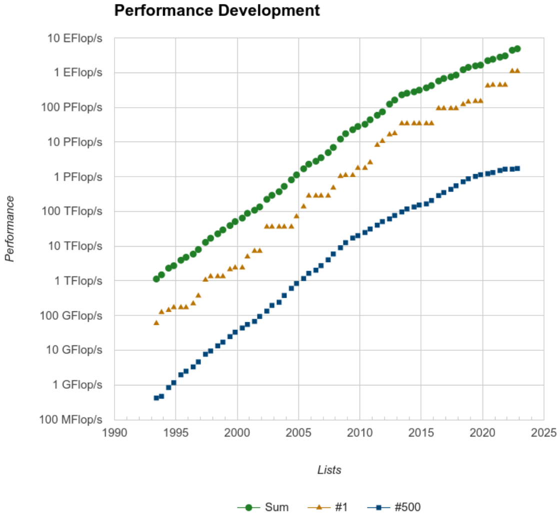
Evaluation of GPU-Compression Algorithms for CUDA-Aware MPI

Marco Vogel and Prof. Dr. Lena Oden

Fernuniversität in Hagen - Chair of Computer Engineering

September 14, 2023

HPC Systems - TOP500



Cuda-Aware MPI

Cuda-Aware MPI

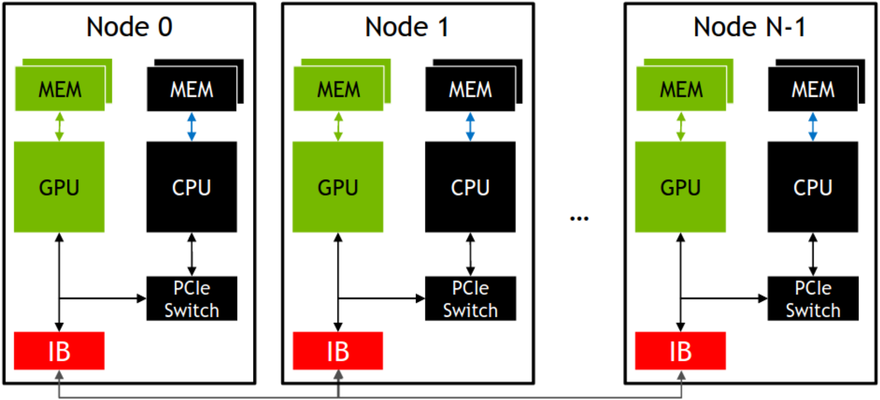


Figure: [1]

Challenge

With a growing number of compute nodes the required amount of communication between them also grows

runtime analysis PSDNS

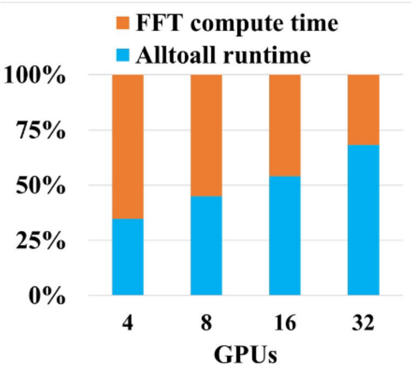


Figure: [2]

potential for optimization

utilize gpu to compress messages before exchanging them between nodes

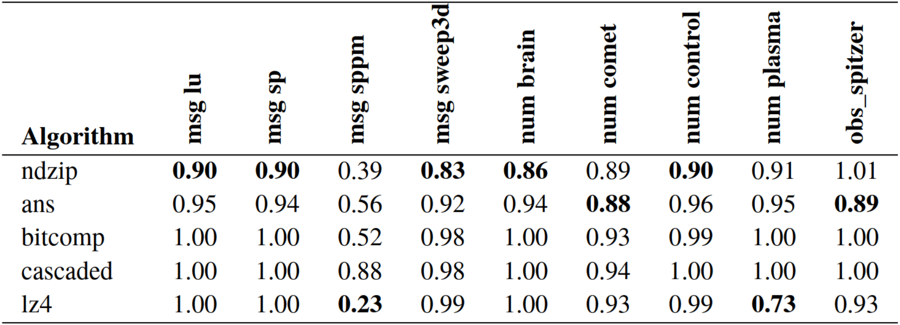
gpu compression

- simulations often utilize floating point numbers (FP64, FP32)
- compressor must be adapted for gpu execution and able to compress FP data well
- selected compression algorithms: ndzip, nvcomp

