GPULZ

Optimizing **LZSS** Lossless Compression for **Multi-byte** Data on Modern **GPUs**

Boyuan Zhang	*Indiana University*
Jiannan Tian	*Indiana University*
Sheng Di	*Argonne National Laboratory*
Xiaodong Yu	*Argonne National Laboratory*
Martin Swany	*Indiana University*
Dingwen Tao	*Indiana University*
Franck Cappello	*Argonne National Laboratory*

INDIANA UNIVERSITY
BLOOMINGTON

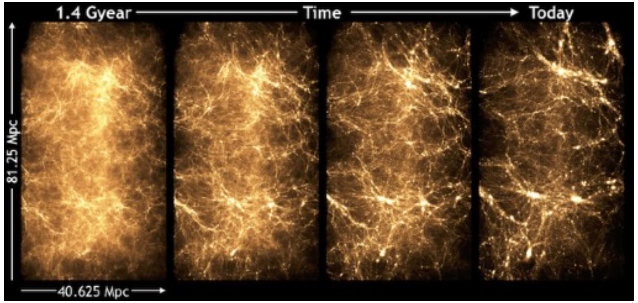
Argonne
NATIONAL LABORATORY

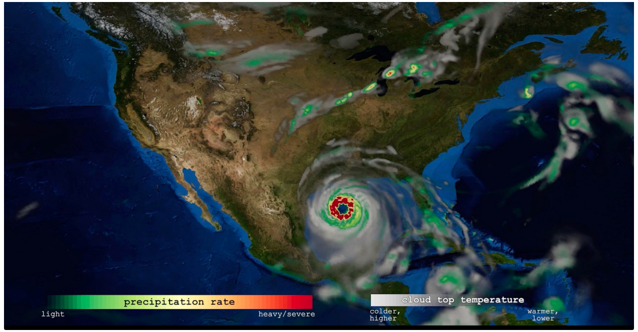
The 2023 International Conference on Supercomputing
Orlando, Florida, United States, June 21, 2023

Big Data Problem for HPC Applications

INDIANA UNIVERSITY BLOOMINGTON | Argonne NATIONAL LABORATORY

application	data scale	bottleneck
HACC cosmology simulation	**20 PB** one-trillion-particle	**use up filesystem (26 PB in total)** Mira@ANL
CESM climate simulation	**50%** vs 20% storage in hardware budget, **2017** vs 2013	**5h30m to store** NSF Blue Waters 1-TBps I/O





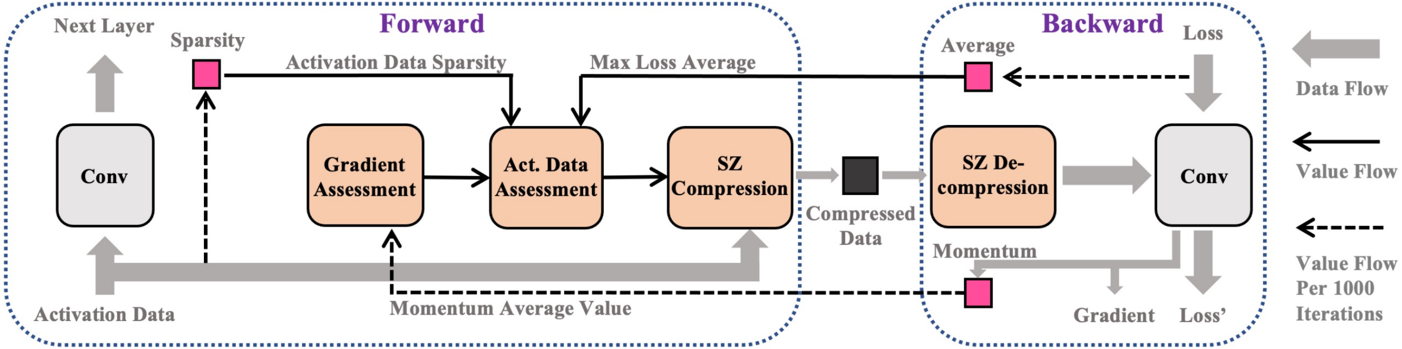
Compressions in Need on GPUs

Applications are ever-increasingly accelerated on GPUs, e.g.,

- Large-scale scientific simulation applications
- Large deep learning models

To optimize, we need to

- Reduce GPU **memory footprint**
- Speed up **CPU-GPU data transfers**



Jin, Sian, et al. "Comet: a novel memory-efficient deep learning training framework by using error-bounded lossy compression." arXiv preprint arXiv:2111.09562 (2021).

Data Compression: **Lossy**

- Not suitable for **communication** because of the error accumulation
- Application specific, error needs to be tuned by user

- Most lossy compressors **integrate one or two lossless compressors** in their pipeline to achieve higher compression ratio (**CR**)

JPEG image compression.
Quality low to high from left to right.

Data Compression: **Lossless**

- Lossless compressors have two main kinds: **entropy encoder** and **dictionary-based encoder**
- **LZ** family compressors are dictionary-based encoders
- Widely seen in **industry compressors** ,e.g., GIF, PNG, gzip, 7zip, and Zstandard.
- **LZSS** is a variation of the very first LZ family compressor, the LZ77 compressor. It generally has higher compression ratio.

PNG

Zstandard

7ZIP

LZSS Algorithm

Sliding window is a buffer (of size W. The window is empty at the beginning, then grows to size W as the input stream is processed, and “slides” along with the coding position.

Pointer contains two numbers: the first one is the starting offset, and the second one is the length of the match.

Literal represents the current byte if there is no match.

