Reduced Communication is *not* All You Need:

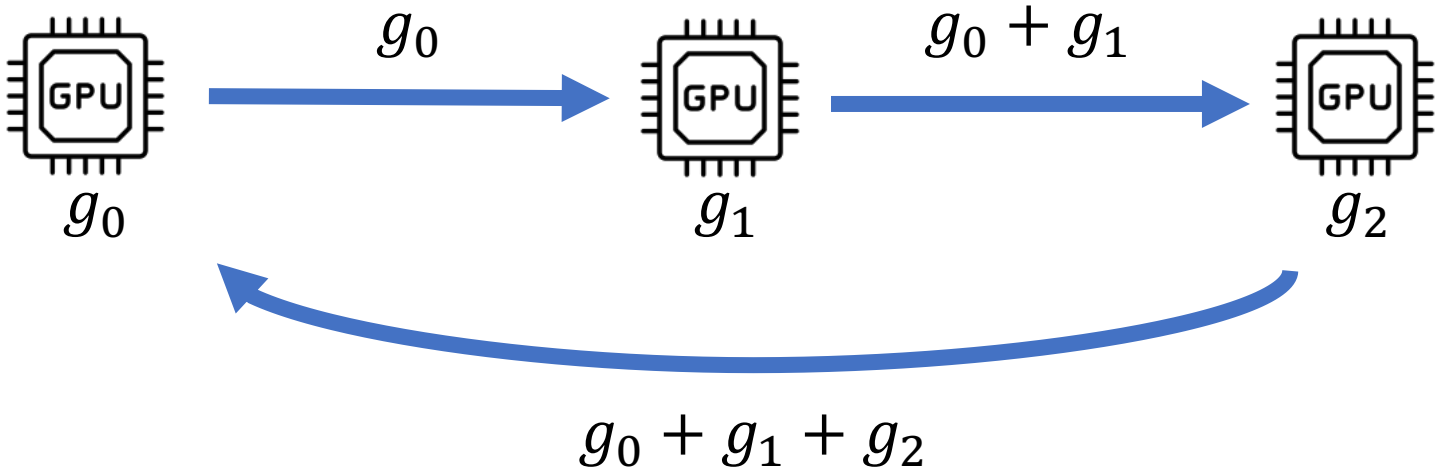
Towards High End-to-end Utility of Gradient Compression

Wenchen Han (UCL), Shay Vargaftik (VMWare Research),

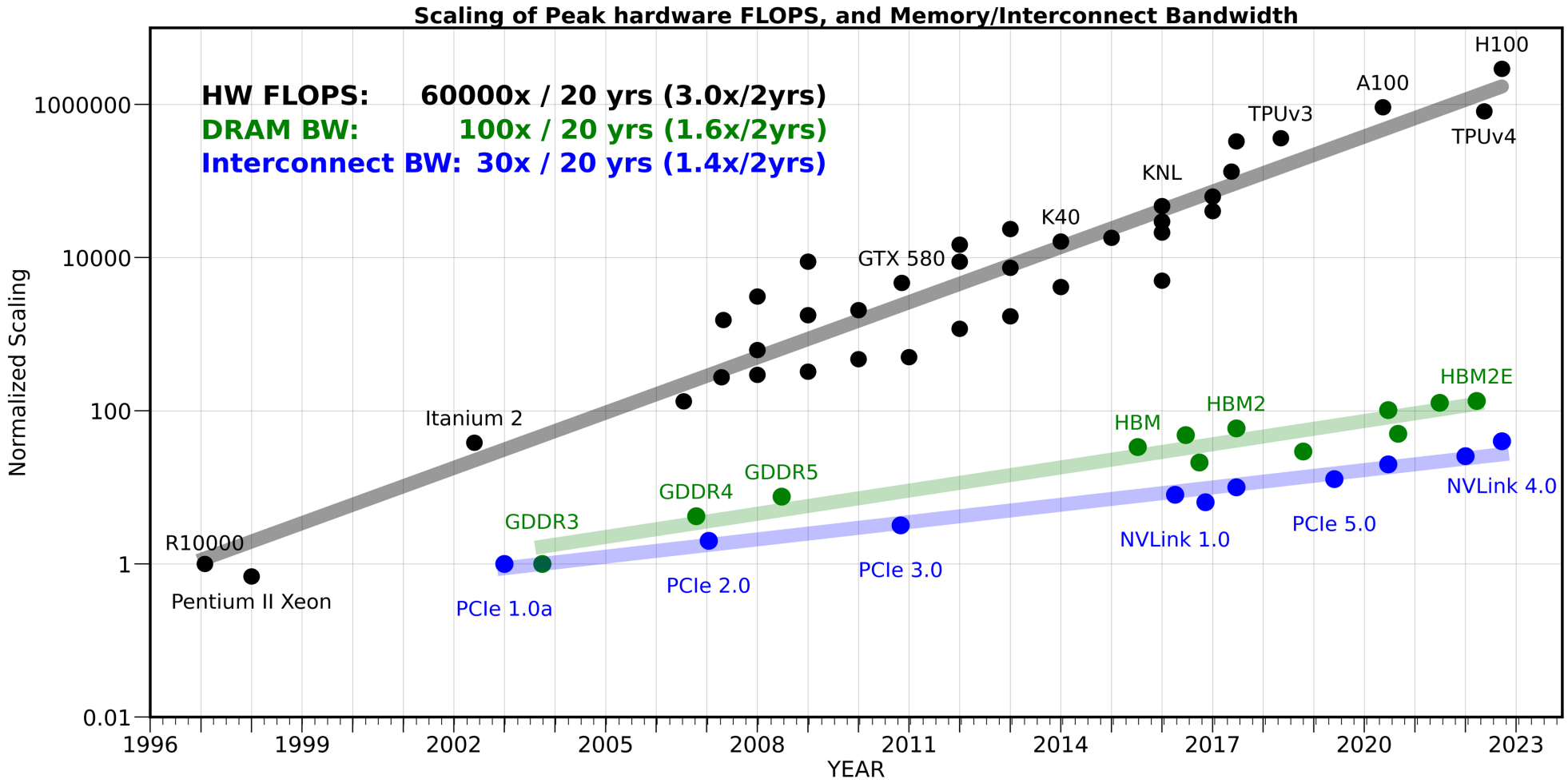
Michael Mitzenmacher (Harvard), Brad Karp (UCL), Ran Ben Basat (UCL)

