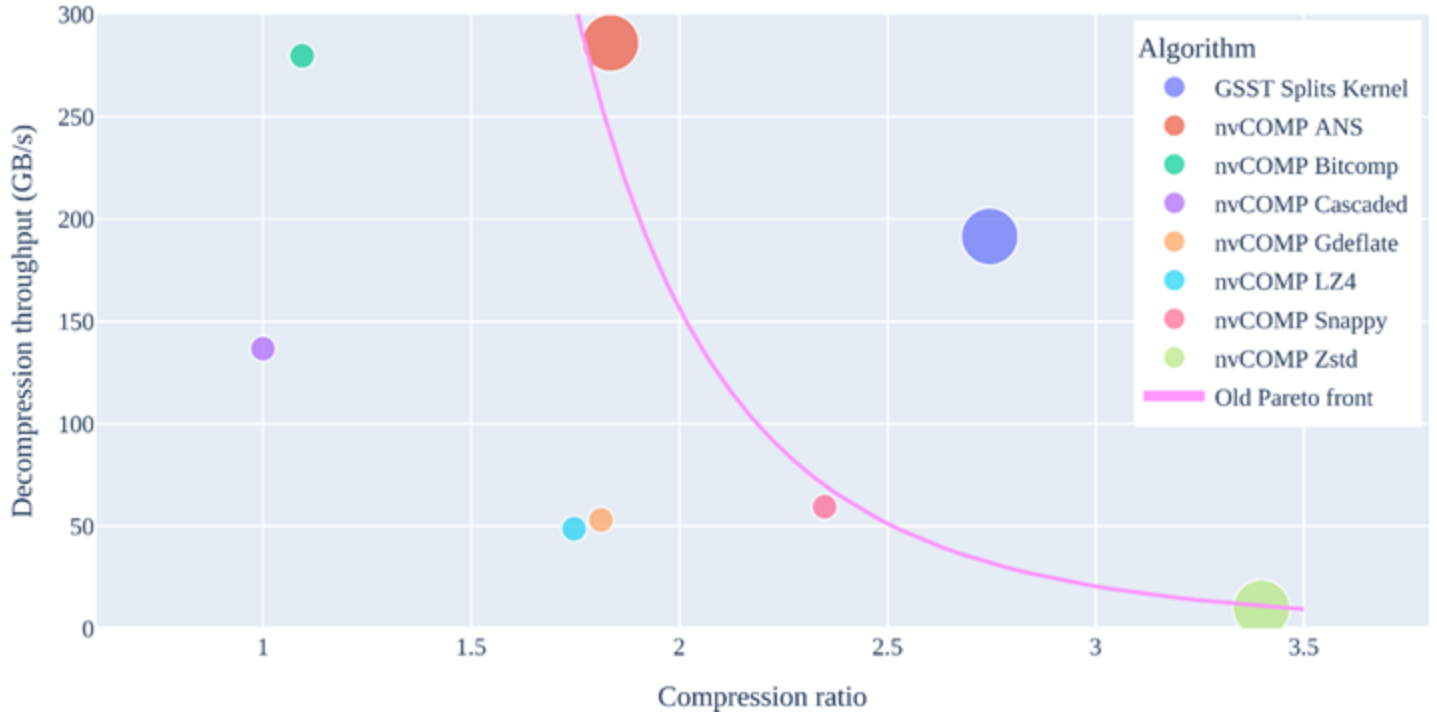
Throughput is measured during the decompression of 10 GB of generated TPC-H text data