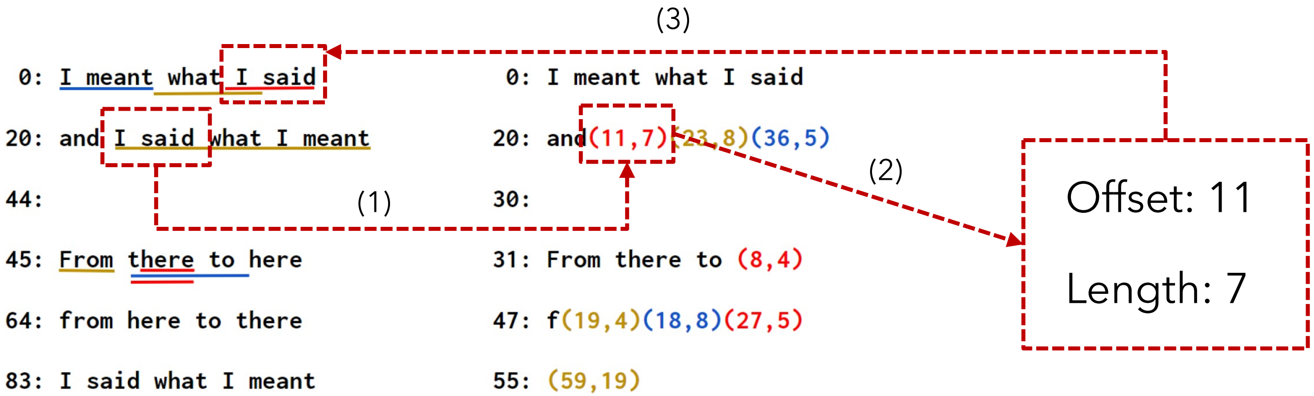
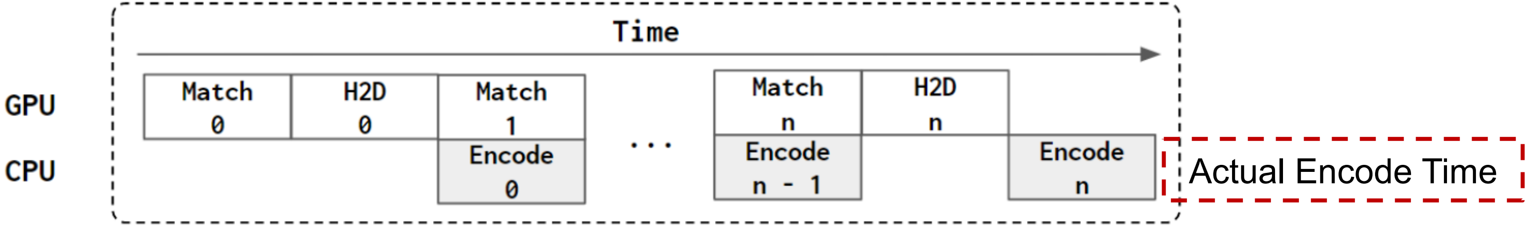


Figure 1: An example of LZSS algorithm. The left is original data, and the right is compressed data. Two numbers in brackets denote length and offset.

Issues of SOTA GPU Based LZSS

- CULZSS has a few performance-impacting issues
  - No support of **multi-byte data**
  - Fixed **data chunk size**
  - Fixed **sliding window size**
  - Under-utilization of **shared memory**
  - **CPU** encoding

- 17.04 **milli**seconds for GPU kernels time
- 387.62 **milli**seconds for end-to-end time
- The Kernel time is only 4.4% of the end-to-end time.



GPULZ: Redesigned LZSS on GPUs

GPULZ has 3 algorithm level optimizations

- Explore **optimal workflow**.
- **Two-pass prefix-sum** with kernel fusion.
- **Multi-byte** matching approach.

Three main kernels in gpulz

Two-level data partition:

- Partition dataset into **blocks** to fit the GPU **global memory**
- Partition blocks into **chunks** to fit the GPU **shared memory**

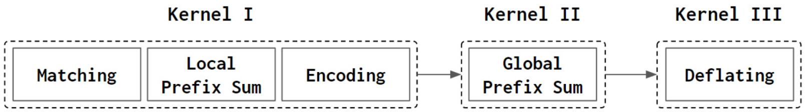


Figure 3: Workflow of our proposed GPULZ.

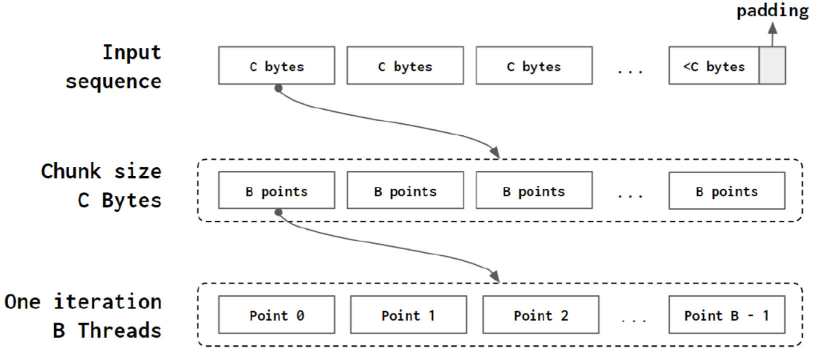


Figure 6: Data partition strategy

Optimal Workflow

INDIANA UNIVERSITY BLOOMINGTON Argonne NATIONAL LABORATORY

- Redesign the pipeline to be **fully operated on GPU** to save CPU-GPU data transfer time
- **Integrate matching kernel** and **encoding kernel** to reuse the shared memory hence reducing the global memory access overhead
- Propose **deflate kernel** to solve the discontinuous memory address

