Zorse: Optimizing LLM Training Efficiency on Heterogeneous GPU Clusters

Runsheng (Benson) Guo[1], Utkarsh Anand[1], Khuzaima Daudjee[1], Rathijit Sen[2]
[1]University of Waterloo, [2]Microsoft GSL

MLSys, May 20 2026

Homogeneous Compute

- **Problem**: Most training systems are optimized for homogeneous GPU clusters, but large homogeneous clusters are largely inaccessible to most users

Homogeneous Compute

- **Problem**: Most training systems are optimized for homogeneous GPU clusters, but large homogeneous clusters are largely inaccessible to most users
  - ➢ GPU availability on the Cloud is limited
  - ➢ New generations of hardware released every year
    - ■ Costly to replace whole cluster

Heterogeneous Training

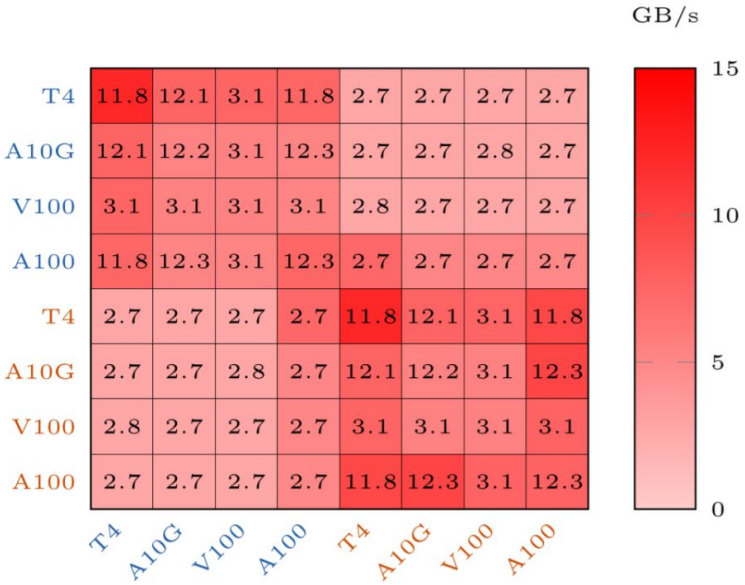
- **Research Problem**: *How can we design a system for efficient training on heterogeneous clusters?*
- **Target Customer***:*
  - Organizations with insufficient high-end homogeneous GPU compute, but have access to larger pools of heterogeneous compute

Network Heterogeneity

- Large amount of network heterogeneity on the cloud
  - Differences in:
    - Inter vs intra node networking
    - Inter vs intra vm types networking
    - Inter vs intra datacenter networking