benchmarking compression algorithms

- realistic test datasets needed
- relevant data points:
  - compression ratio
  - compression throughput
  - decompression throughput

compression ratio FP64

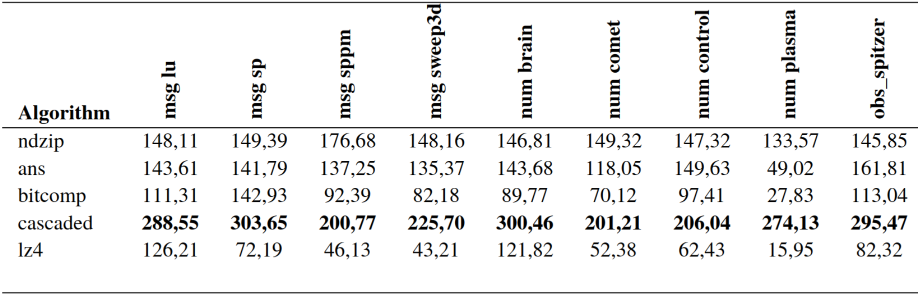


Algorithm	msg lu	msg sp	msg sppm	msg sweep3d	num brain	num comet	num control	num plasma	obs_spitzer
ndzip	**0.90**	**0.90**	0.39	**0.83**	**0.86**	0.89	**0.90**	0.91	1.01
ans	0.95	0.94	0.56	0.92	0.94	**0.88**	0.96	0.95	**0.89**
bitcomp	1.00	1.00	0.52	0.98	1.00	0.93	0.99	1.00	1.00
cascaded	1.00	1.00	0.88	0.98	1.00	0.94	1.00	1.00	1.00
lz4	1.00	1.00	**0.23**	0.99	1.00	0.93	0.99	**0.73**	0.93

compression throughput FP64 (GB/s)

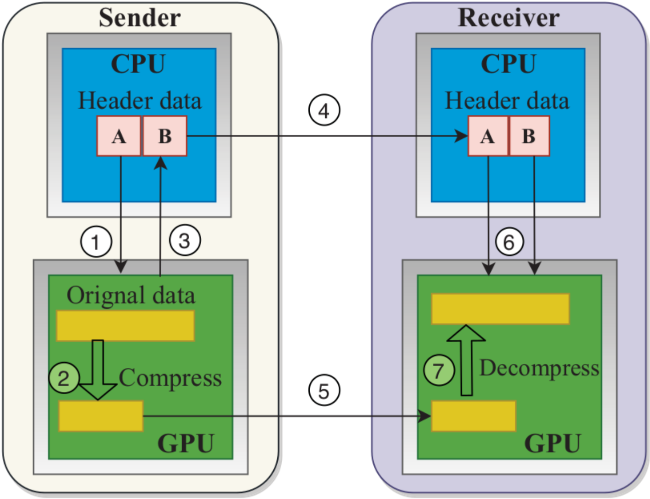
Algorithm	**msg lu**	**msg sp**	**msg sppm**	**msg sweep3d**	**num brain**	**num comet**	**num control**	**num plasma**	**obs_spitzer**
ndzip	111.30	113.10	152.70	113.36	111.62	110.90	110.57	95.47	109.10
ans	71.75	89.88	80.20	76.83	93.33	77.82	72.21	36.84	78.35
bitcomp	**264.38**	**274.26**	**292.01**	**269.85**	**267.40**	**257.57**	**267.63**	**216.88**	**271.05**
cascaded	35.56	37.50	47.12	42.53	39.05	40.63	39.06	25.78	37.80
lz4	1.64	1.63	10.67	1.70	1.63	1.84	1.58	2.69	1.78

decompression throughput FP64 (GB/s)