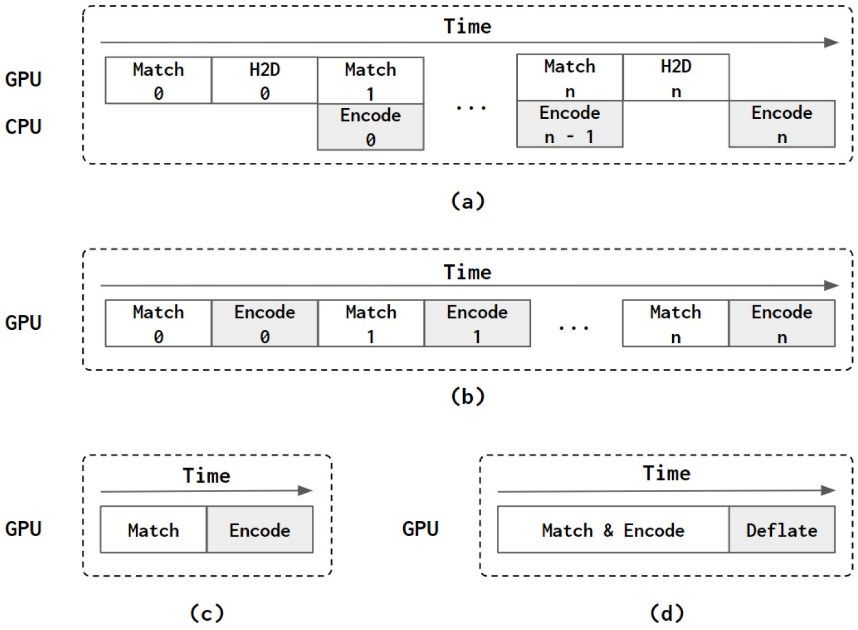


Figure 4: Three workflows of GPU LZSS.

Two-Pass Prefix-Sum

- **Local prefix sum**: enable the integration of matching and encoding
- **Global prefix sum**: add an implicit global synchronization

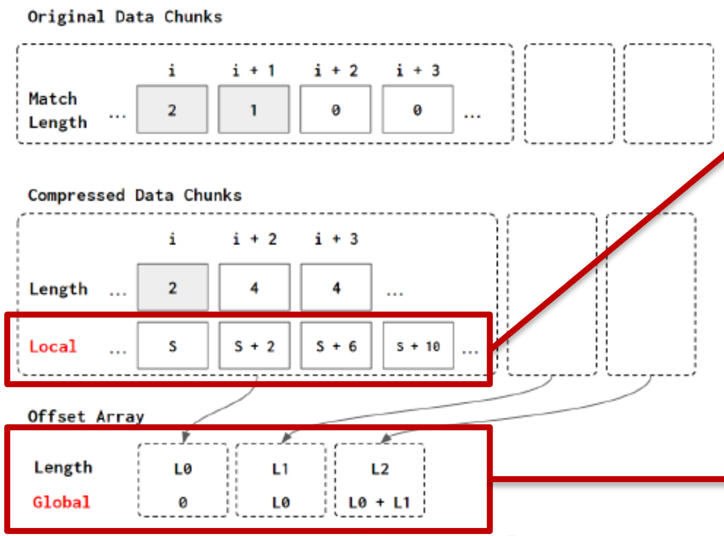
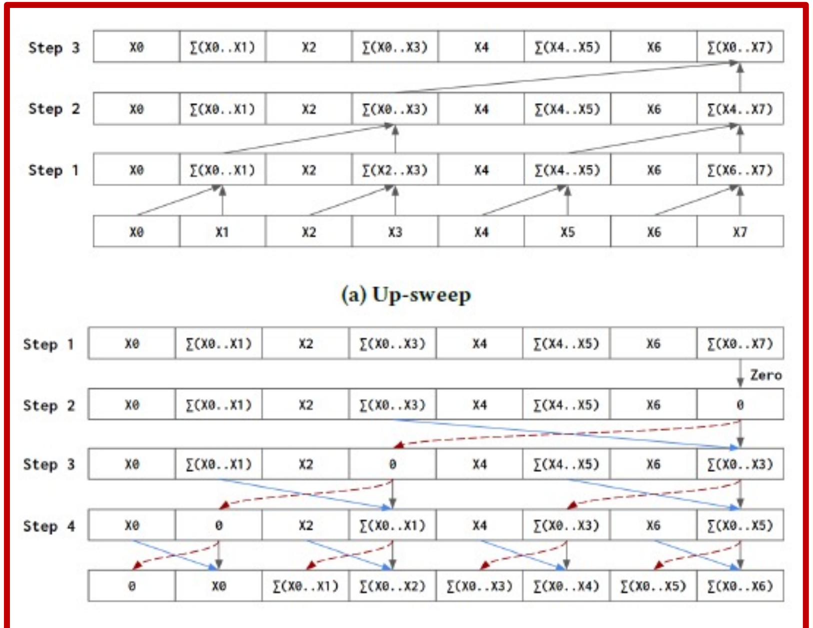


Figure 5: Our proposed two-pass prefix sum.



(a) Up-sweep

(b) Down-sweep

Figure 7: Two-sweep algorithm

`cub::DeviceScan::ExclusiveSum()`

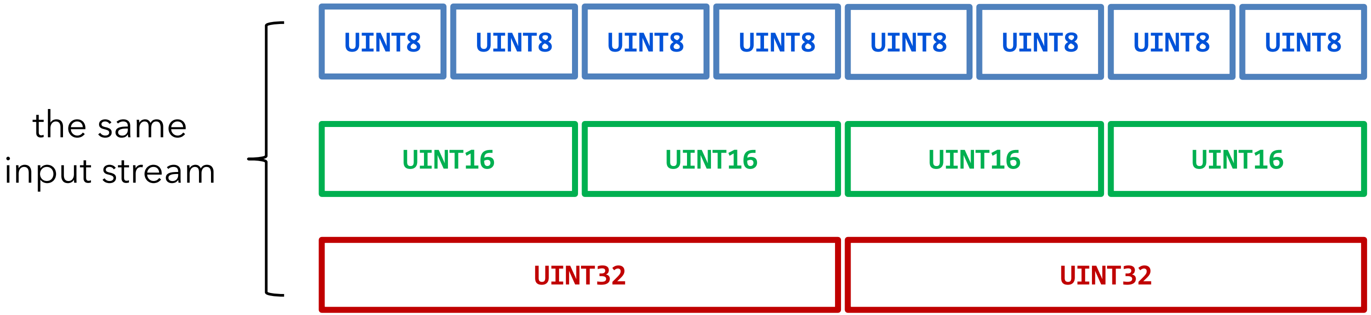
Multi-byte Matching

Redesign the matching strategy

- Find matches for **multi-byte symbols** instead of single bytes
- To gain potential compression ratio improvement
- To gain compression throughput increase