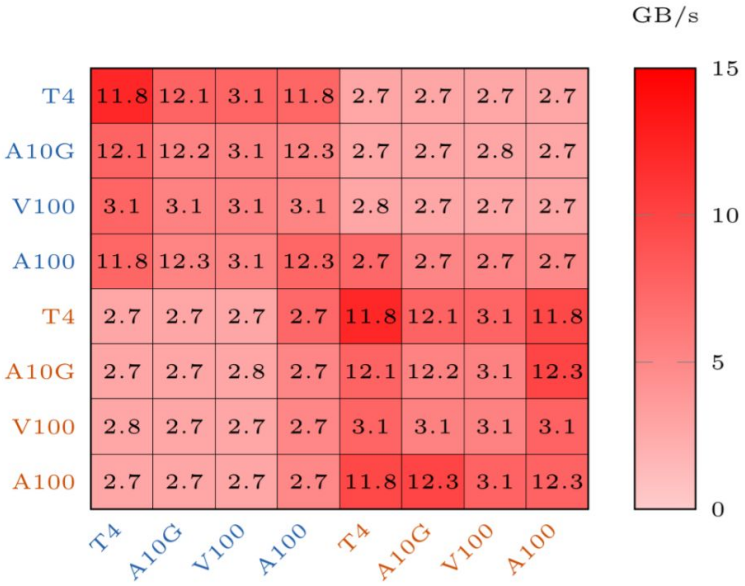
Network Heterogeneity

Inter-VM Bandwidth

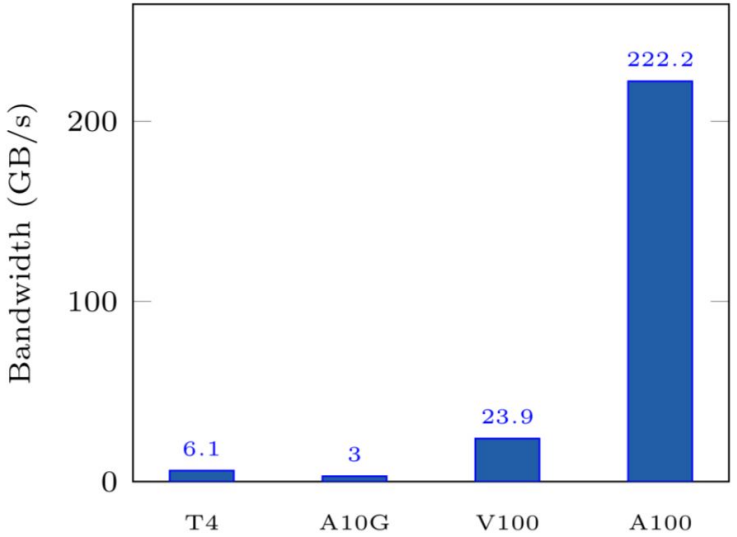


Network Heterogeneity

Inter-VM Bandwidth



Intra-VM Bandwidth



Network Heterogeneity

- **Problem**: Communication collectives are bottlenecked by the slowest links
  - ➢ Faster links will be underutilized
  - ➢ Barrier in scaling DP
- **Solution**: Use pipeline parallelism across slower links

Combining PP + DP

Problem:

- Integrating PP with DP trades off memory for networking
  - ➢ Data + Pipeline Parallelism
    - high memory, low communication
    - Training state not sharded
  - ➢ Data (FSDP) + Pipeline Parallelism
    - low memory, high communication
    - Parameters need to be gathered for each microbatch

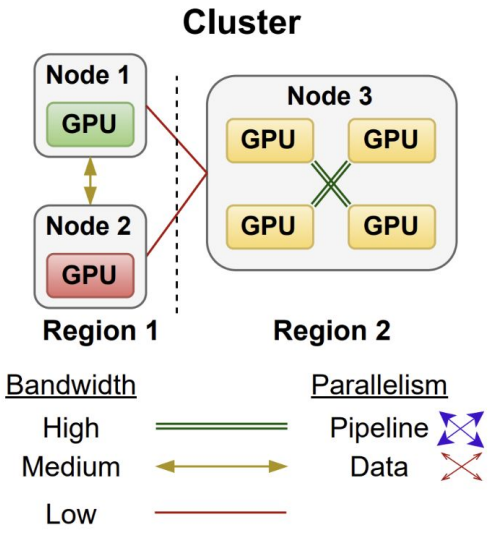
PP Flexibility

Problem:

- For simplicity, most frameworks assume:
  - ➢ Each PP stage is assigned to the same number of GPUs
  - ➢ Same GPU type within each PP stage

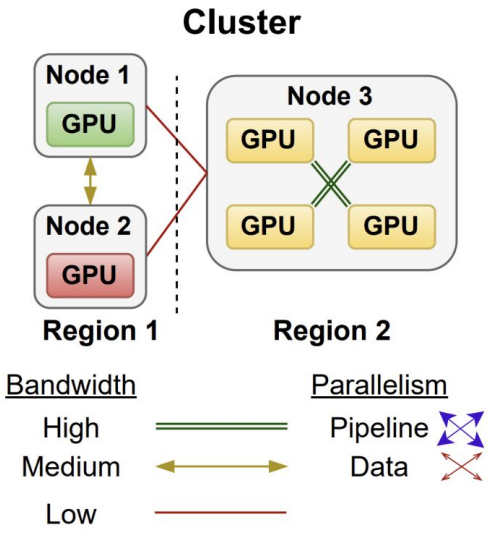
PP Flexibility

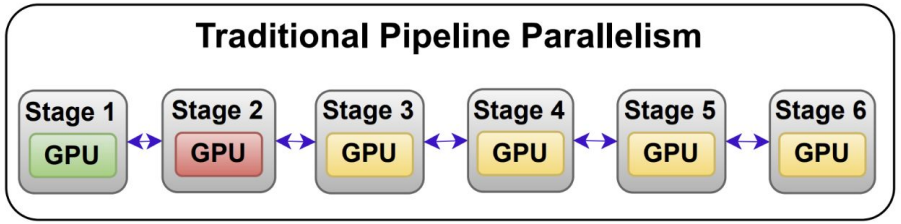
- Can lead to suboptimal configurations in heterogeneous clusters



PP Flexibility

- Can lead to suboptimal configurations in heterogeneous clusters
  - ➢ Requires larger degree of PP → larger pipelining overhead





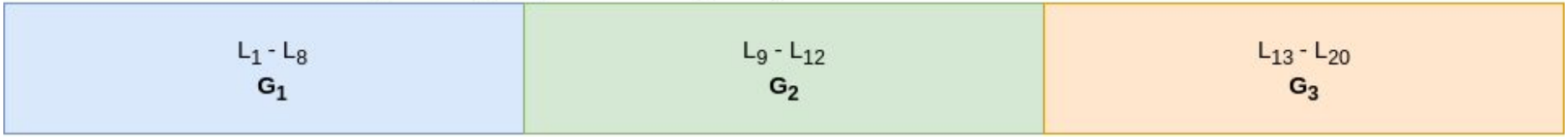
Zorse Design

- A efficient system for distributed training on highly heterogeneous clusters
  - ➢ Pipeline Efficient ZeRO DP: Novel integration of data and pipeline parallelism that is both communication and memory efficient
  - ➢ Heterogeneous pipeline parallelism
  - ➢ A planner to find efficient training configurations over large search space