GSST Performance

Decompression throughput of **191 GB/s** with compression ratio of **2.74**

ANS has a 49% higher throughput, but GSST 49% higher compression ratio

Zstd has a 23% higher compression ratio, but GSST has 18 times higher throughput

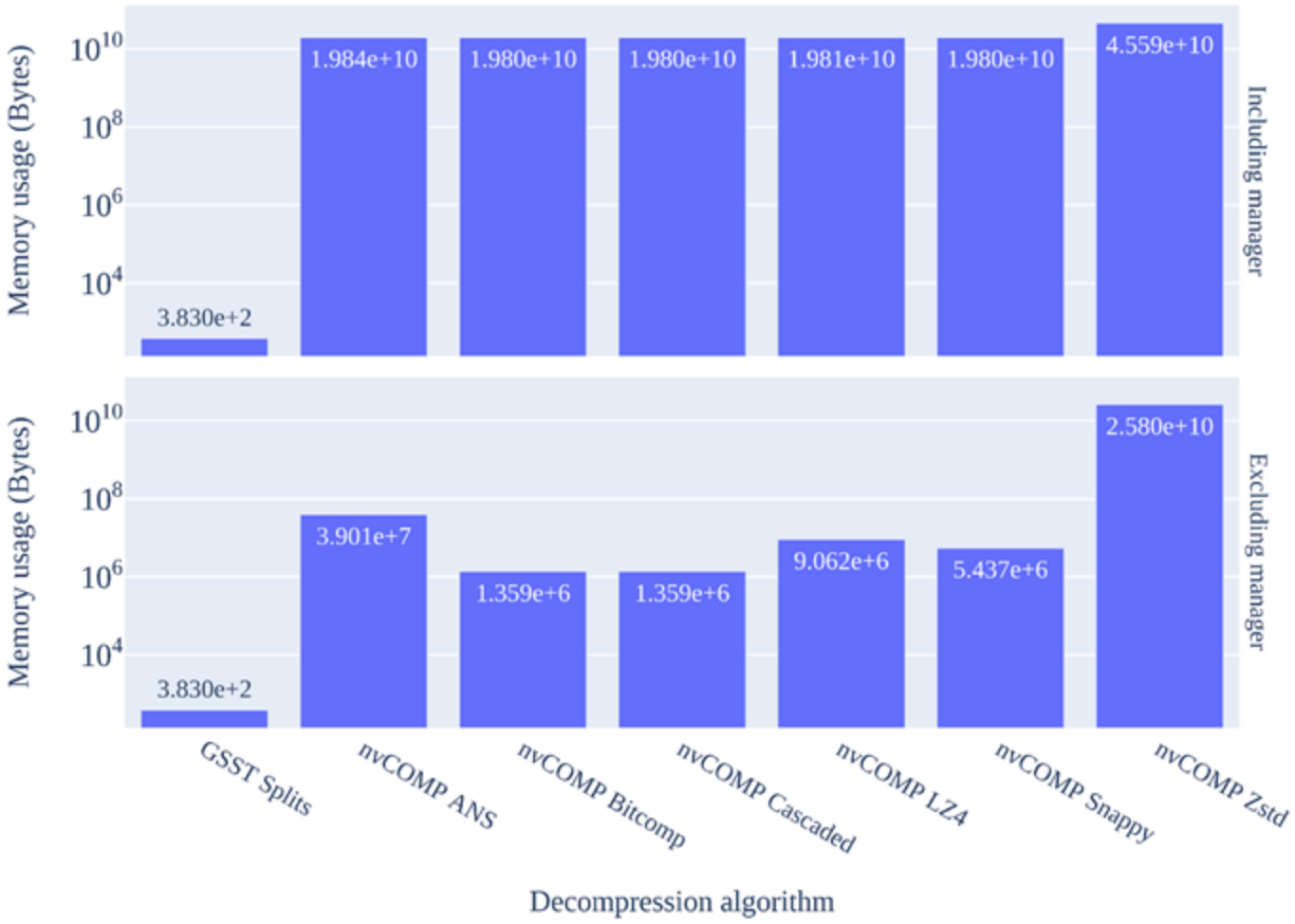


Memory Usage

State-of-the-art GPU compression in nvCOMP is extremely memory inefficient

3,500 times less memory usage compared to nvCOMP's most efficient decompression Bitcomp

67 million times less memory usage compared to nvCOMP's most memory-intensive decompression Zstd



TU Delft

Thank You!

GSST: High Throughput Parallel String Decompression on GPU | Robin Vonk

31-3-2025