Dynamically apply multi/single-byte approach

- Multi-byte approach to low-CR dataset may cause decrease in compression ratio
- When the **actual match length is longer than window size**, the compression ratio will increase



Evaluation: Setup

INDIANA UNIVERSITY BLOOMINGTON Argonne NATIONAL LABORATORY

IU BigRed 200 HPC Cluster node

- 2x 64-core AMD EPYC 7742 CPUs at 2.25GHz .
- 4 NVIDIA Ampere A100 GPUs (108 SMs, 40GB), CUDA 11.4.120.

Workstation

- 2x 28-core Intel Xeon Gold 6238R CPUs at 2.20GHz.
- 2x NVIDIA GTX A4000 GPUs (40 SMs, 16 GB), CUDA 11.7.99.

Metrics

- Compression ratio (**CR**)
- Compression throughput ("**throughput**")

Datasets

- TPC-H benchmark
- SDRBench

Baselines

- CULZSS
- nvCOMP's LZ4

Evaluation: Impacts on CR

		chunk size: 2048			chunk size: 4096			chunk size: 8192			chunk size: 16384		
window size ↓		1 byte	2 bytes	4 bytes	1 byte	2 bytes	4 bytes	1 byte	2 bytes	4 bytes	1 byte	2 bytes	4 bytes
hurr	32	3.14	3.77	3.58	3.18	3.84	3.66	n/a	3.88	3.70	n/a	n/a	3.72
quant	64	3.79	4.39	4.05	3.86	4.50	4.18	n/a	4.56	4.25	n/a	n/a	4.28
	128	4.39	4.91	4.44	4.51	5.09	4.64	n/a	5.18	4.75	n/a	n/a	4.81
	255	4.89	5.32	4.78	5.07	5.59	5.15	n/a	5.73	5.36	n/a	n/a	5.47
hacc	32	1.55	1.67	1.59	1.55	1.68	1.60	n/a	1.68	1.61	n/a	n/a	1.61
quant	64	1.71	1.82	1.71	1.72	1.84	1.73	n/a	1.85	1.74	n/a	n/a	1.75
	128	1.87	1.97	1.83	1.88	2.00	1.86	n/a	2.02	1.88	n/a	n/a	1.89
	255	2.01	2.12	1.92	2.03	2.18	1.99	n/a	2.20	2.03	n/a	n/a	2.05
nyx	32	3.97	5.07	4.80	4.04	5.20	4.95	n/a	5.27	5.02	n/a	n/a	5.06
quant	64	5.06	6.18	5.73	5.19	6.42	6.00	n/a	6.54	6.14	n/a	n/a	6.21
	128	6.14	7.19	6.52	6.36	7.57	6.99	n/a	7.79	7.25	n/a	n/a	7.38
	255	7.08	8.03	7.11	7.46	8.65	7.94	n/a	9.01	8.42	n/a	n/a	8.64
tpch	32	1.31	1.25	1.29	1.32	1.26	1.30	n/a	1.26	1.30	n/a	n/a	1.30
int32	64	1.37	1.30	1.34	1.38	1.31	1.35	n/a	1.31	1.35	n/a	n/a	1.36
	128	1.43	1.34	1.38	1.44	1.35	1.39	n/a	1.36	1.40	n/a	n/a	1.41
	255	1.50	1.38	1.41	1.51	1.39	1.43	n/a	1.40	1.44	n/a	n/a	1.45
tpch	32	1.55	1.58	1.46	1.56	1.59	1.47	n/a	1.60	1.48	n/a	n/a	1.48
string	64	2.02	1.96	1.72	2.04	1.99	1.76	n/a	2.01	1.78	n/a	n/a	1.79
	128	2.57	2.43	2.03	2.62	2.50	2.12	n/a	2.54	2.17	n/a	n/a	2.20
	255	3.08	2.84	2.27	3.19	3.00	2.47	n/a	3.09	2.58	n/a	n/a	2.64
rtm	32	2.45	2.72	2.88	2.47	2.75	2.91	n/a	2.77	2.93	n/a	n/a	2.94
float32	64	2.59	2.80	2.92	2.61	2.83	2.96	n/a	2.85	2.98	n/a	n/a	2.99
	128	2.66	2.84	2.94	2.69	2.88	2.99	n/a	2.89	3.01	n/a	n/a	3.02
	255	2.69	2.85	2.97	2.72	2.90	3.02	n/a	2.92	3.05	n/a	n/a	3.07

Compression ratio (**CR**) of gpuLZ. Note that some fields are noted as "n/a" due to out of the limited shared memory.

Evaluation: Impacts on CR

INDIANA UNIVERSITY BLOOMINGTON

Argonne NATIONAL LABORATORY

window size ↓		chunk size: 2048 1 byte	2 bytes	4 bytes	chunk size: 4096 1 byte	2 bytes	4 bytes	chunk size: 8192 1 byte	2 bytes	4 bytes	chunk size: 16384 1 byte	2 bytes	4 bytes
hurr	32	3.14	3.77	3.58	3.18	3.84	3.66	n/a	3.88	3.70	n/a	n/a	3.72
quant	64	3.79	4.39	4.05	3.86	4.50	4.18	n/a	4.56	4.25	n/a	n/a	4.28
	128	4.39	4.91	4.44	4.51	5.09	4.64	n/a	5.18	4.75	n/a	n/a	4.81
	255	4.89	5.32	4.78	5.07	5.59	5.15	n/a	5.73	5.36	n/a	n/a	5.47
hacc	32	1.55	1.67	1.59	1.55	1.68	1.60	n/a	1.68	1.61	n/a	n/a	1.61
quant	64	1.71	1.82	1.71	1.72	1.84	1.73	n/a	1.85	1.74	n/a	n/a	1.75
	128	1.87	1.97	1.83	1.88	2.00	1.86	n/a	2.02	1.88	n/a	n/a	1.89