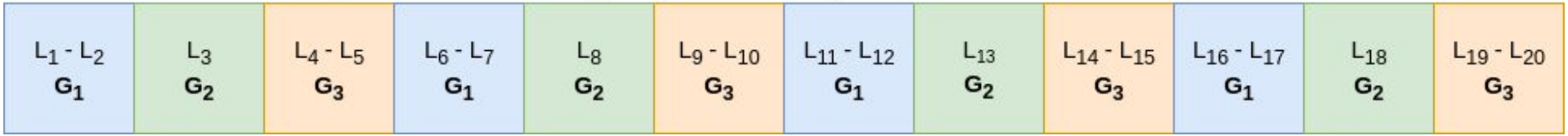
Pipeline-Efficient ZeRO DP

- Interleaved Pipeline Parallelism
  - ➢ Each GPU is assigned multiple smaller ministages

Traditional: One large stage per GPU group

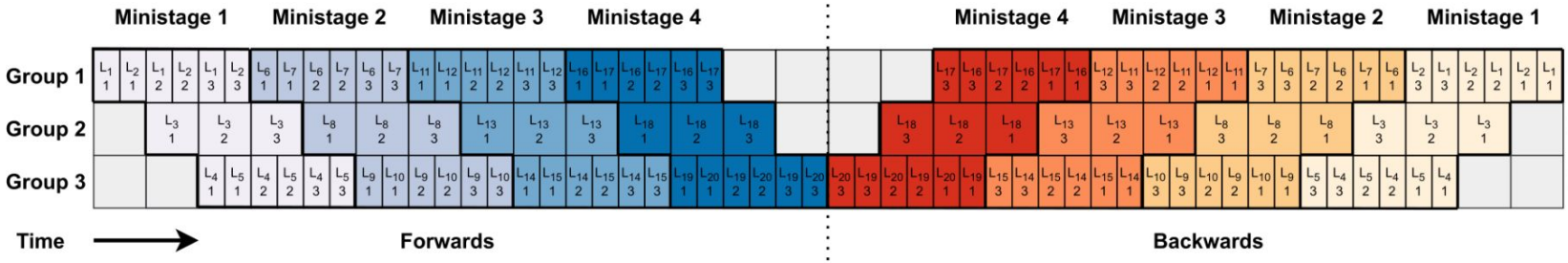


Interleaved: Multiple smaller stages per GPU group



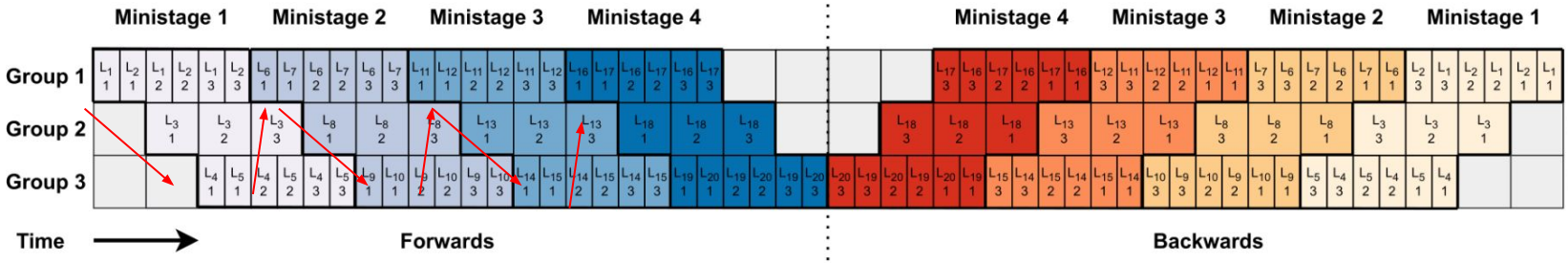
Pipeline-Efficient ZeRO DP

- Interleaved Pipeline Parallelism
  - ➢ Each GPU is assigned multiple smaller ministages
    - ■ 3 GPU groups, 4 ministages per group



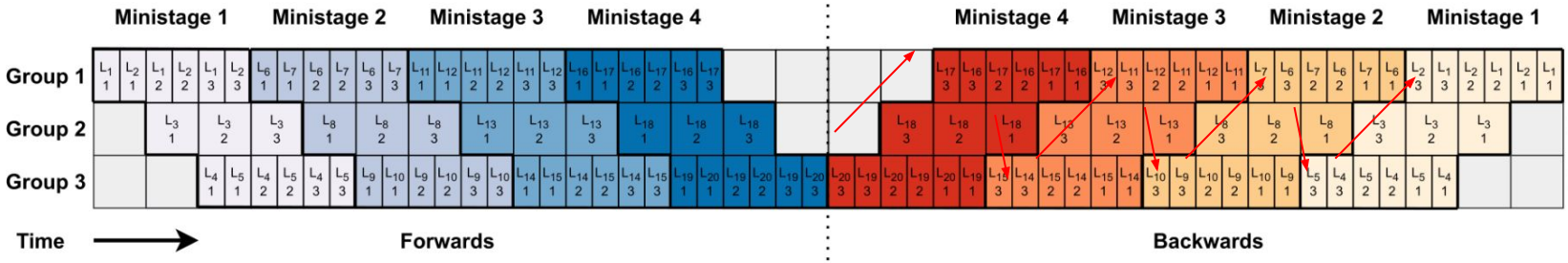
Pipeline-Efficient ZeRO DP

- Interleaved Pipeline Parallelism
  - ➢ Each GPU is assigned multiple smaller ministages
  - ➢ Ministage computation is interleaved