Distributed Data Parallel Training

- Multiple workers training the same model with different data.
- Periodic synchronization to aggregate the gradients (Δmodel).
- Can be achieved by all-reduce.



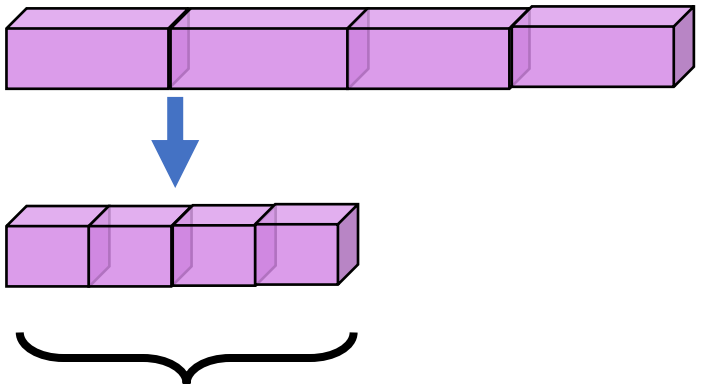
Problem: Communication Bottleneck



[1]

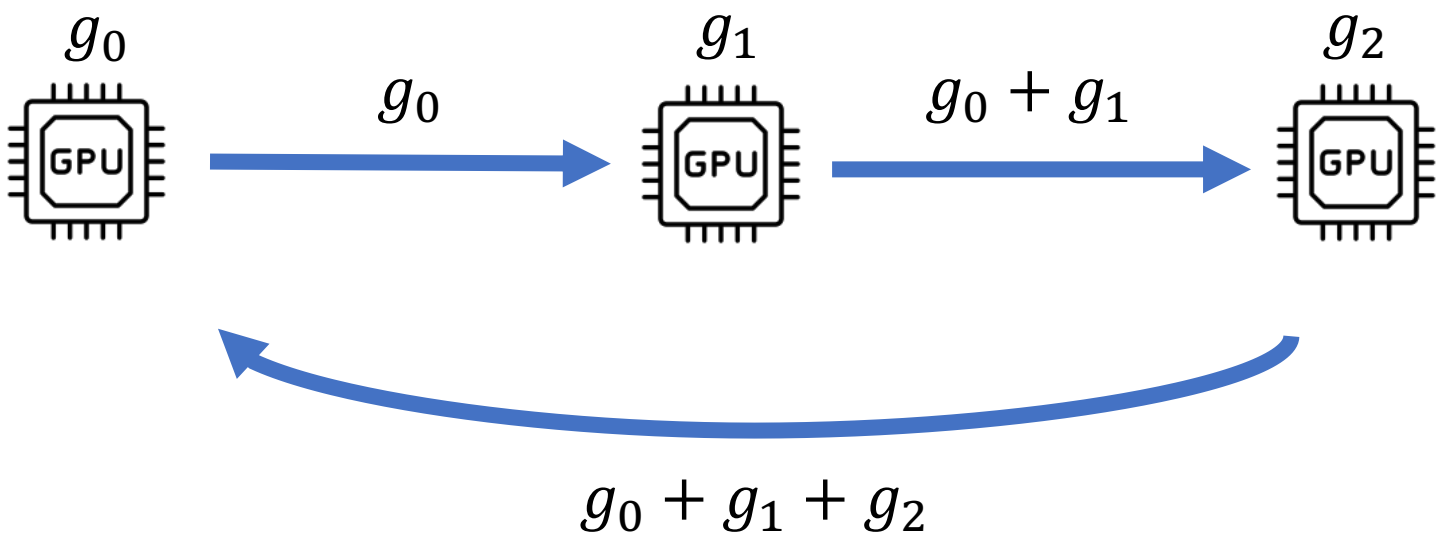
[1] https://github.com/amirgholami/ai_and_memory_wall

Gradient Compression Comes to Rescue



1 slot = 1 floating point (FP32) $= 32$ bits

1 slot = 1 half precision (FP16) $= 16$ bits



Is Reduced Communication ALL We Need?

- Throughput: speed to execute a round (in rounds per second).
- End-to-end utility: how much time to reach a target accuracy.