window size ↓		chunk size: 2048 1 byte	2 bytes	4 bytes	chunk size: 4096 1 byte	2 bytes	4 bytes	chunk size: 8192 1 byte	2 bytes	4 bytes	chunk size: 16384 1 byte	2 bytes	4 bytes
hurr	32	3.14	3.77	3.58	3.18	3.84	3.66	n/a	3.88	3.70	n/a	n/a	3.72
quant	64	3.79	4.39	4.05	3.86	4.50	4.18	n/a	4.56	4.25	n/a	n/a	4.28
	128	4.39	4.91	4.44	4.51	5.09	4.64	n/a	5.18	4.75	n/a	n/a	4.81
	255	4.89	5.32	4.78	5.07	5.59	5.15	n/a	5.73	5.36	n/a	n/a	5.47
hacc	32	1.55	1.67	1.59	1.55	1.68	1.60	n/a	1.68	1.61	n/a	n/a	1.61
quant	64	1.71	1.82	1.71	1.72	1.84	1.73	n/a	1.85	1.74	n/a	n/a	1.75
	128	1.87	1.97	1.83	1.88	2.00	1.86	n/a	2.02	1.88	n/a	n/a	1.89
	255	2.01	2.12	1.92	2.03	2.18	1.99	n/a	2.20	2.03	n/a	n/a	2.05
nyx	32	3.97	5.07	4.80	4.04	5.20	4.95	n/a	5.27	5.02	n/a	n/a	5.06
quant	64	5.06	6.18	5.73	5.19	6.42	6.00	n/a	6.54	6.14	n/a	n/a	6.21
	128	6.14	7.19	6.52	6.36	7.57	6.99	n/a	7.79	7.25	n/a	n/a	7.38
	255	7.08	8.03	7.11	7.46	8.65	7.94	n/a	9.01	8.42	n/a	n/a	8.64
tpch	32	1.31	1.25	1.29	1.32	1.26	1.30	n/a	1.26	1.30	n/a	n/a	1.30
int32	64	1.37	1.30	1.34	1.38	1.31	1.35	n/a	1.31	1.35	n/a	n/a	1.36
	128	1.43	1.34	1.38	1.44	1.35	1.39	n/a	1.36	1.40	n/a	n/a	1.41
	255	1.50	1.38	1.41	1.51	1.39	1.43	n/a	1.40	1.44	n/a	n/a	1.45
tpch	32	1.55	1.58	1.46	1.56	1.59	1.47	n/a	1.60	1.48	n/a	n/a	1.48
string	64	2.02	1.96	1.72	2.04	1.99	1.76	n/a	2.01	1.78	n/a	n/a	1.79
	128	2.57	2.43	2.03	2.62	2.50	2.12	n/a	2.54	2.17	n/a	n/a	2.20

Compression ratio (**CR**) of gpuLZ. Note that some fields are noted as “n/a” due to out of the limited shared memory.

Evaluation: Impacts on CR

INDIANA UNIVERSITY BLOOMINGTON · Argonne NATIONAL LABORATORY

Table 1: Compression ratio of GPULZ. Note that some fields are noted as "n/a" due to out of the limited shared memory.

	window size ↓	chunk size: 2048			chunk size: 4096			chunk size: 8192			chunk size: 16384		
		1 byte	2 bytes	4 bytes	1 byte	2 bytes	4 bytes	1 byte	2 bytes	4 bytes	1 byte	2 bytes	4 bytes
hurr quant	32	3.14	3.77	3.58	3.18	3.84	3.66	n/a	3.88	3.70	n/a	n/a	3.72
	64	3.79	4.39	4.05	3.86	4.50	4.18	n/a	4.56	4.25	n/a	n/a	4.28
	128	4.39	4.91	4.44	4.51	5.09	4.64	n/a	5.18	4.75	n/a	n/a	4.81
	255	4.89	5.32	4.78	5.07	5.59	5.15	n/a	5.73	5.36	n/a	n/a	5.47
hacc quant	32	1.55	1.67	1.59	1.55	1.68	1.60	n/a	1.68	1.61	n/a	n/a	1.61
	64	1.71	1.82	1.71	1.72	1.84	1.73	n/a	1.85	1.74	n/a	n/a	1.75
	128	1.87	1.97	1.83	1.88	2.00	1.86	n/a	2.02	1.88	n/a	n/a	1.89
	255	2.01	2.12	1.92	2.03	2.18	1.99	n/a	2.20	2.03	n/a	n/a	2.05
nyx quant	32	3.97	5.07	4.80	4.04	5.20	4.95	n/a	5.27	5.02	n/a	n/a	5.06
	64	5.06	6.18	5.73	5.19	6.42	6.00	n/a	6.54	6.14	n/a	n/a	6.21
	128	6.14	7.19	6.52	6.36	7.57	6.99	n/a	7.79	7.25	n/a	n/a	7.38
	255	7.08	8.03	7.11	7.46	8.65	7.94	n/a	9.01	8.42	n/a	n/a	8.64
tpch int32	32	1.31	1.25	1.29	1.32	1.26	1.30	n/a	1.26	1.30	n/a	n/a	1.30
	64	1.37	1.30	1.34	1.38	1.31	1.35	n/a	1.31	1.35	n/a	n/a	1.36
	128	1.43	1.34	1.38	1.44	1.35	1.39	n/a	1.36	1.40	n/a	n/a	1.41
	255	1.50	1.38	1.41	1.51	1.39	1.43	n/a	1.40	1.44	n/a	n/a	1.45
tpch string	32	1.55	1.58	1.46	1.56	1.59	1.47	n/a	1.60	1.48	n/a	n/a	1.48
	64	2.02	1.96	1.72	2.04	1.99	1.76	n/a	2.01	1.78	n/a	n/a	1.79
	128	2.57	2.43	2.03	2.62	2.50	2.12	n/a	2.54	2.17	n/a	n/a	2.20
	255	3.08	2.84	2.27	3.19	3.00	2.47	n/a	3.09	2.58	n/a	n/a	2.64
rtm float32	32	2.45	2.72	2.88	2.47	2.75	2.91	n/a	2.77	2.93	n/a	n/a	2.94
	64	2.59	2.80	2.92	2.61	2.83	2.96	n/a	2.85	2.98	n/a	n/a	2.99
	128	2.66	2.84	2.94	2.69	2.88	2.99	n/a	2.89	3.01	n/a	n/a	3.02
	255	2.69	2.85	2.97	2.72	2.90	3.02	n/a	2.92	3.05	n/a	n/a	3.07