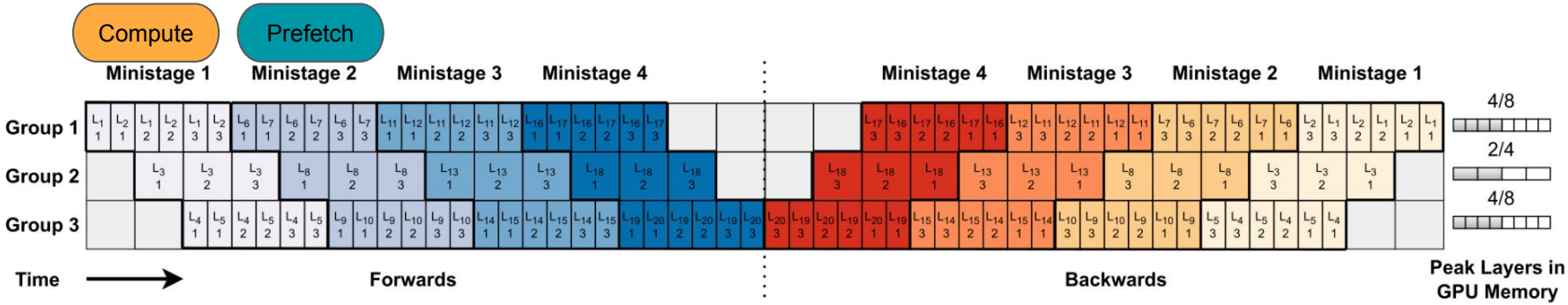
Pipeline-Efficient ZeRO DP

- Interleaved Pipeline Parallelism
  - ➢ Each GPU is assigned multiple smaller ministages
  - ➢ Ministage computation is interleaved



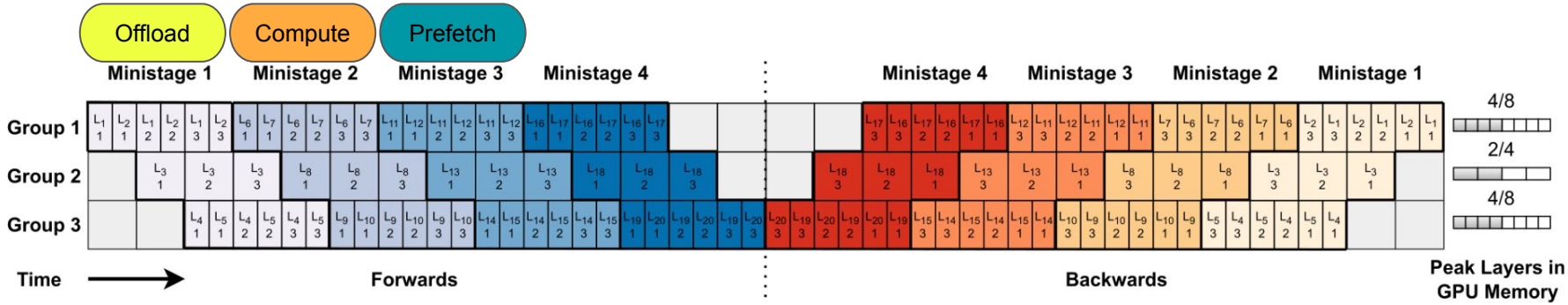
Pipeline-Efficient ZeRO DP

- Activations and parameters are offloaded to CPU when not immediately needed
  - ➢ Next stage parameter prefetch and allgather overlapped with computation