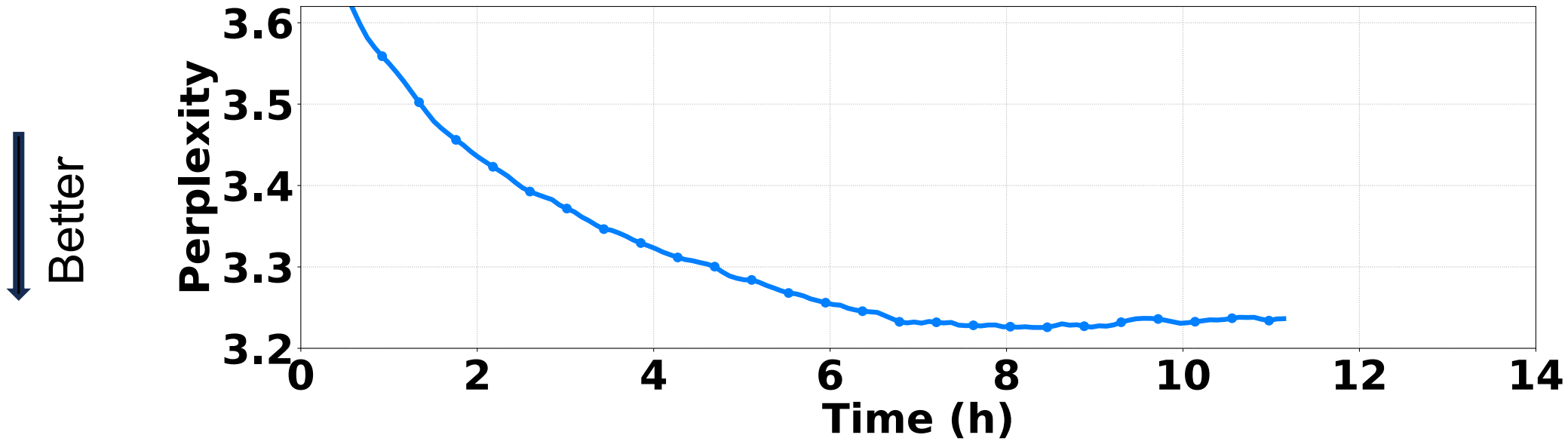
Method	Reduced communication	Throughput speedup	End-to-end speedup
TopK [2] compression	87.5%	17%	-65%
Theoretical upper bound	100%	83%	83%

What Else Do We Need?

1. Choosing the correct optimization goal.

Choosing the correct optimization goal.

- End-to-end: time to reach a target accuracy: TTA.

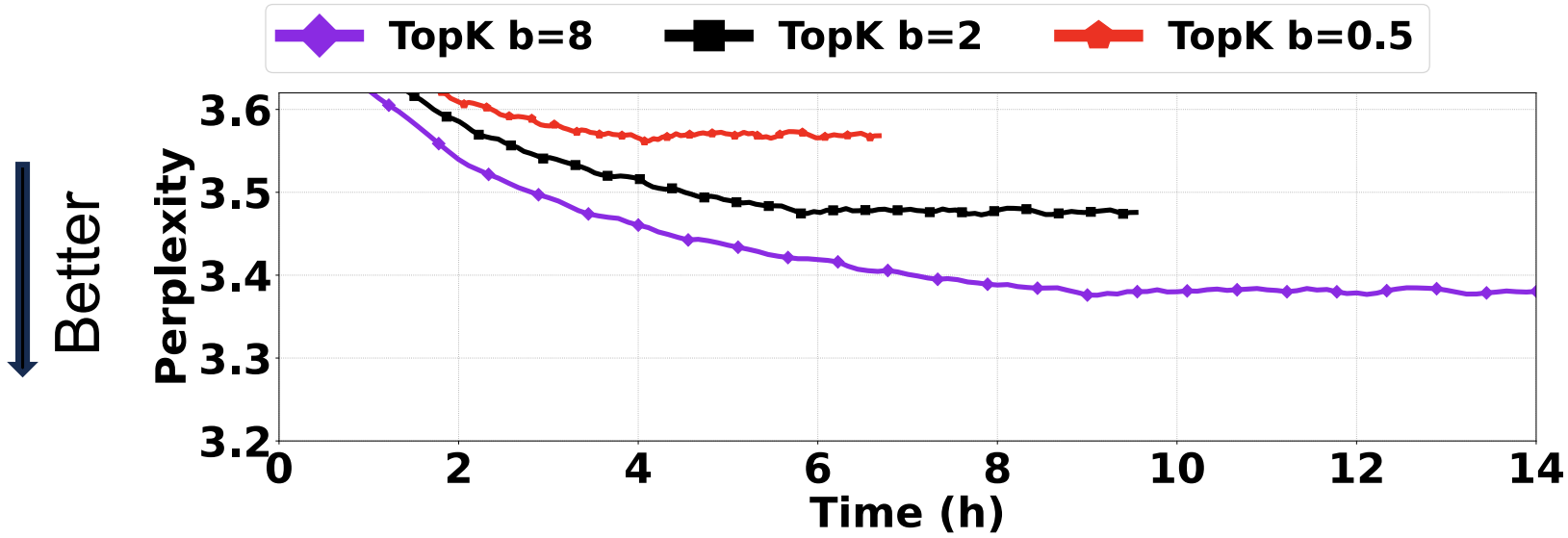


Slower Throughput, Better Utility.

- Compression error matters.

	b=0.5 *	b=2	b=8
Throughput	5.53	3.87	2.5

Worse



*Lower b (bits per coordinate), higher compression ratio.

What Else Do We Need?

2. Better designs.

- Reducing compression overhead.
- Compatibility to all-reduce.