Chunk size **C**

- 1.02x compression ratio
- 1.33× throughput with smaller C.

Window size W

- 1.4x compression ratio
- 3.9× throughput with smaller W

Symbol length S

- Compression ratio improvement varies
- 4.5× throughput with larger S

Evaluation: Impacts on Throughput

INDIANA UNIVERSITY BLOOMINGTON | Argonne NATIONAL LABORATORY

Table 2: Compression throughput of GPULZ on both A100 (blue) and A4000 (gray) GPUs. The red bars show the performance gain when scaling from A4000 to A100.

	window size ↓	chunk size: 2048			chunk size: 4096			chunk size: 8192			chunk size: 16384		
		1 byte	2 bytes	4 bytes	1 byte	2 bytes	4 bytes	1 byte	2 bytes	4 bytes	1 byte	2 bytes	4 bytes
hurr	32	8.1 / 4.9 (1.7×)	14.9 / 9.6 (1.6×)	29.0 / 18.1 (1.6×)	6.9 / 3.1 (2.2×)	14.8 / 8.4 (1.8×)	28.0 / 17.4 (1.6×)	n/a	11.3 / 4.4 (2.6×)	26.6 / 13.3 (2.0×)	n/a	n/a	16.0 / 7.6 (2.1×)
quant	64	4.6 / 2.9 (1.6×)	8.9 / 5.6 (1.6×)	17.5 / 11.2 (1.6×)	4.5 / 2.2 (2.0×)	8.6 / 5.3 (1.6×)	17.2 / 11.0 (1.6×)	n/a	7.4 / 3.1 (2.4×)	16.6 / 9.2 (1.8×)	n/a	n/a	11.8 / 5.6 (2.1×)
	128	2.5 / 1.6 (1.6×)	4.9 / 3.1 (1.6×)	11.0 / 6.7 (1.6×)	2.4 / 1.3 (1.8×)	4.7 / 2.9 (1.6×)	10.1 / 6.3 (1.6×)	n/a	4.3 / 2.0 (2.2×)	9.5 / 5.7 (1.7×)	n/a	n/a	7.4 / 3.7 (2.0×)
	255	1.4 / 0.9 (1.6×)	2.8 / 1.8 (1.6×)	7.0 / 4.4 (1.6×)	1.3 / 0.8 (1.7×)	2.6 / 1.6 (1.6×)	5.7 / 3.6 (1.6×)	n/a	2.3 / 1.1 (2.1×)	5.3 / 3.3 (1.6×)	n/a	n/a	4.3 / 2.3 (1.9×)
hacc	32	7.4 / 4.2 (1.8×)	13.8 / 8.5 (1.6×)	29.0 / 18.1 (1.6×)	5.8 / 2.6 (2.2×)	13.1 / 7.5 (1.7×)	27.5 / 15.5 (1.8×)	n/a	9.3 / 3.4 (2.7×)	24.6 / 10.6 (2.3×)	n/a	n/a	14.5 / 6.1 (2.4×)
quant	64	4.5 / 2.8 (1.6×)	8.2 / 5.3 (1.5×)	19.2 / 11.4 (1.7×)	4.1 / 2.1 (2.0×)	8.2 / 5.1 (1.6×)	19.1 / 11.4 (1.7×)	n/a	6.6 / 2.7 (2.4×)	17.4 / 8.4 (2.1×)	n/a	n/a	11.1 / 5.3 (2.1×)
	128	2.6 / 1.7 (1.5×)	4.8 / 3.1 (1.6×)	12.4 / 6.3 (2.0×)	2.6 / 1.4 (1.9×)	4.7 / 3.0 (1.5×)	11.1 / 6.7 (1.6×)	n/a	4.2 / 2.0 (2.1×)	11.1 / 6.0 (1.8×)	n/a	n/a	7.9 / 3.6 (2.2×)
	255	1.5 / 1.0 (1.6×)	2.7 / 1.8 (1.5×)	7.4 / 4.4 (1.7×)	1.5 / 0.8 (1.8×)	2.7 / 1.7 (1.6×)	6.7 / 3.9 (1.7×)	n/a	2.5 / 1.2 (2.1×)	6.0 / 3.5 (1.7×)	n/a	n/a	4.9 / 2.5 (2.0×)
nyx	32	9.5 / 6.0 (1.6×)	15.7 / 10.1 (1.5×)	30.1 / 19.1 (1.6×)	7.5 / 4.0 (1.9×)	15.8 / 9.1 (1.7×)	30.3 / 18.8 (1.6×)	n/a	12.4 / 5.6 (2.2×)	29.2 / 14.7 (2.0×)	n/a	n/a	18.1 / 8.4 (2.2×)
quant	64	5.7 / 3.6 (1.6×)	9.4 / 6.2 (1.5×)	19.8 / 11.6 (1.7×)	5.4 / 2.8 (1.9×)	9.3 / 6.2 (1.5×)	18.0 / 11.4 (1.6×)	n/a	8.1 / 3.8 (2.2×)	17.9 / 10.8 (1.7×)	n/a	n/a	12.9 / 6.3 (2.0×)
