Training Large Language Models

How to reduce memory and compute

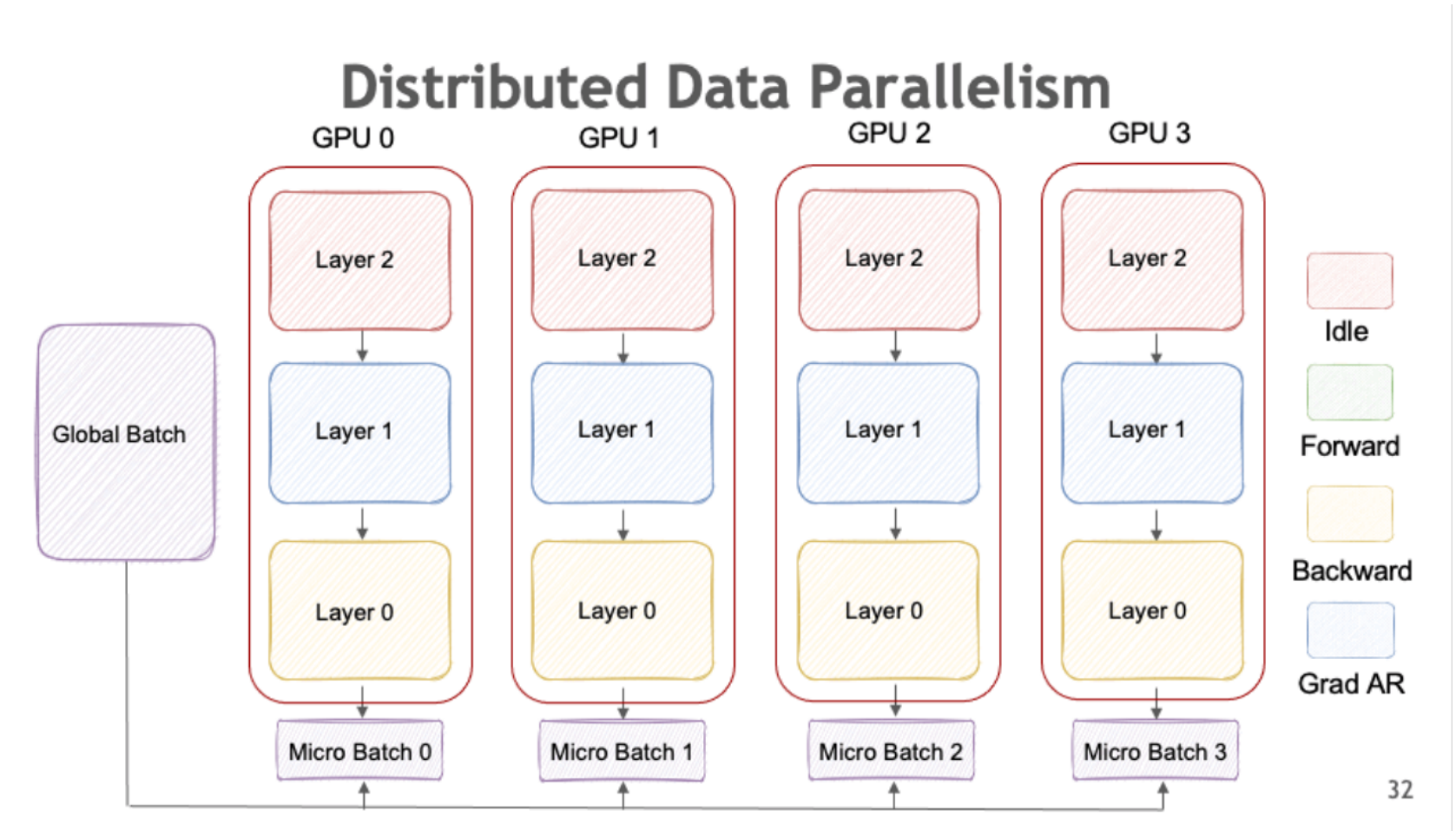
Learning goals

- Learn about different techniques to reduce compute and memory
- Learn about distributed training with data/tensor parallelism
- Learn about Flash Attention