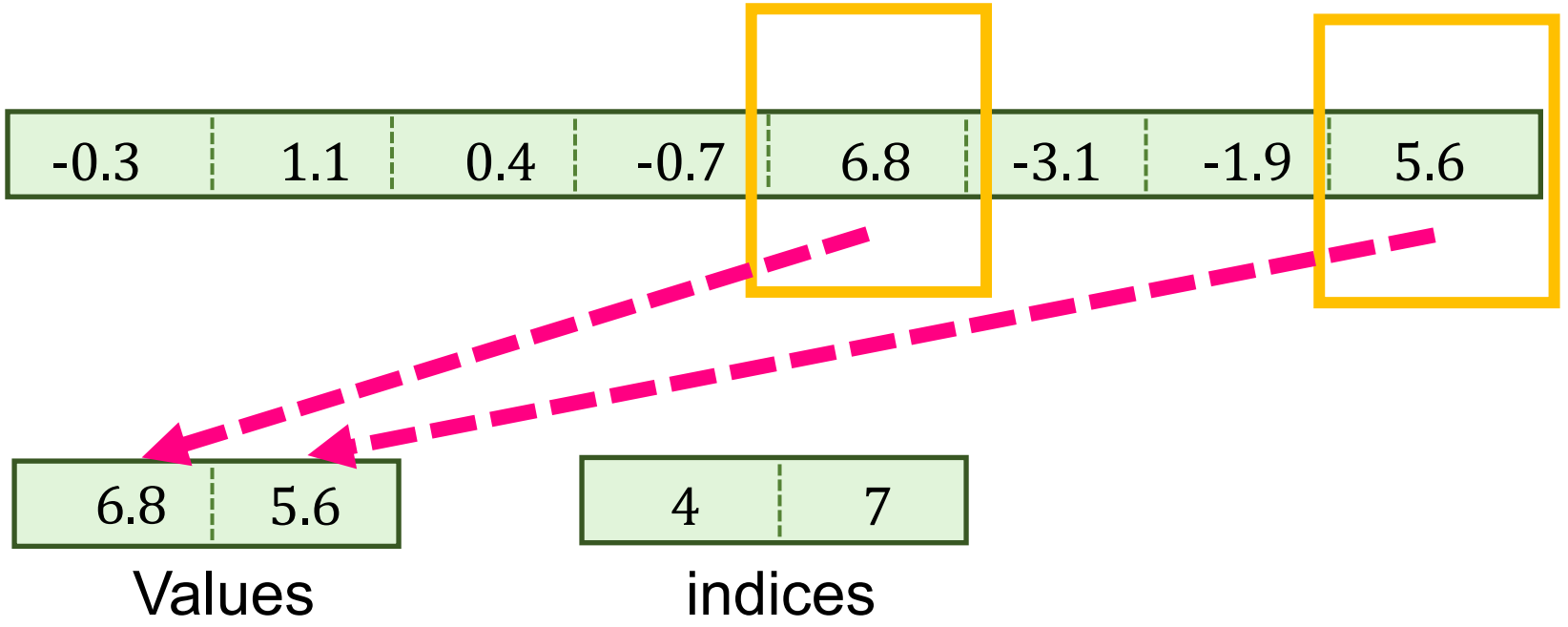
(1). High Compression Overhead

- Compression overhead: the proportion of time it takes for the computation of compression.