	128	3.1 / 1.9 (1.6×)	5.5 / 3.6 (1.5×)	11.3 / 7.1 (1.6×)	3.1 / 1.7 (1.9×)	5.9 / 3.4 (1.7×)	10.3 / 6.8 (1.5×)	n/a	5.0 / 2.5 (2.0×)	10.2 / 6.5 (1.6×)	n/a	n/a	8.7 / 4.6 (1.9×)
	255	1.8 / 1.0 (1.7×)	3.6 / 2.1 (1.7×)	6.9 / 4.9 (1.4×)	1.7 / 0.9 (1.8×)	3.2 / 1.9 (1.7×)	6.6 / 4.1 (1.6×)	n/a	3.1 / 1.4 (2.2×)	6.3 / 3.9 (1.6×)	n/a	n/a	5.3 / 2.8 (1.9×)
tpch	32	7.1 / 3.9 (1.8×)	12.1 / 8.3 (1.5×)	25.4 / 14.9 (1.7×)	5.2 / 2.3 (2.3×)	11.7 / 6.2 (1.9×)	21.4 / 13.5 (1.6×)	n/a	7.7 / 3.1 (2.5×)	19.4 / 9.3 (2.1×)	n/a	n/a	10.5 / 5.0 (2.1×)
int32	64	4.4 / 2.7 (1.6×)	7.9 / 5.1 (1.5×)	16.3 / 10.2 (1.6×)	3.8 / 1.7 (2.2×)	7.5 / 4.5 (1.7×)	14.6 / 9.8 (1.5×)	n/a	5.9 / 2.4 (2.5×)	14.2 / 7.1 (2.0×)	n/a	n/a	8.2 / 4.2 (2.0×)
	128	2.4 / 1.6 (1.5×)	4.8 / 3.0 (1.6×)	10.2 / 6.3 (1.6×)	2.2 / 1.1 (2.0×)	4.5 / 2.8 (1.6×)	9.2 / 5.6 (1.7×)	n/a	3.8 / 1.7 (2.3×)	8.4 / 5.0 (1.7×)	n/a	n/a	6.4 / 3.1 (2.1×)
	255	1.3 / 0.9 (1.6×)	2.8 / 1.7 (1.6×)	6.7 / 4.0 (1.7×)	1.2 / 0.7 (1.8×)	2.4 / 1.6 (1.5×)	5.8 / 3.5 (1.7×)	n/a	2.1 / 1.0 (2.1×)	5.0 / 3.1 (1.6×)	n/a	n/a	3.8 / 2.0 (1.9×)
tpch	32	7.1 / 4.2 (1.7×)	12.5 / 8.0 (1.6×)	22.9 / 13.8 (1.7×)	5.3 / 2.5 (2.2×)	11.4 / 6.2 (1.8×)	21.1 / 12.6 (1.7×)	n/a	8.0 / 3.5 (2.3×)	19.0 / 8.4 (2.3×)	n/a	n/a	10.0 / 4.8 (2.1×)
string	64	4.7 / 3.1 (1.5×)	8.4 / 5.2 (1.6×)	15.2 / 9.5 (1.6×)	4.2 / 2.0 (2.1×)	7.6 / 5.1 (1.5×)	15.6 / 9.2 (1.7×)	n/a	6.3 / 2.9 (2.2×)	14.0 / 6.8 (2.0×)	n/a	n/a	8.2 / 4.1 (2.0×)
	128	2.4 / 1.7 (1.4×)	4.8 / 3.3 (1.5×)	10.7 / 6.0 (1.8×)	2.6 / 1.3 (2.0×)	4.7 / 3.1 (1.5×)	9.4 / 5.7 (1.7×)	n/a	4.0 / 2.0 (2.0×)	8.2 / 4.8 (1.7×)	n/a	n/a	5.7 / 3.5 (1.6×)
	255	1.4 / 0.9 (1.5×)	2.6 / 1.8 (1.4×)	6.9 / 3.7 (1.8×)	1.4 / 0.8 (1.8×)	2.6 / 1.7 (1.5×)	6.1 / 3.3 (1.8×)	n/a	2.3 / 1.2 (1.9×)	4.7 / 3.3 (1.4×)	n/a	n/a	3.7 / 2.3 (1.6×)
rtm	32	7.3 / 4.2 (1.7×)	14.3 / 9.0 (1.6×)	28.4 / 17.6 (1.6×)	6.1 / 2.8 (2.2×)	13.9 / 7.5 (1.8×)	28.6 / 17.2 (1.7×)	n/a	11.2 / 3.9 (2.9×)	26.6 / 13.1 (2.0×)	n/a	n/a	16.2 / 7.4 (2.2×)
float32	64	4.5 / 2.9 (1.6×)	9.3 / 5.5 (1.7×)	17.7 / 11.2 (1.6×)	3.9 / 2.2 (1.8×)	8.3 / 5.1 (1.6×)	17.4 / 10.9 (1.6×)	n/a	7.0 / 3.0 (2.3×)	16.8 / 9.1 (1.8×)	n/a	n/a	12.4 / 5.5 (2.2×)
	128	2.5 / 1.8 (1.4×)	4.9 / 3.4 (1.4×)	10.8 / 6.8 (1.6×)	2.4 / 1.4 (1.7×)	4.7 / 3.1 (1.5×)	10.0 / 6.4 (1.6×)	n/a	4.2 / 1.9 (2.2×)	9.6 / 5.6 (1.7×)	n/a	n/a	7.5 / 3.7 (2.1×)
	255	1.4 / 1.0 (1.4×)	3.1 / 2.0 (1.5×)	8.1 / 4.8 (1.7×)	1.3 / 0.8 (1.6×)	3.1 / 1.8 (1.7×)	6.1 / 3.8 (1.6×)	n/a	2.6 / 1.3 (2.0×)	5.8 / 3.3 (1.8×)	n/a	n/a	4.5 / 2.6 (1.7×)

Chunk size C

- 1.02x compression ratio
- 1.33× throughput with smaller C.

