Systems for Machine Learning (CMSC828G)

Long context in LLMs

Abhinav Bhatele, Department of Computer Science

UNIVERSITY OF MARYLAND

Questions

- What are the longest sequence lengths you have used?
- What is your specific use-case?

Many tasks require long context

- Understanding and generating code
- Summarizing large documents
- Long-form question answering

- Longer context can also improve ML performance
- Users want to try more complex tasks with LLMs everyday

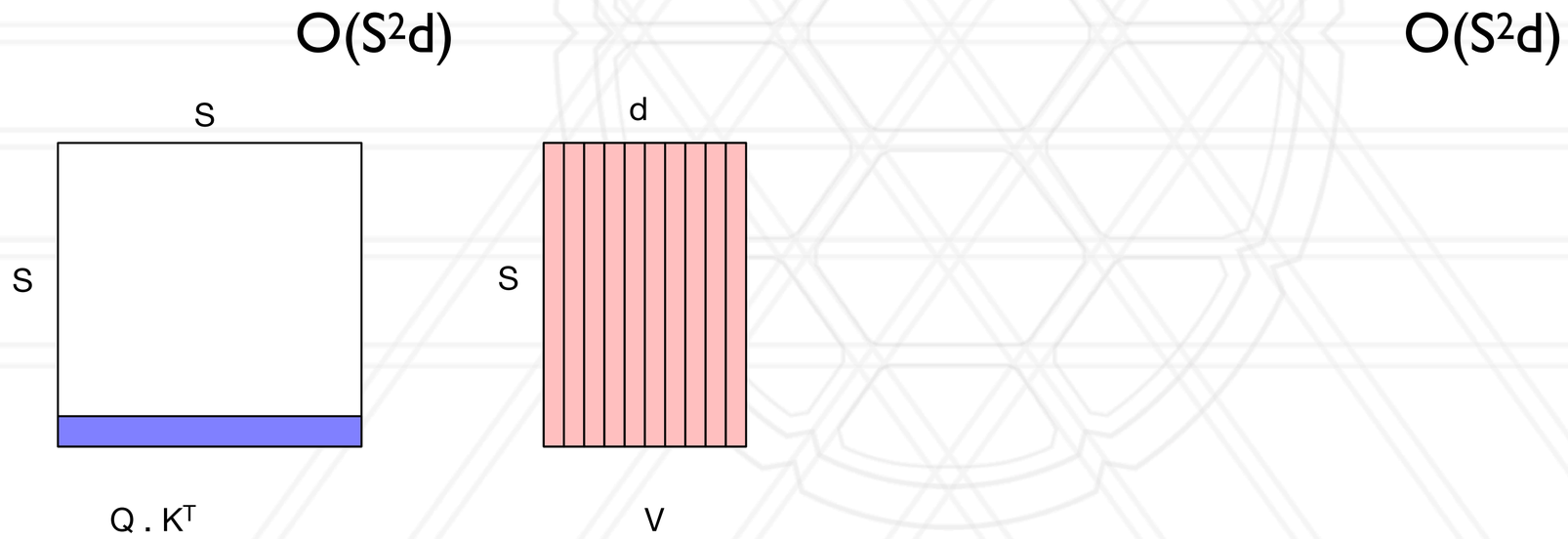
Challenges with long sequences

- Quadratic scaling in attention
- Both for compute and memory

$O(S^2d)$ $O(S^2d)$

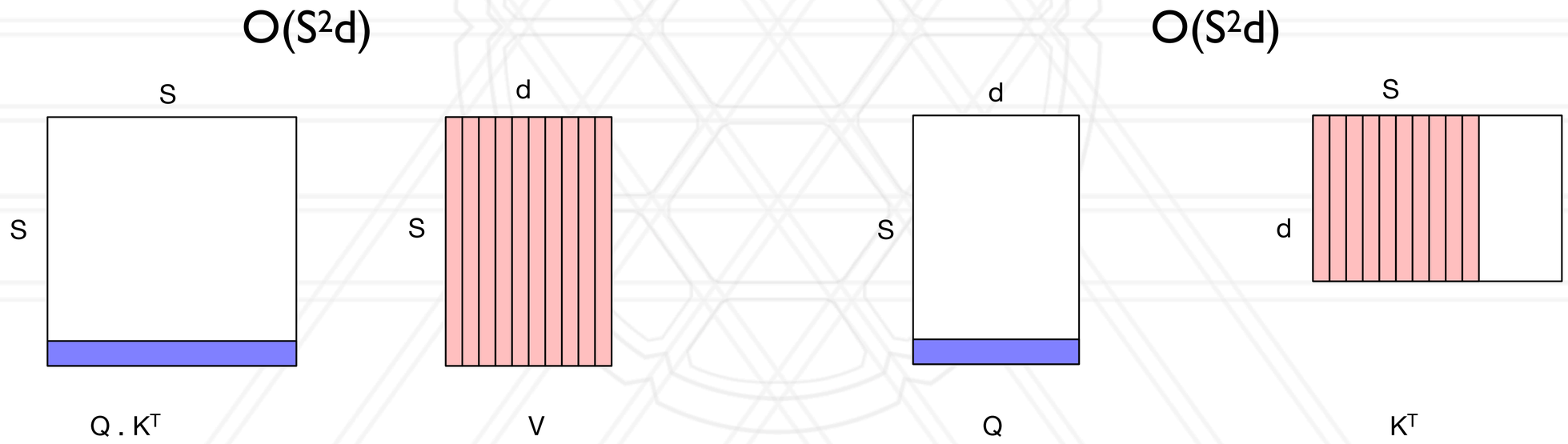
Challenges with long sequences

- Quadratic scaling in attention
- Both for compute and memory



Challenges with long sequences

- Quadratic scaling in attention
- Both for compute and memory



Systems challenges