TopK	$b=2$
Compression overhead	12.5%

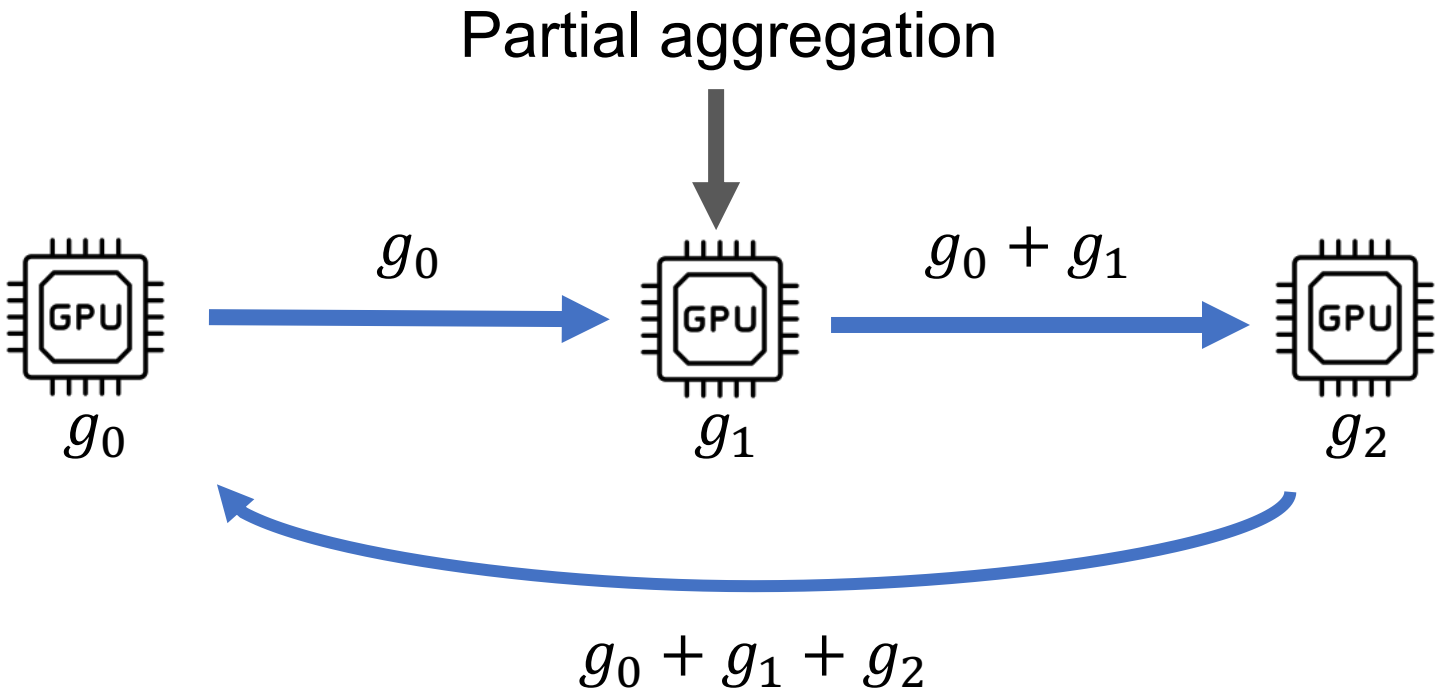
(1). High Compression Overhead

- Inefficient TopK selection [3].
- Inefficient random memory access.



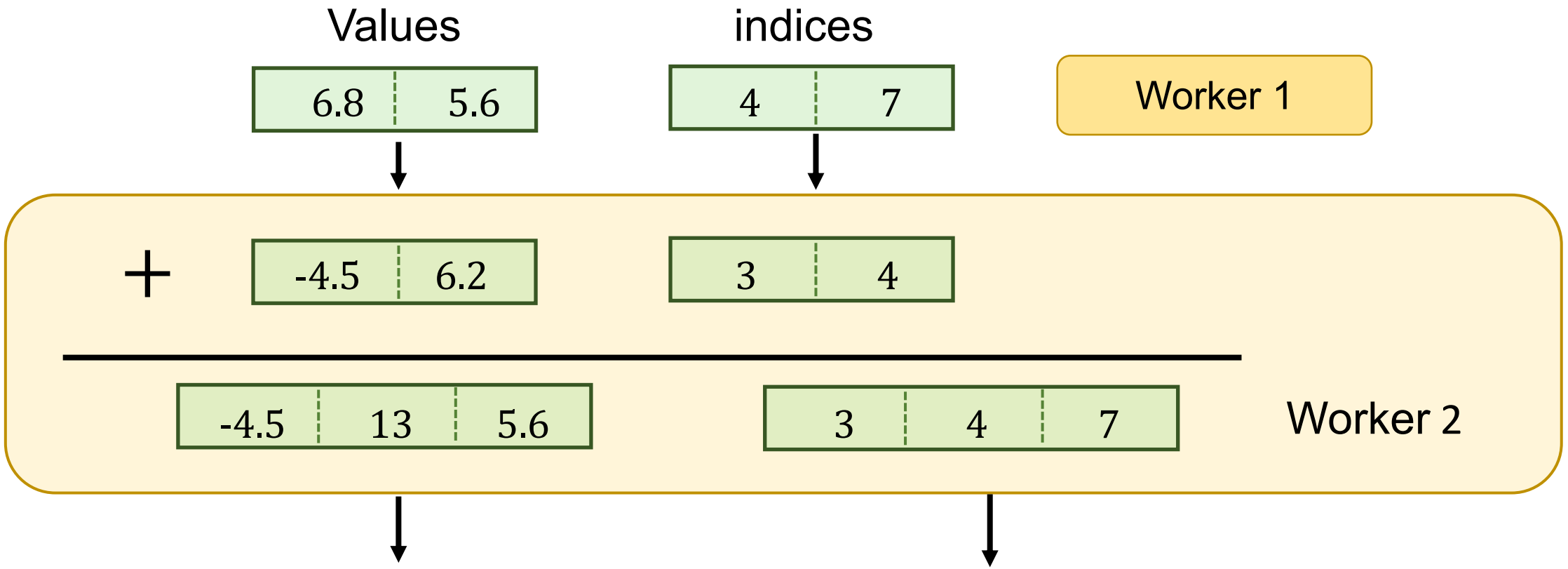
(2). Incompatibility to All-reduce

- Problem: partial aggregation without increased communication.