Window size W

- 1.4x compression ratio
- 3.9× throughput with smaller W

Symbol length S

- Compression ratio improvement varies
- 4.5× throughput with larger S

Evaluation: Comparison with SOTA Works

INDIANA UNIVERSITY BLOOMINGTON

Argonne NATIONAL LABORATORY

GPULZ throughput speedup on A4000 (best case)

- **272.1×** over CULZSS
- **8.7×** over nvCOMP-LZ4

GPULZ has an up to

- **1.4×** CR compared to CULZSS
- **2.1×** CR over nvCOMP's LZ4

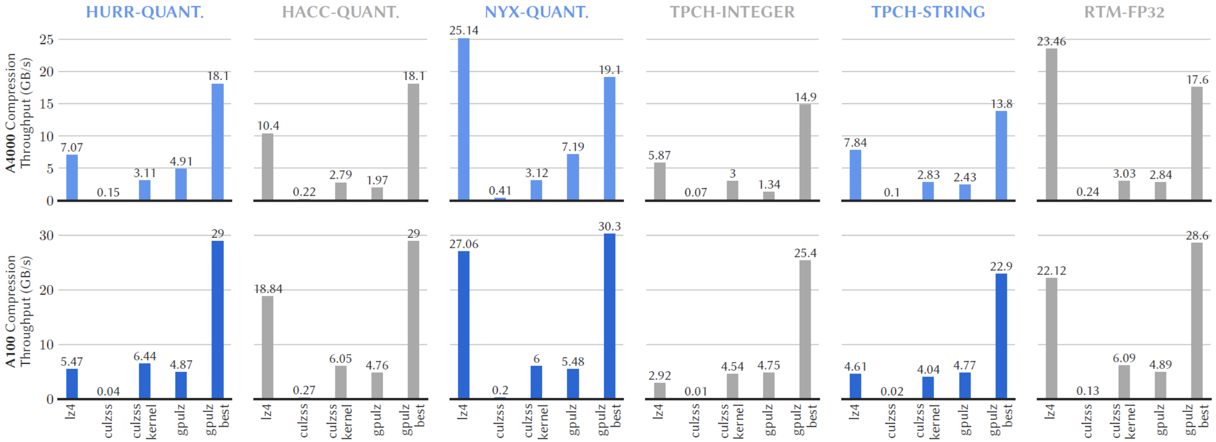


Figure 9: Compression throughput of different GPU compressors on A100 and A4000.

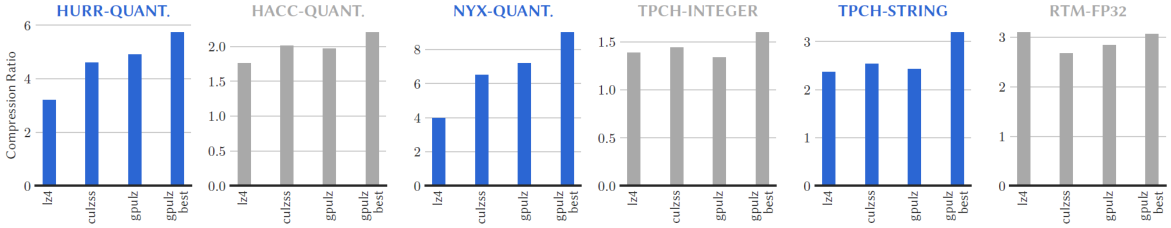


Figure 8: Compression ratio of different GPU compressors.

A Use-case of GPULZ

- Applied as a last step lossless encoder in cuSZ
- Improvement of compression ratio of **1.9x ~ 8.7x** on average
- CPU SZ on 32 cores has a throughput of **2 ~ 3** GB/s but the overall throughput is limited by CPU GPU data movement

Table 3: Comparison of compression ratio and throughput (GB/s) between original cuSZ and improved cuSZ (with GPULZ) on A100 platform.

Dataset	cuSZ		cuSZ w/ GPULZ	
	CR	THR	CR	THR
CESM	22.6	12.0	43.2	2.7
Hurricane	24.3	31.9	29.1	5.9
Nyx	30.1	87.2	74.8	10.4
RTM	28.6	49.2	249.8	7.2

Conclusion & Future Work

INDIANA UNIVERSITY BLOOMINGTON Argonne NATIONAL LABORATORY

In this paper, we propose a series of optimizations on LZSS algorithms for multi-byte data on GPUs. Specifically,

1. We develop a new strategy for multi-byte pattern matching,
2. We explore the optimal workflow
3. We optimize the prefix-sum operation,
4. and fuse multiple GPU kernels to improve both compression ratio and throughput

gpuLZ achieves up to 272.1× speedup and up to 1.4× higher compression ratio over state-of-the-art solutions.

In the future, we plan to

1. evaluate gpuLZ on more multi-byte datasets. We will attempt to develop an analytical model for searching the optimal parameter combination for different datasets.
2. In addition, we will integrate gpuLZ into more data-intensive applications running on different parallel and distributed systems.
3. adapt gpuLZ to other GPU platforms by using code translation tools such as HIPFY for AMD GPUs and SYCLomatic for Intel GPUs

Acknowledgment

This R&D was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations–the Office of Science and the National Nuclear Security Administration, responsible for the planning and preparation of a capable exascale ecosystem. This repository was based upon work supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357, and also supported by the National Science Foundation under Grants SHF-1617488, SHF-1619253, OAC-2003709, OAC-1948447/2034169, and OAC-2003624.

Thank you.

Questions?

github.com/hipdac-lab/ICS23-GPULZ

contact us

Boyuan Zhang
bozhan@iu.edu

Dr. Dingwen Tao
ditao@iu.edu