DISTRIBUTED TRAINING

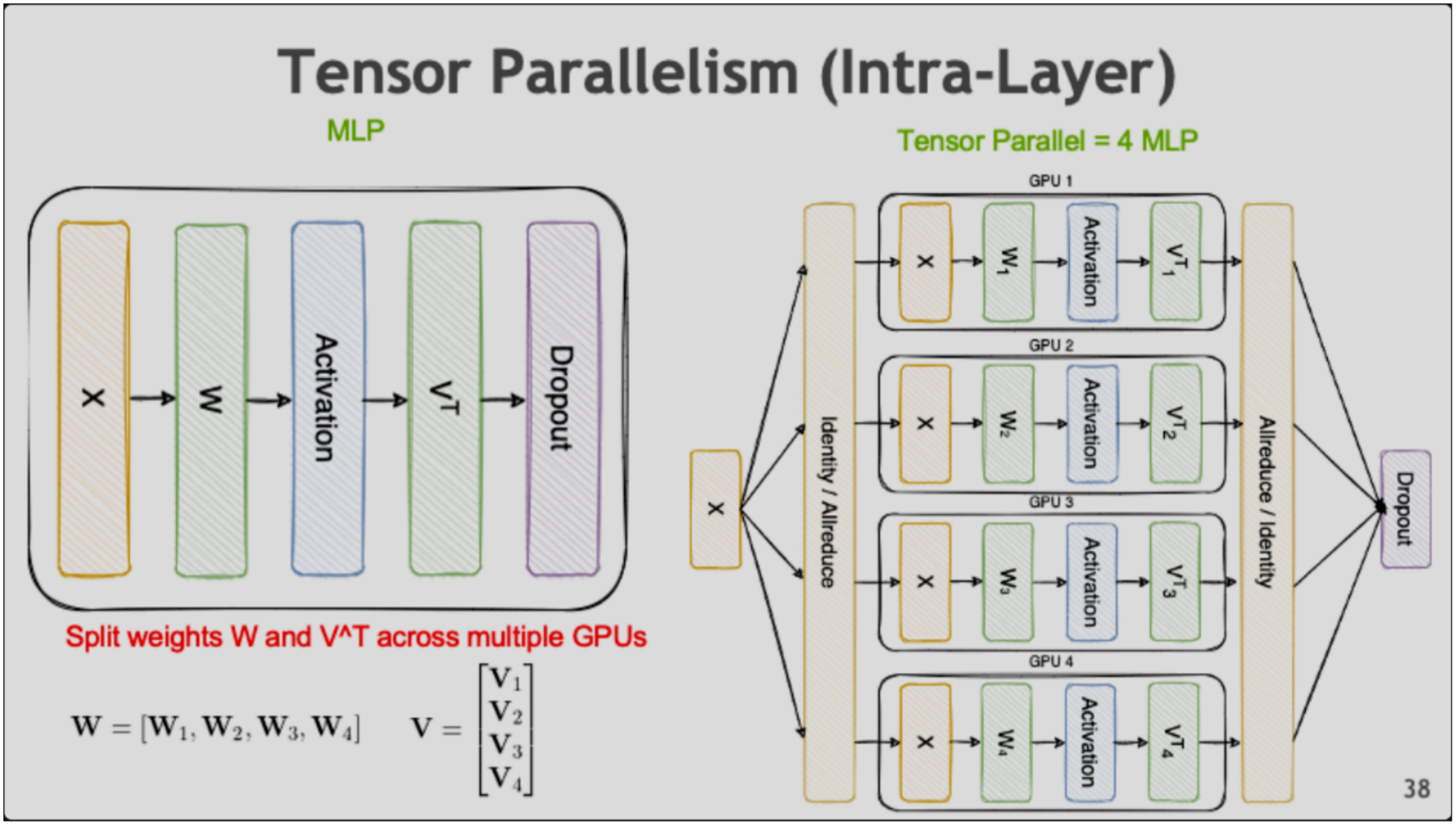
- Training LLMs faster on many GPUs
- Avoiding OOM issues
- **Data parallelism:** split the data on different model replicas
- **Tensor parallellism:** split model parameters accross GPUs

DATA PARALELISM (ANIMATED GIF!)



Source: Nvidia

TENSOR PARALELISM (ANIMATED GIF!)