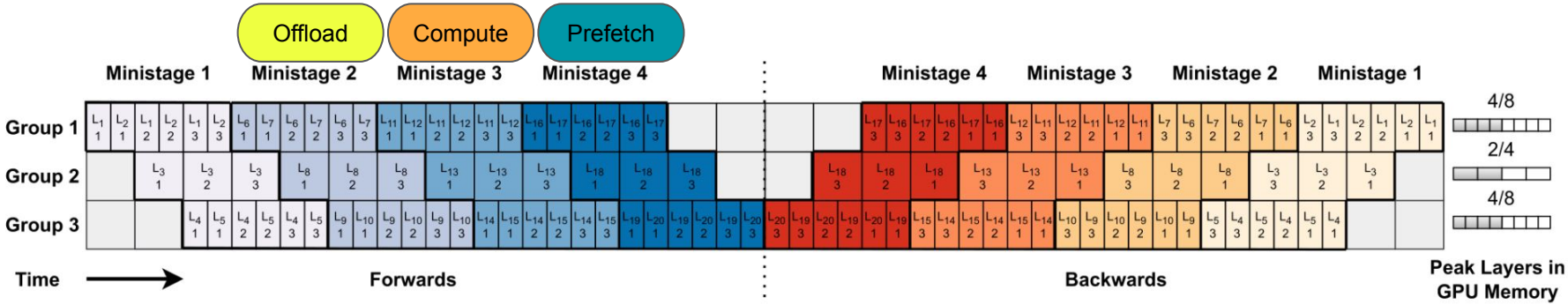
Pipeline-Efficient ZeRO DP

- Activations and parameters are offloaded to CPU when not immediately needed
  - ➢ Next stage parameter prefetch and allgather overlapped with computation
  - ➢ Ministages sharded and offloaded when not in use
  - ➢ 2 ministages kept in memory at a time



Pipeline-Efficient ZeRO DP

- Activations and parameters are offloaded to CPU when not immediately needed
  - ➢ Next stage parameter prefetch and allgather overlapped with computation
  - ➢ Ministages sharded and offloaded when not in use
  - ➢ 2 ministages kept in memory at a time

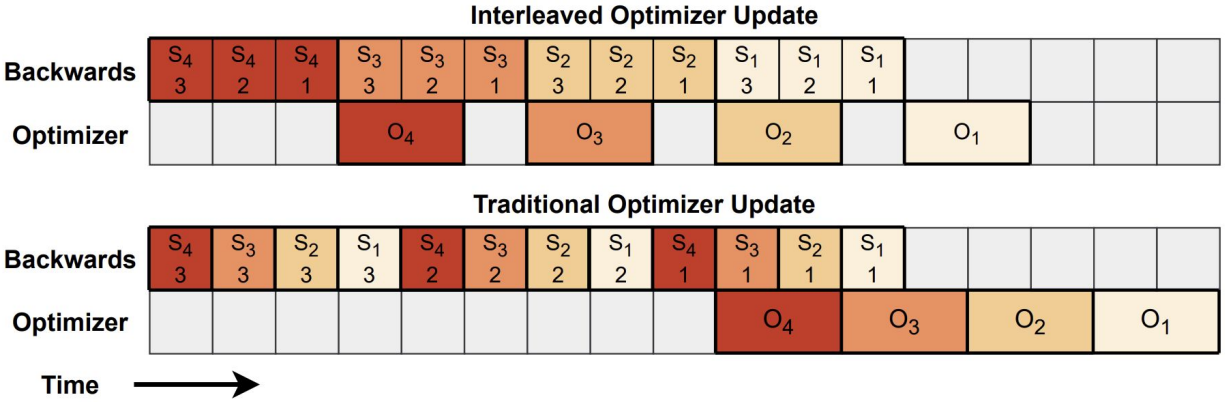


Other Optimizations

- Shard gradients, optimizer state like FSDP
- Checkpoint & Offload activations
  - ➢ Otherwise need to accumulate all activations in memory before backwards pass
  - ➢ Prefetch in backwards pass

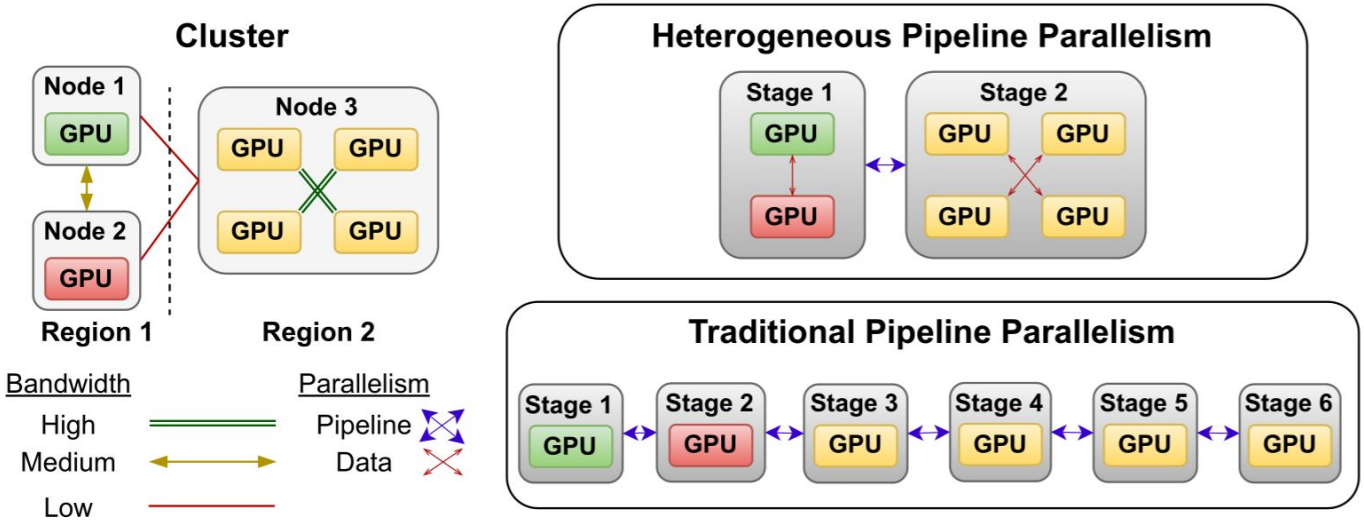
Other Optimizations

- Interleave optimizer updates
  - ➢ Better overlap of communication-computation
  - ➢ Reduce peak memory utilization