Algorithm	**msg lu**	**msg sp**	**msg sppm**	**msg sweep3d**	**num brain**	**num comet**	**num control**	**num plasma**	**obs_spitzer**
ndzip	148,11	149,39	176,68	148,16	146,81	149,32	147,32	133,57	145,85
ans	143,61	141,79	137,25	135,37	143,68	118,05	149,63	49,02	161,81
bitcomp	111,31	142,93	92,39	82,18	89,77	70,12	97,41	27,83	113,04
cascaded	**288,55**	**303,65**	**200,77**	**225,70**	**300,46**	**201,21**	**206,04**	**274,13**	**295,47**
lz4	126,21	72,19	46,13	43,21	121,82	52,38	62,43	15,95	82,32

message exchange process

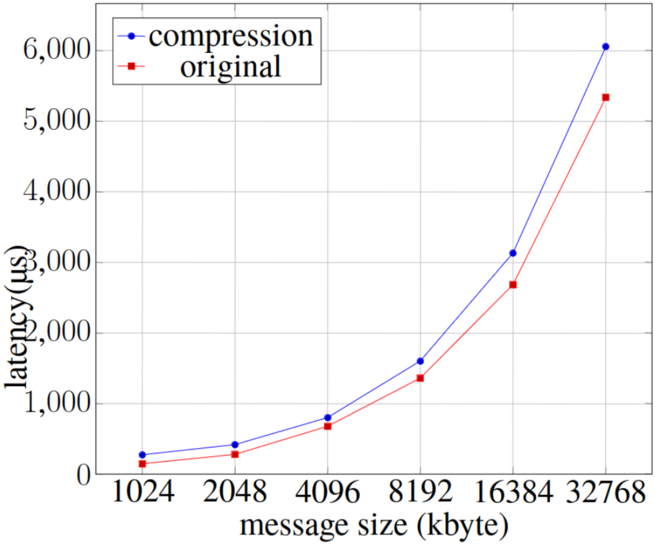


Source: [3][p.447]

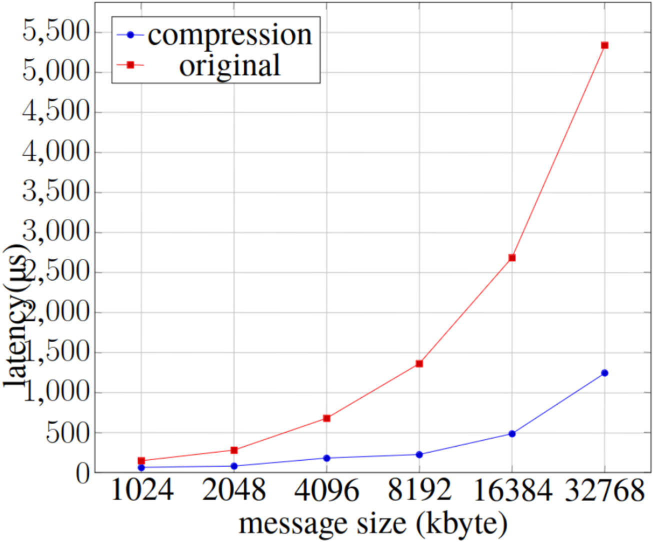
Integration in OSU benchmarks

- modification of the osu_latency.c and osu_bcast.c source code
  - messages exchanged between nodes are filled with selected test data sets
  - data gets compressed by gpu of sending node
  - data gets decompressed on gpu of receiving node

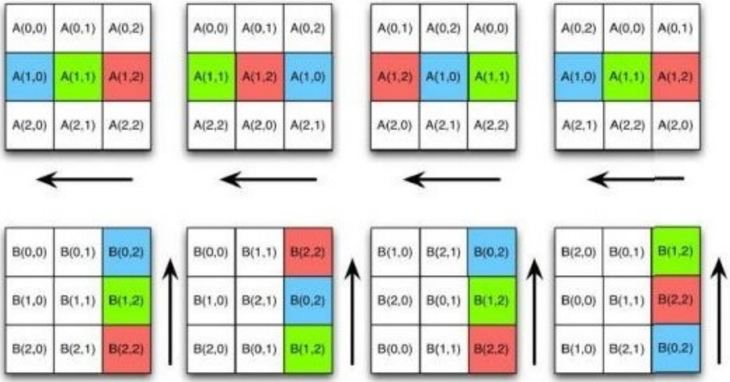
osu_latency - obs_spitzer.fp64



osu_latency - msg_sppm.fp64



Cannon's algorithm



A(0,0)	A(0,1)	A(0,2)
A(1,0)	A(1,1)	A(1,2)
A(2,0)	A(2,1)	A(2,2)