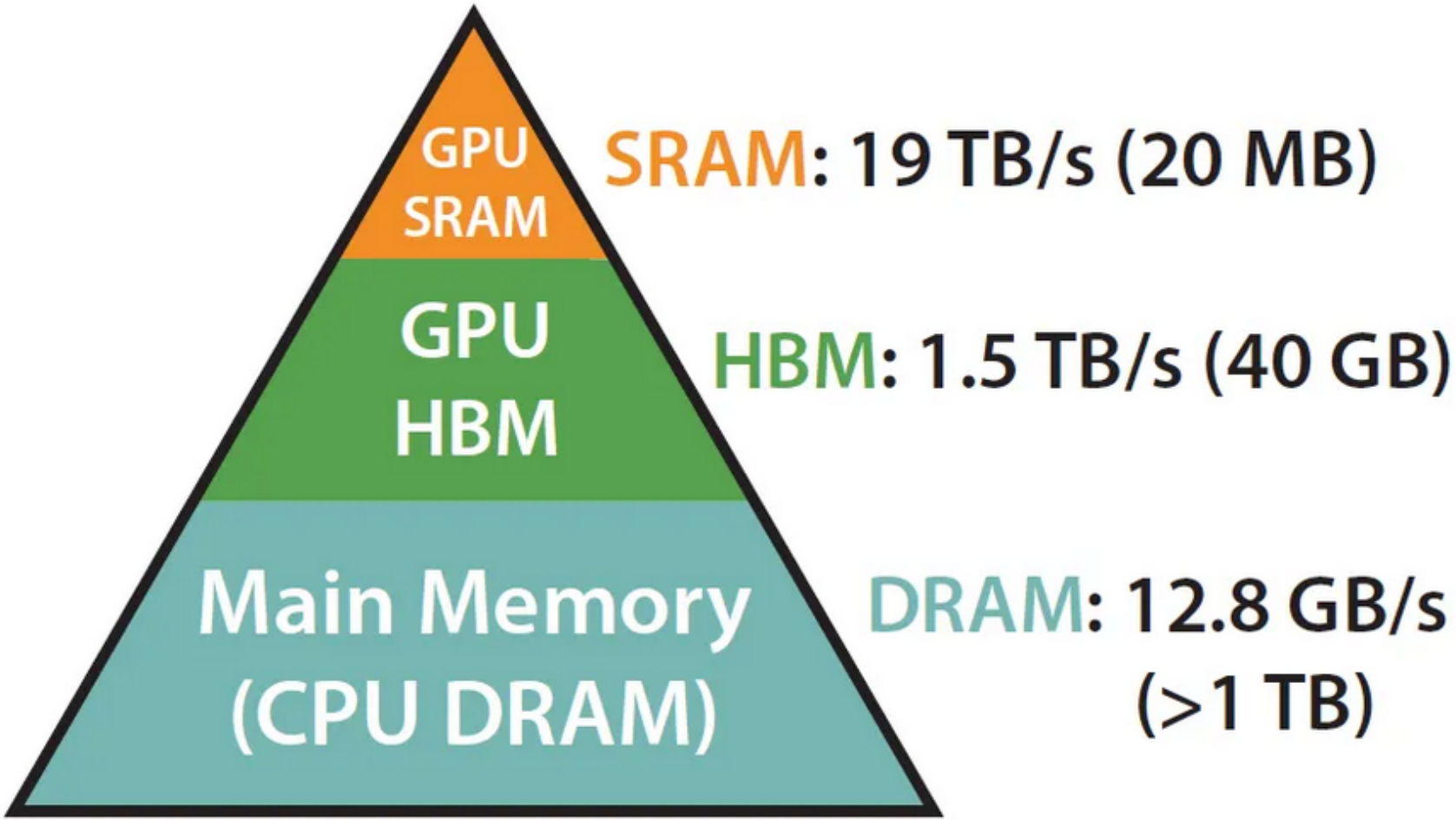
Source: Nvidia

FlashAttention

Fast and Memory-Efficient Exact Attention with IO-Awareness

- Fast
  - 15 % faster than BERT
  - 3x faster than GPT-2
  - 2.4x faster than Megatron-LM
- Memory-efficient
  - Reducing from $O(n^2)$ to $O(n)$
- Exact
  - Same as “vanilla attention”, not an approximation
- IO aware
  - Reducing memory load/store operations