- GPU memory limits batch size and sequence length
- Larger sequence lengths increase number of flops required
- Leads to larger messages on the network
- More data movement in memory (larger matrices), network, and I/O (datasets, checkpoints)

Solutions

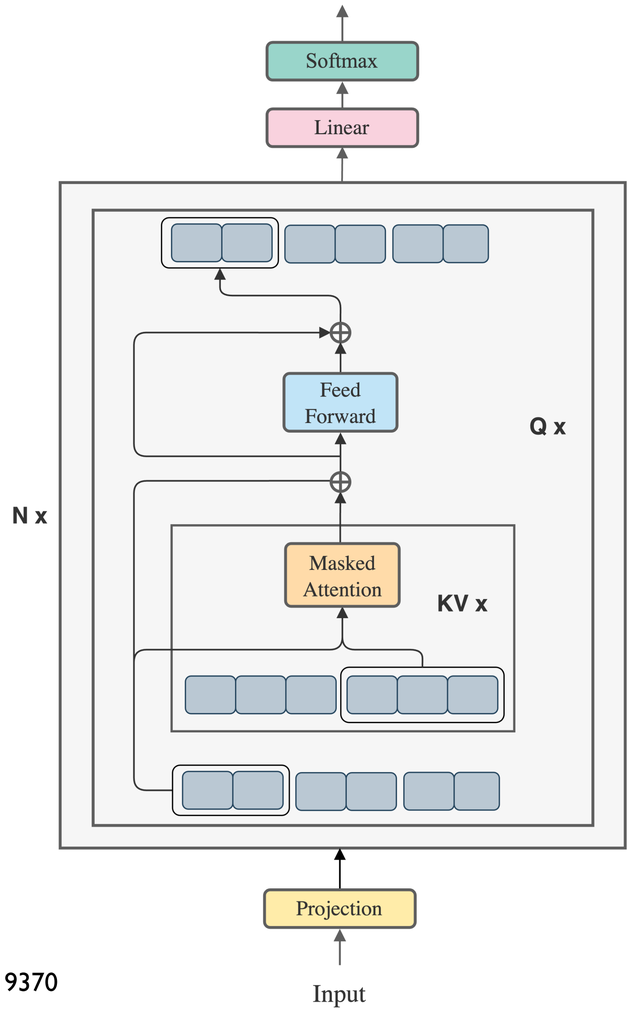
- Memory optimizations: activation checkpointing, ZeRO-style memory optimizations
- Low-rank approximations

$$A \approx BC^T$$

- Approximate / sparse attention: H_2O, Top-K
- Separate category: parallelizing attention

Blockwise Parallel Transformer

- Compute attention "block-wise" to reduce memory costs
- Eliminates the need to materialize the full attention matrix
- Enables training with much longer sequences



https://arxiv.org/abs/2305.19370

Blockwise Parallel Transformer

$$\text{Output}_i = \text{FFN}\big(\text{Attention}(Q_i, K, V) + Q_i\big) + \text{Attention}(Q_i, K, V) + Q_i.$$