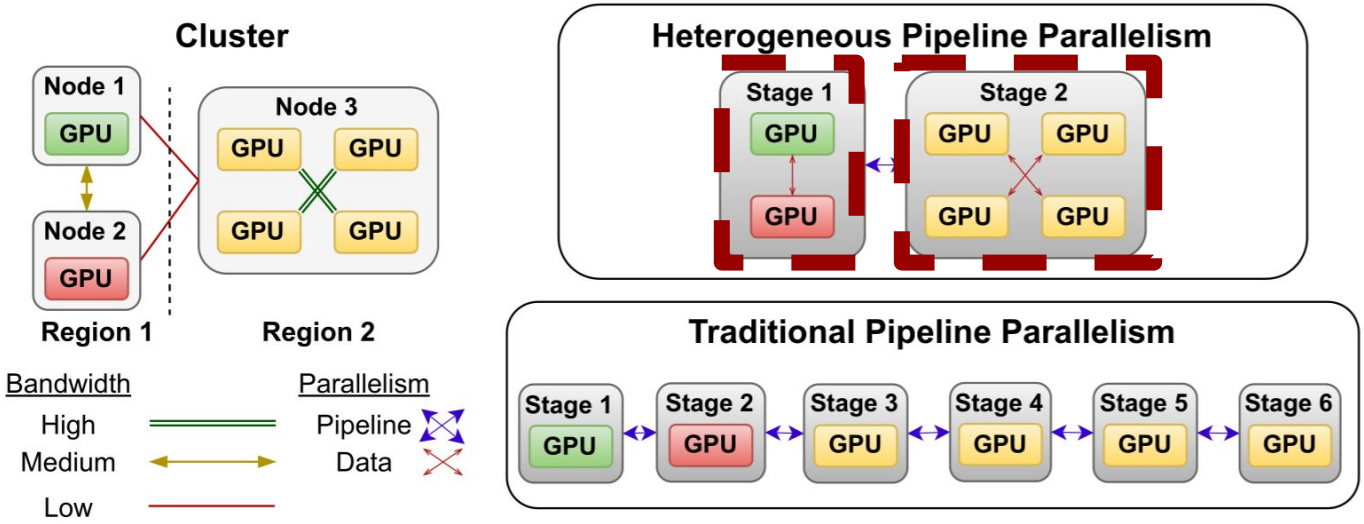
Heterogeneous Pipeline Parallelism

- Zorse supports different numbers of GPUs per pipeline stage, different GPUs within each stage: Fewer stages, leverages faster interconnects