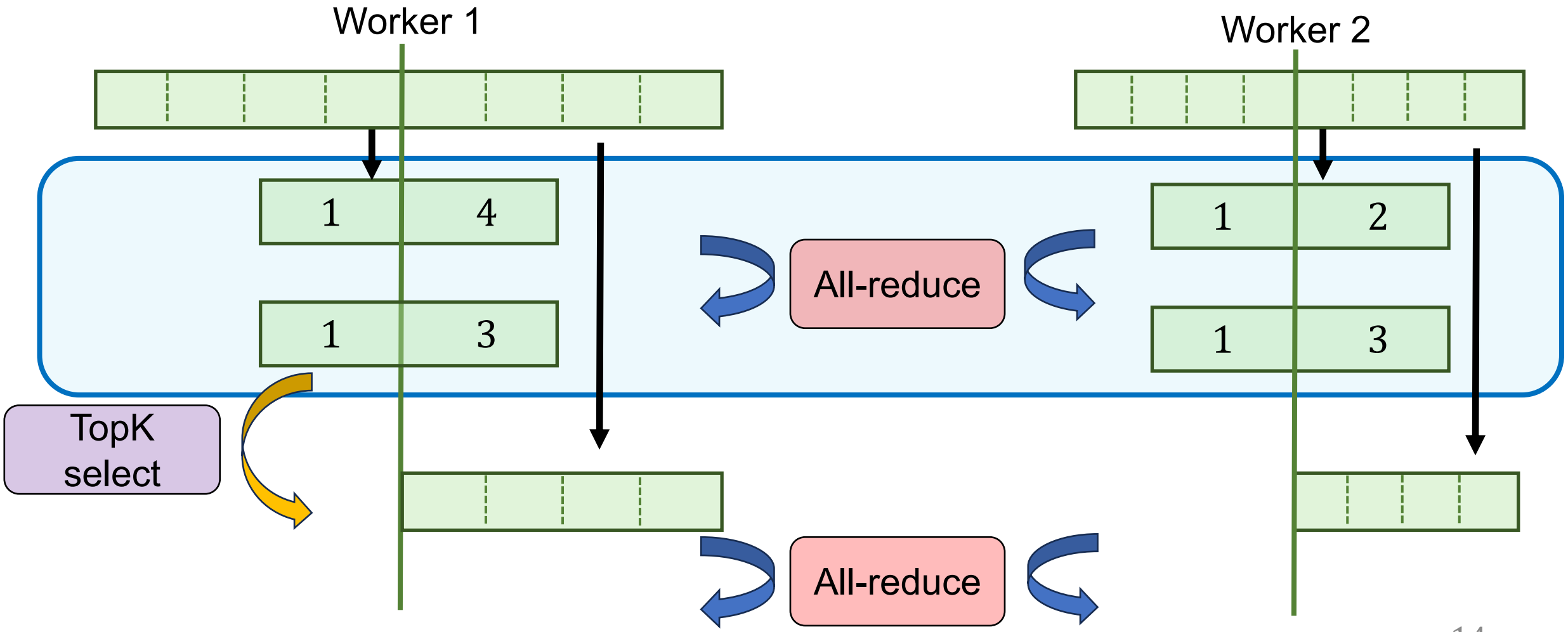


(2). Incompatibility to All-reduce

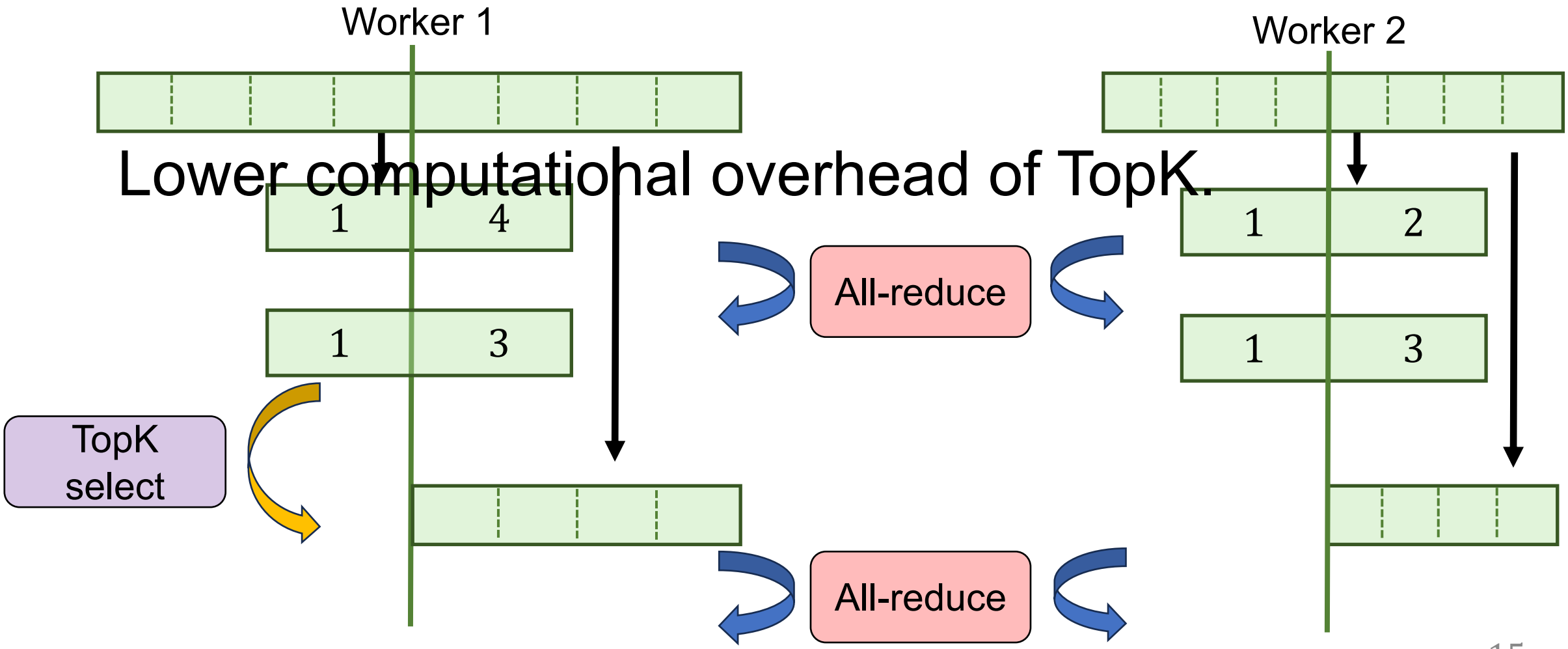
Bandwidth budget: 4 slots.