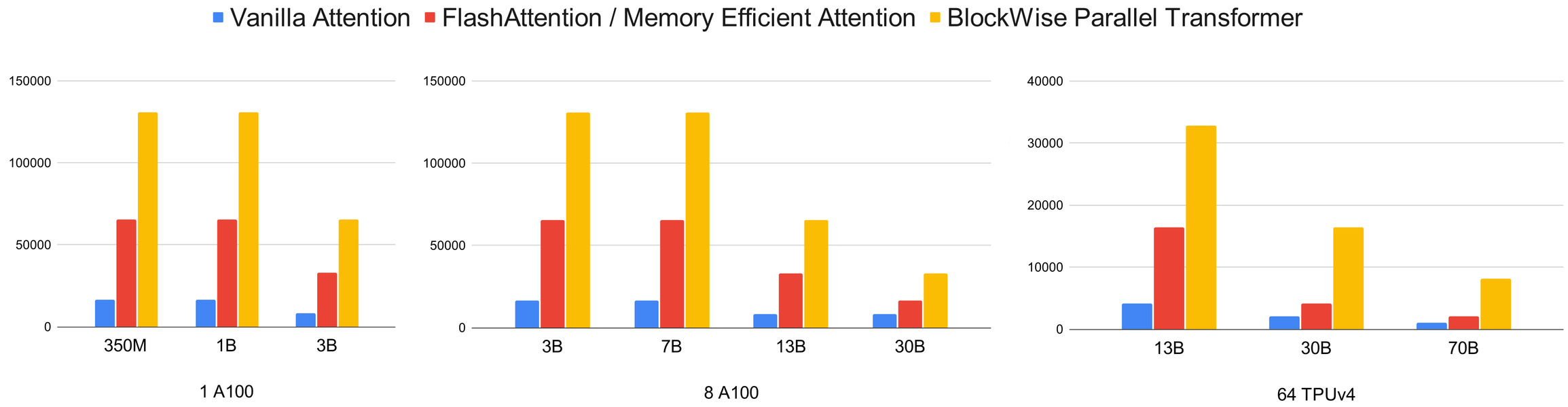
Algorithm 1 Reduce memory cost with BPT.

Required: Input sequence x. Number of query blocks B_q. Number of key and value blocks B_{kv}.
Initialize
Project input sequence x into query, key and value.
Split query sequence into B_q of query input blocks.
Split key and value sequences into B_{kv} of key-value input blocks.
for $outer = 1$ **to** B_q **do**
  Choose the $outer$-th query.
  for $inner = 1$ **to** B_{kv} **do**
    Choose the $inner$-th key and $inner$-th value block.
    Compute attention using query, key and value, and record normalization statistics.
  end for
  Combine each blocks by scaling them to get attention output for the $outer$-th input block.
  Compute feedforward on attention output and add residual connection.
end for