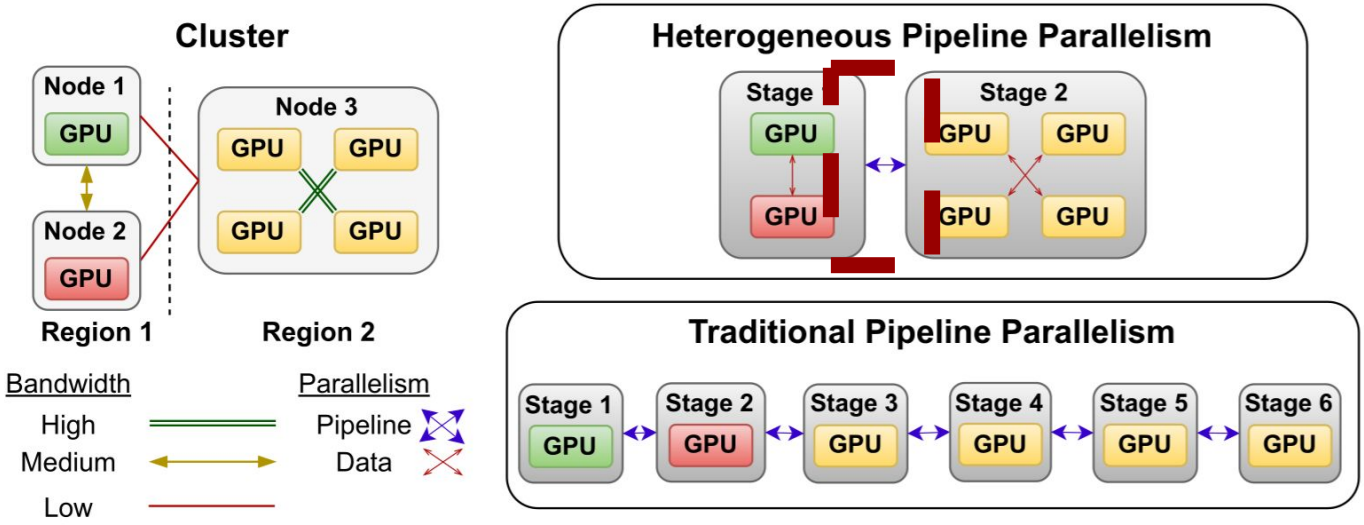
Heterogeneous Pipeline Parallelism

- Uneven # of microbatches per GPU



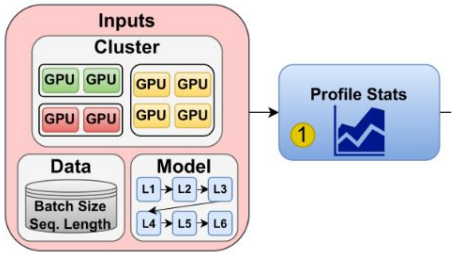
Heterogeneous Pipeline Parallelism

- Uneven # of microbatches per GPU
- Reshards microbatches across stages



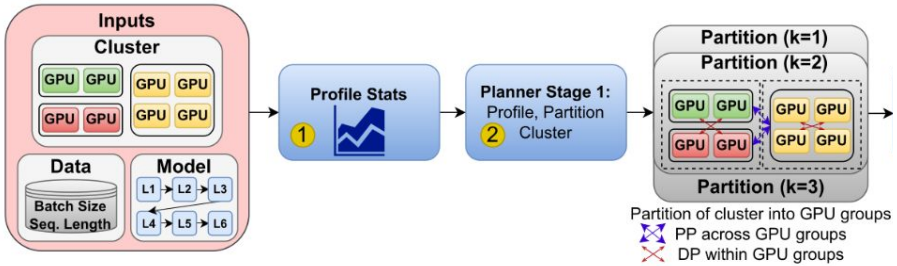
Zorse Planner

- (1) Profile cluster & model stats



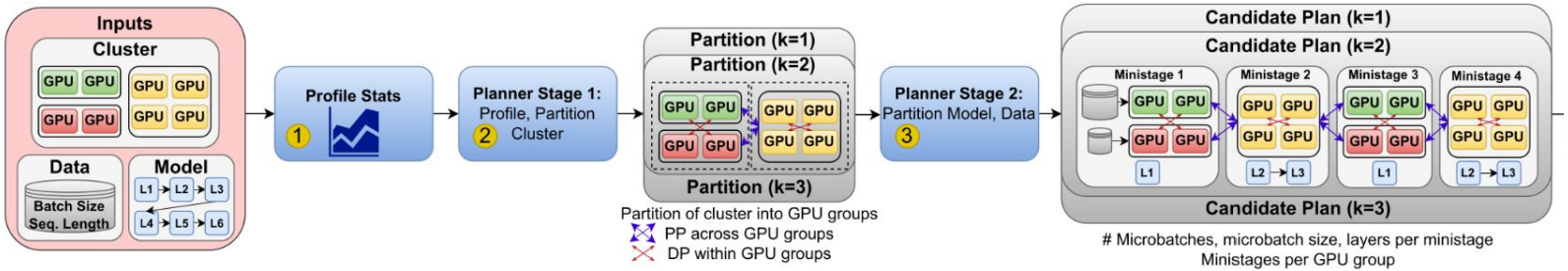
Zorse Planner

- (1) Profile cluster & model stats
- Two phase optimization to simplify search space
  - ➢ (2) Phase 1: Partition cluster into GPU groups minimizing cross group bandwidth



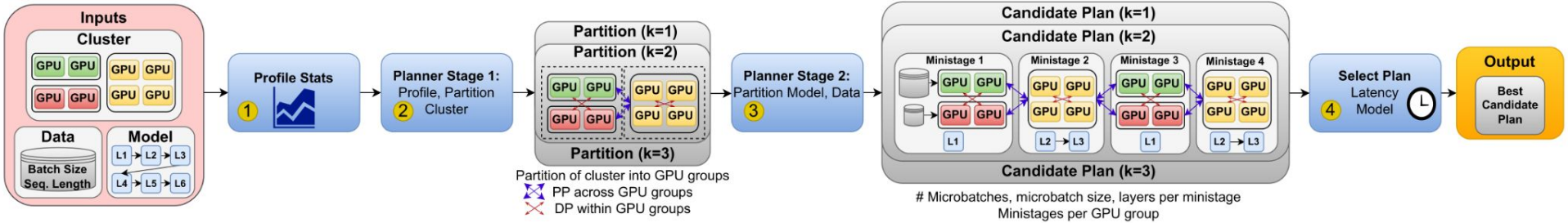
Zorse Planner

- (1) Profile cluster & model stats
- Two phase optimization to simplify search space
  - ➢ (2) Phase 1: Partition cluster into GPU groups minimizing cross group bandwidth
  - ➢ (3) Phase 2: Enumerate feasible # microbatches, and how to configure interleaved pipeline parallelism