B(0,0)	B(0,1)	B(0,2)
B(1,0)	B(1,1)	B(1,2)
B(2,0)	B(2,1)	B(2,2)

Initial A, B

A(0,0)	A(0,1)	A(0,2)
A(1,1)	A(1,2)	A(1,0)
A(2,2)	A(2,0)	A(2,1)

B(0,0)	B(1,1)	B(2,2)
B(1,0)	B(2,1)	B(0,2)
B(2,0)	B(0,1)	B(1,2)

A, B after skewing

A(0,1)	A(0,2)	A(0,0)
A(1,2)	A(1,0)	A(1,1)
A(2,0)	A(2,1)	A(2,2)

B(1,0)	B(2,1)	B(0,2)
B(2,0)	B(0,1)	B(1,2)
B(0,0)	B(1,1)	B(2,2)

A, B after shifting
k=1

A(0,2)	A(0,0)	A(0,1)
A(1,0)	A(1,1)	A(1,2)
A(2,1)	A(2,2)	A(2,0)

B(2,0)	B(0,1)	B(1,2)
B(0,0)	B(1,1)	B(2,2)
B(1,0)	B(2,1)	B(0,2)

A, B after shifting
k=2

C(1,2) = A(1,0) * B(0,2) + A(1,1) * B(1,2) + A(1,2) * B (2,2)

Source Code modifications

- matrices filled with previously selected test datasets
- transferred matricies to gpu memory at startup
- integrated compression and decompression into matrix exchange between nodes

results obs_spitzer

m_size	512	1024	2048	4096	8192
compute	0.21	1.21	9.41	69.85	534.59
communication	2.74	6.87	24.68	94.21	371.46
total runtime	2.96	8.08	34.09	164.06	906.06

---

m_size	512	1024	2048	4096	8192
compute	0.21	1.16	8.76	68.03	537.87
compression	0.23	0.36	0.80	2.59	9.71
communication	5.79	9.76	27.32	95.5	366.91
total runtime	6.24	11.3	36.9	166.59	914.51

results msg_sppm

m_size	512	1024	2048	4096	8192
compute	0.21	1.21	9.40	70.03	534.60
communication	2.73	7.14	25.06	93.67	370.06
total runtime	2.94	8.36	34.47	163.70	904.67

m_size	512	1024	2048	4096	8192
compute	0.2	1.13	8.72	67.86	537.29
compression	0.21	0.28	0.62	1.9	6.93
communication	4	5.55	14.62	42.31	155.06
total runtime	4.43	6.98	23.98	112.1	699.31

Conclusion and Future

- Benefits of message compression largely dependent on compression ratio of transferred data
- for small messages compression could become prohibitive
- more optimizations possible
- future work:
  - Integration in MPI Implementation
  - adaptive use of compression
  - support for more MPI operations
  - support multiple algorithms

Sources

- FZ Juelich, “Multi GPU Programming with MPI.” https://www.fz-juelich.de/SharedDocs/Downloads/IAS/JSC/EN/slides/msa-seminar/2020-01-21-CUDA-aware-MPI.pdf, 2020.
- Q. Zhou, P. Kousha, Q. Anthony, K. Shafie Khorassani, A. Shafi, H. Subramoni, and D. K. Panda, “Accelerating mpi all-to-all communication with online compression on modern gpu clusters,” in *International Conference on High Performance Computing*, pp. 3–25, Springer, 2022.
- Zhou, Q and Chu, C and Kumar, NS and Kousha, P and Ghazimirsaeed, SM and Subramoni, H and Panda, DK, “Designing high-performance mpi libraries with on-the-fly compression for modern gpu clusters,” in *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 444–453, IEEE, 2021.

source code and slides

github.com/vogma/cannon_cuda_compression