Our Solution: Chunked TopK (TopKC)



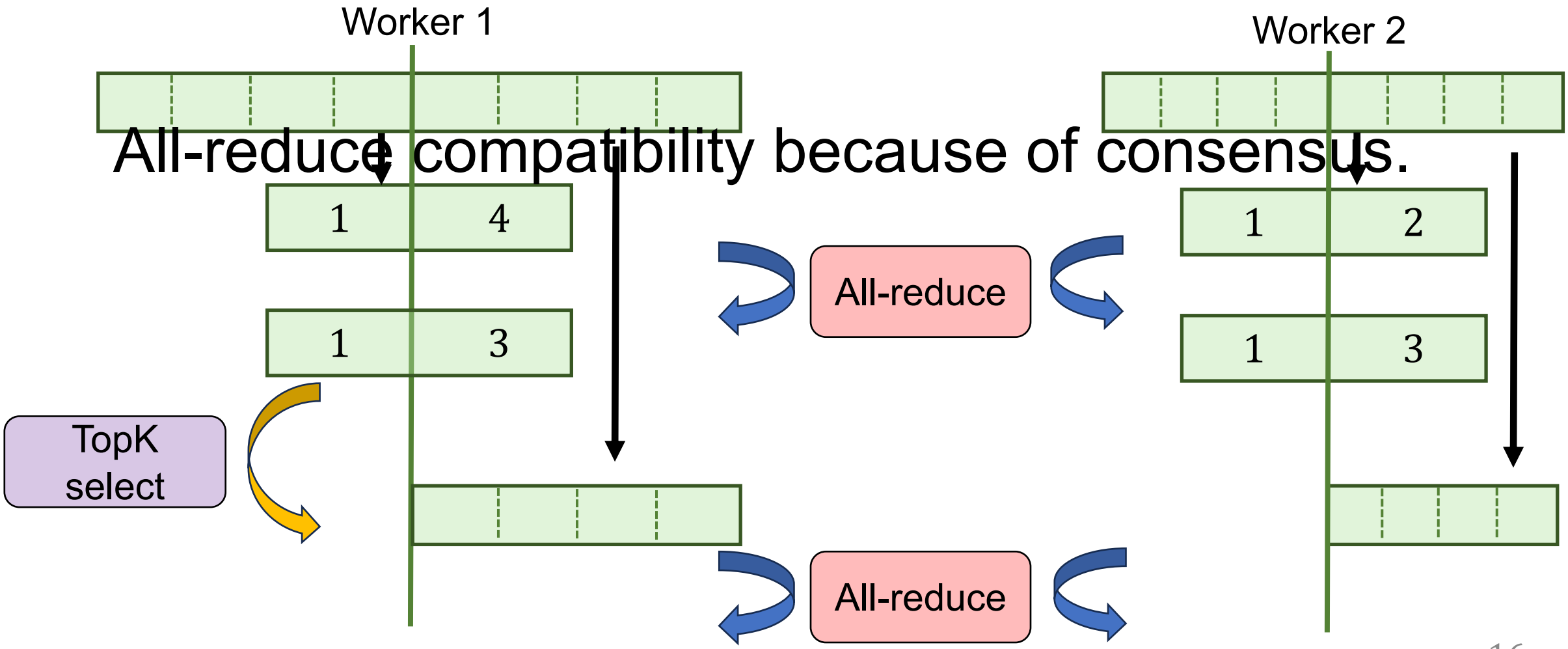
Our Solution: Chunked TopK (TopKC)



Lower computational overhead of TopK.

Our Solution: Chunked TopK (TopKC)