Zorse Planner

- (1) Profile cluster & model stats
- Two phase optimization to simplify search space
  - ➢ (2) Phase 1: Partition cluster into GPU groups minimizing cross group bandwidth
  - ➢ (3) Phase 2: Enumerate feasible # microbatches, and how to configure interleaved pipeline parallelism
- (4) Select best configuration based on latency model

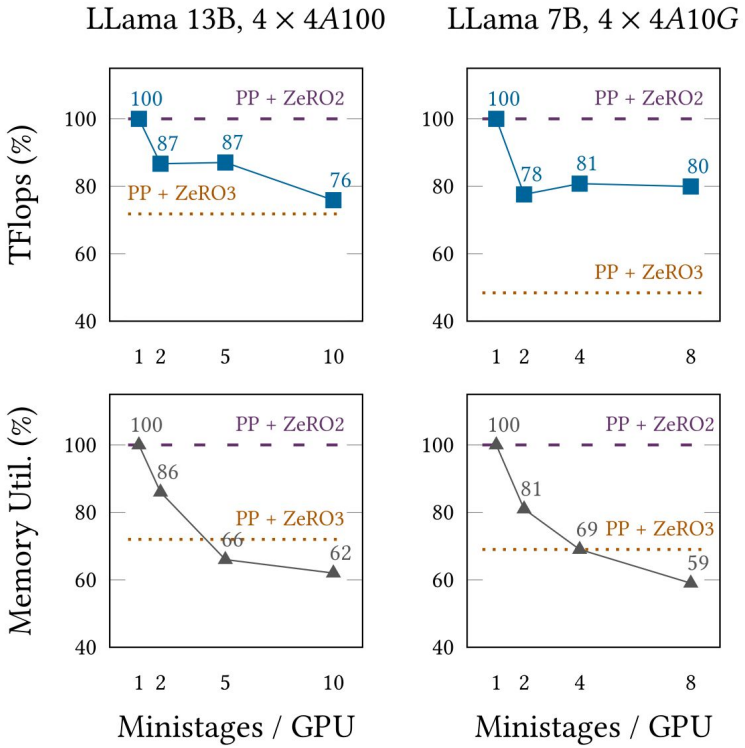


Zorse Evaluation

- **Models**
  - ➢ Llama Variants (7B - 65B)
- **Clusters**
  - ➢ Med-size cluster of low, middle, & high end GPUs
  - ➢ Small-size cluster of high-end GPUs
  - ➢ Large-size cluster of low- & middle-end GPUs
- **Summary**
  - ➢ Compared against 3D parallel heterogeneous training baselines
  - ➢ Zorse achieves up to **3x** speedup over SOTA systems

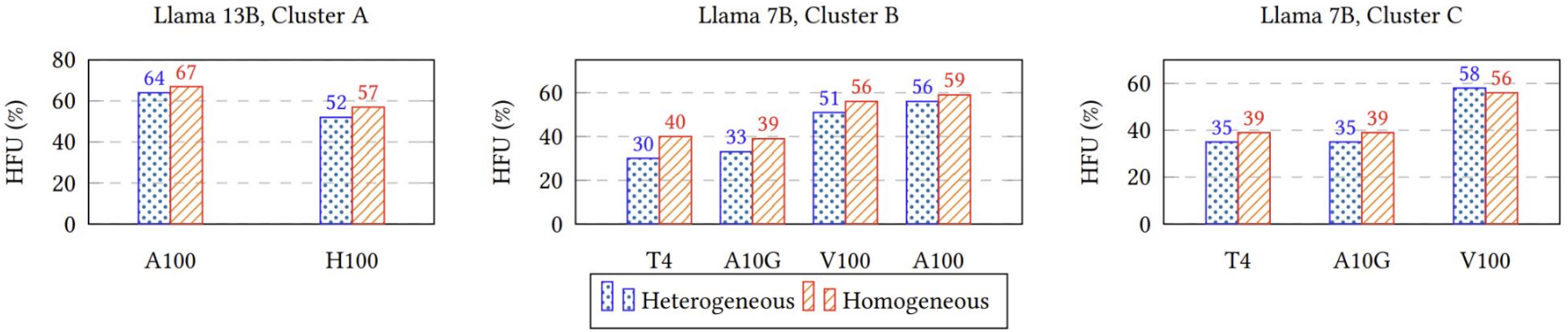
PP+DP Memory-Communication Tradeoff

- Tradeoff between memory utilization and runtime can be tuned via the amount of interleaving
- Zorse can achieve a good balance between memory utilization and runtime



Zorse GPU Utilization

- Zorse maintains high GPU utilization relative to homogeneous training
- Not only relying on the most powerful GPUs



Zorse Conclusion

- Zorse support efficient heterogeneous training by:
  - Achieving both memory and communication efficiency in data & pipeline parallelism
    - Pipeline Efficient ZeRO DP
  - Supporting more flexible configurations of data & pipeline parallelism
    - Heterogeneous pipeline parallelism
  - Using a planner to efficiently find optimized training configs
- See paper for more experiments!