GPU MEMORY HIERARCHY

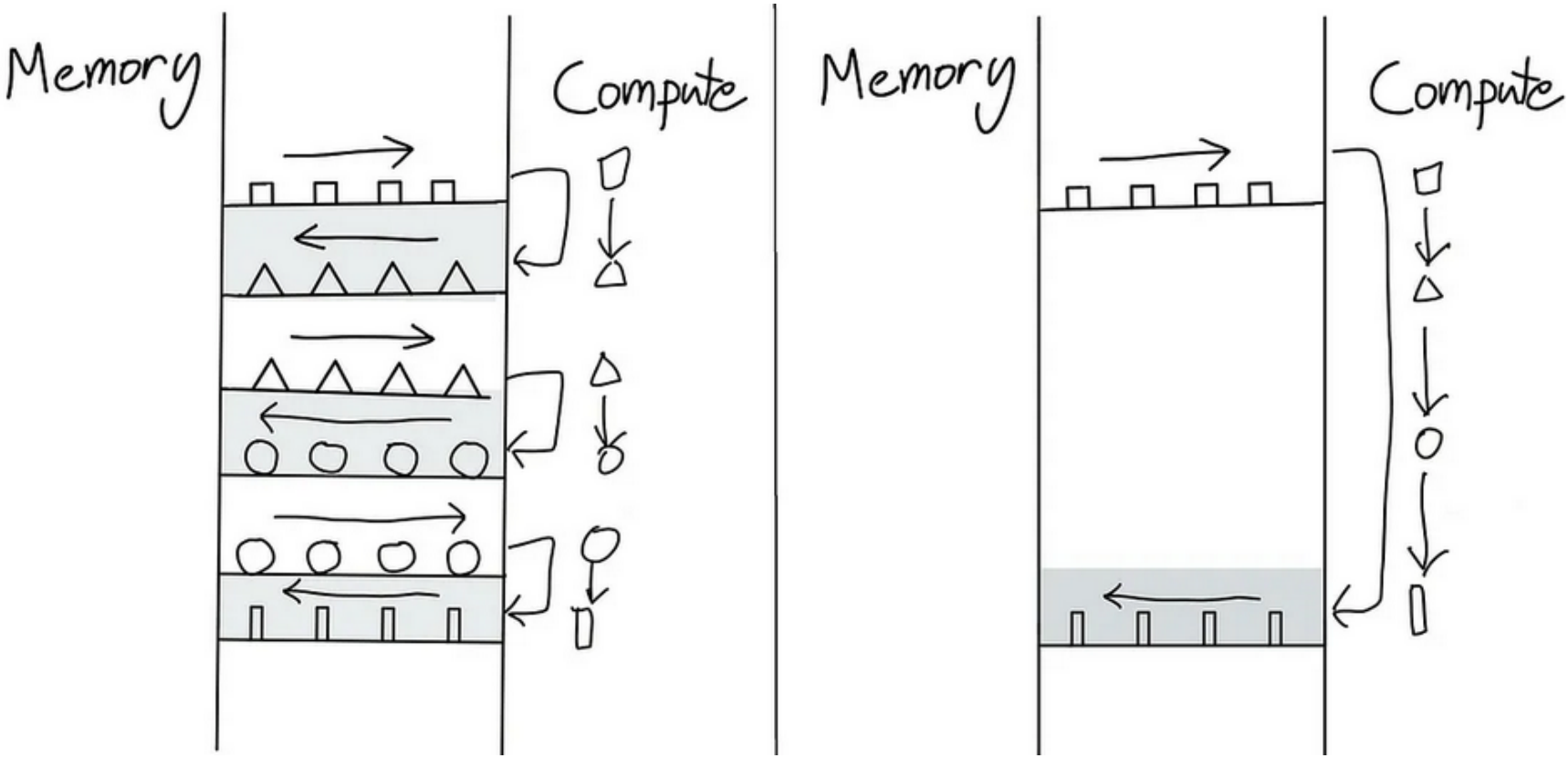


Source: Dao et al. (2022)

COMPUTING CONSIDERATIONS

- GPU compute has been growing faster than memory bandwidth
  - GPU has to wait for data
- Transformer operations are memory-bound
  - Elementwise operations with high memory access
- IO aware means reducing memory load/store operations
- FlashAttention implements the following:
  - Operation fusion to reduce memory access
  - Tiling or chunking the softmax matrix into blocks
  - Recomputation for better memory utilization

OPERATION FUSION



Source: `https://horace.io/brrr_intro.html`

LIMITATIONS AND PROSPECTS

- FlashAttention requires writing attention to CUDA language
  - A new CUDA kernel for each new attention implementation
  - CUDA is lower-level than PyTorch
  - Implementation may not be transferable accross GPUs
- Towards IO-Aware Deep Learning
  - Extending beyonde attention
- Multi-GPU IO-Aware Methods
  - FlashAttention computation may be parallelizable accross multiple GPUs