https://arxiv.org/abs/2305.19370

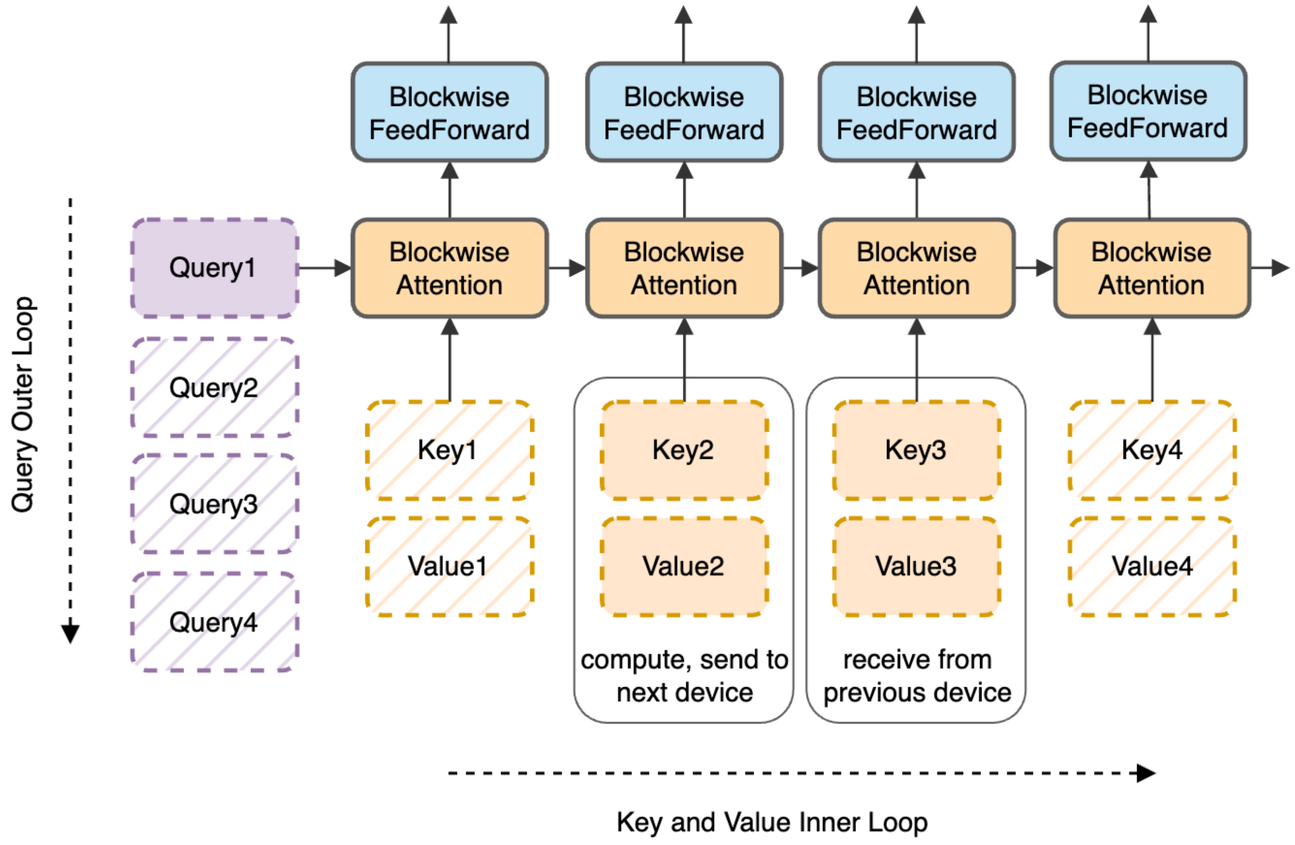
Sequence lengths supported

1 A100	PartitionSpec	Vanilla Attention	MemoryEfficient	Blockwise Parallel
350M	(1,1,1)	16K (16384)	65K (65536)	131K (131072)
1B	(1,1,1)	16K (16384)	65K (65536)	131K (131072)
3B	(1,1,1)	8K (8192)	16K (16384)	65K (65536)
8 A100	**PartitionSpec**	**Vanilla Attention**	**MemoryEfficient**	**Blockwise Parallel**
3B	(1,1,8)	16K (16384)	65K (65536)	131K (131072)
7B	(1,1,8)	16K (16384)	65K (65536)	131K (131072)
13B	(1,1,8)	8K (8192)	33K (32768)	65K (65536)
30B	(1,1,8)	8K (8192)	16K (16384)	65K (65536)
64 TPUv4	**PartitionSpec**	**Vanilla Attention**	**MemoryEfficient**	**Blockwise Parallel**
13B	(1,1,64)	4K (4096)	16K (16384)	33K (32768)
30B	(1,1,64)	2K (2048)	4K (4096)	16K (16384)
70B	(1,1,64)	1k (1024)	2K (2048)	8K (8192)



Ring Attention with BPT

- Parallelize attention across multiple GPUs
- Arrange GPUs into a ring
- Split input sequence into blocks



Ring Attention with BPT

Algorithm 1 Reducing Transformers Memory Cost with Ring Attention.

Required: Input sequence x. Number of hosts N_h.
Initialize
Split input sequence into N_h blocks that each host has one input block.
Compute query, key, and value for its input block on each host.
for Each transformer layer **do**
  for $count = 1$ **to** $N_h - 1$ **do**
    for For each host concurrently. **do**
      Compute memory efficient attention incrementally using local query, key, value blocks.
      Send key and value blocks to next host and receive key and value blocks from previous host.
    end for
  end for
  for For each host concurrently. **do**
    Compute memory efficient feedforward using local attention output.
  end for
end for

Max context length supported

	Max context size supported ($\times$1e3)				
	Vanilla	Memory Efficient Attn	Memory Efficient Attn and FFN	Ring Attention (Ours)	Ours vs SOTA
8x A100 NVLink					
3B	4	32	64	**512**	8x
7B	2	16	32	**256**	8x
13B	2	4	16	**128**	8x
32x A100 InfiniBand					
7B	4	64	128	**4096**	32x
13B	4	32	64	**2048**	32x

UNIVERSITY OF
MARYLAND

Abhinav Bhatele

5218 Brendan Iribe Center (IRB) / College Park, MD 20742

phone: 301.405.4507 / e-mail: bhatele@cs.umd.edu