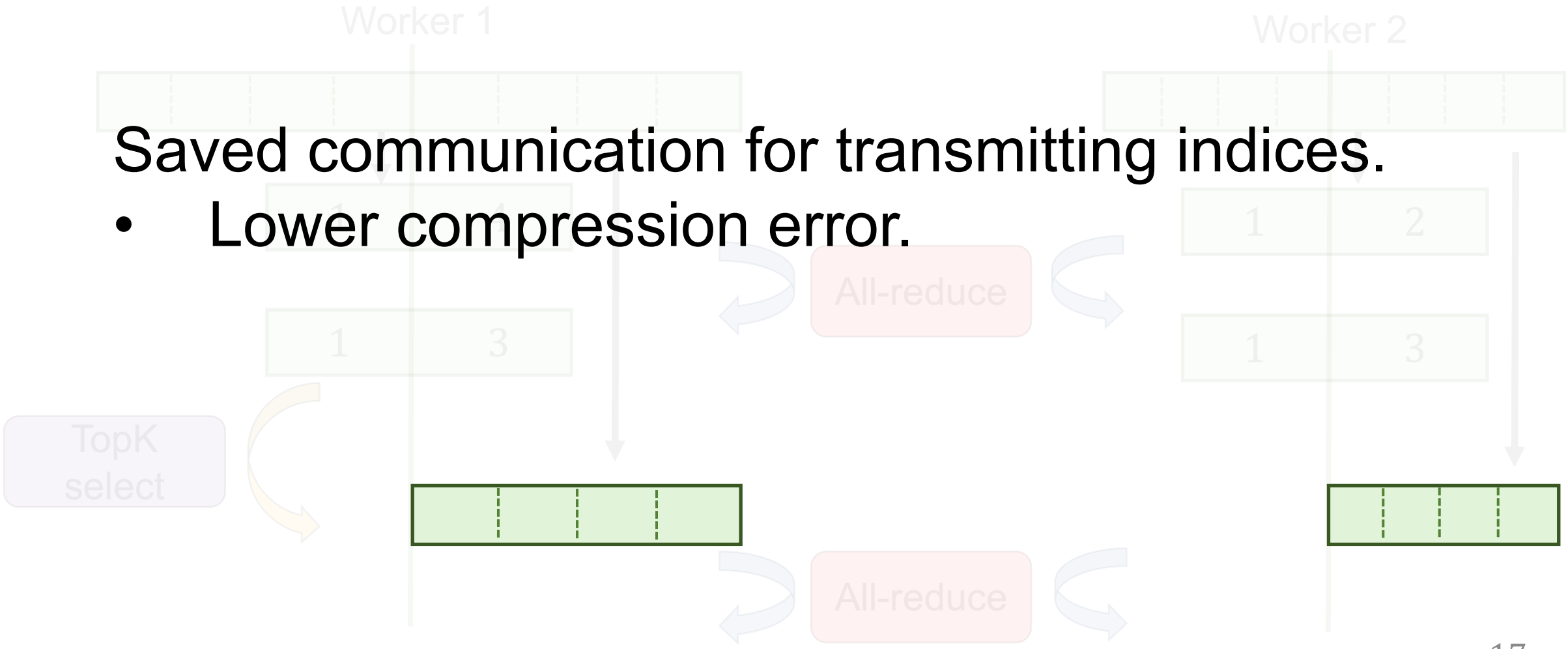
All-reduce compatibility because of consensus.



Our Solution: Chunked TopK (TopKC)

Saved communication for transmitting indices.

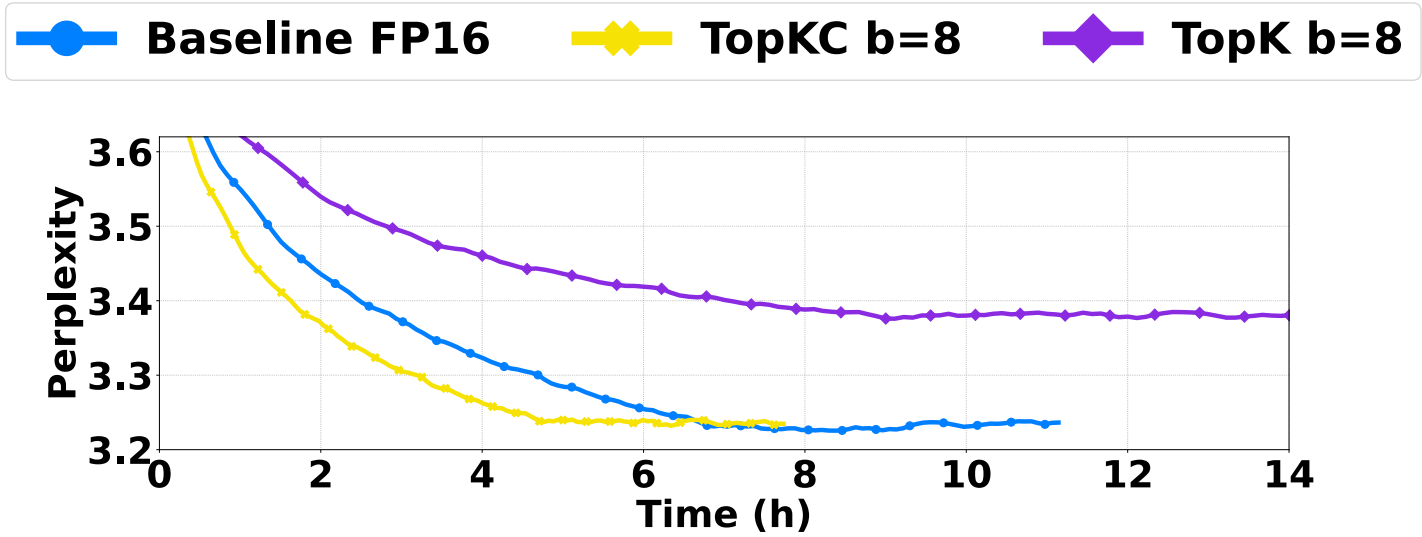
- Lower compression error.



Experimental Results

- Less compression overhead ✅
- All-reduce avoiding many-to-one communication ✅
- Lower compression error ✅.

Method	Compression error
TopK (b=8)	8.65%
TopKC (b=8)	2.80%



Conclusions

- What we need beyond reduced communication?
  - Setting the end-to-end TTA as the optimization goal.
  - Higher throughput by optimizing expensive components.
  - Compatibility to all-reduce.

Our paper: https://arxiv.org/pdf/2407.01378

My email: wenchen.han.22@ucl.ac.uk

References

[1] Memory Footprint and FLOPs for SOTA Models in CV/NLP/Speech. https://github.com/amirgholami/ai_and_memory_wall
[2] Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. 2018. Sparsified SGD with memory. Advances in neural information processing systems 31 (2018)
[3] Anil Shanbhag, Holger Pirk, and Samuel Madden. 2018. Efficient top-k query processing on massively parallel hardware. In Proceedings of the 2018 International Conference on Management of Data. 1557–1570