Parallel Strategies and Their Interactions in Training DeepSeek-V3

Abstract

DeepSeek-V3, a 671B parameter Mixture-of-Experts (MoE) model trained on 14.8T tokens, employs 16-way pipeline parallelism (PP), 64-way expert parallelism (EP), and ZeRO-1 data parallelism (DP) to achieve efficient training across 512 A100 80GB GPUs. This paper provides a comprehensive analysis of the roles, interactions, and dependencies among these parallel strategies throughout the training process, encompassing stages: data sharding, forward propagation, loss computation, backward propagation, gradient synchronization, parameter updates, and load balancing. Quantitative data on computation, communication, and storage demands are integrated to illustrate resource utilization. A simplified dependency tree, included as an imported diagram, visualizes the execution flow and interdependencies. The study evaluates how optimizations such as DualPipe, FP8 mixed precision, and efficient communication kernels mitigate critical bottlenecks, ensuring stable and efficient training.

1 Introduction

DeepSeek-V3, a large-scale Mixture-of-Experts (MoE) model with 671 billion parameters, is trained on 14.8 trillion tokens using 512 NVIDIA A100 80GB GPUs. The training process leverages three parallel strategies: 16-way pipeline parallelism (PP), 64-way expert parallelism (EP), and ZeRO-1 data parallelism (DP). These strategies interact through intricate computational, communication, and memory dependencies, carefully orchestrated to optimize training efficiency. This paper analyzes the roles of EP, DP, and PP, their interdependencies, execution order, and resource demands across training stages, highlighting optimizations that address communication, synchronization, and memory bottlenecks.

2 Parallel Strategies Overview

DeepSeek-V3's training distributes computational and memory loads across 512 GPUs, interconnected via 400Gbps InfiniBand for inter-node communication and NVLink for intra-node communication, using the Megatron-LM framework with FP8 mixed precision. The model employs 16-way pipeline parallelism (PP), dividing the model into 16 stages, each containing Transformer blocks with Multi-Head Linear Attention (MLA) and MoE layers. Additionally, 64-way expert parallelism (EP) distributes 512 experts (480 routing and 32 shared) across 8 nodes (64 GPUs), with each token activating 4 routing experts. ZeRO-1 data parallelism (DP) shards parameters across all 512 GPUs to minimize memory redundancy. Notably, tensor parallelism (TP) is avoided, with FP8 and memory optimizations simplifying dependencies.

3 Interaction Dependencies and Execution Order

The training process of DeepSeek-V3 is structured into distinct stages: data sharding, forward propagation, loss computation, backward propagation, gradient synchronization, parameter updates, and load balancing. Each stage involves specific roles for EP, DP, and PP, with complex dependencies governing their interactions and execution order. Resource demands, including computation, communication, and storage, are quantified to provide insight into resource utilization, as summarized in Table 1.

3.1 Data Sharding and Initialization

In the data sharding and initialization stage, ZeRO-1 data parallelism (DP) shards 14.8 trillion tokens and 671 billion parameters across 512 GPUs, allocating approximately 12.51 GB of parameters and 65.5 KB of data per GPU. Expert parallelism (EP) distributes 512 experts (480 routing and 32 shared) across 64 GPUs, with each

Table 1: Resource Demands Across Training Stages

Stage	Computation (GB)	Communication (GB)	Storage (GB)
Data Sharding	0.0655 (data) + 12.51 (params)	$\sim$0	12.51 (params) + 0.0655 (data)
Forward Propagation	1.07 (in/out) + 4.29 (MoE)	11.2 (EP 10.7 + PP 0.502)	19.2 (params 12.51 + acts 4.28 + KV 1.34 + MoE 1.07)
Loss Computation	6.55 (out/loss) + 0.0336 (gating)	0	25.78 (fwd 19.2 + out 6.55 + gating 0.0336)
Backward Propagation	1.07 (grads) + 4.29 (MoE)	11.2 (EP 10.7 + PP 0.502)	19.2 (recomp) or 23.48 (stored acts)
Gradient Sync.	1.31 (grads)	2.62 (DP All-Reduce)	13.82 (rel acts) or 24.79
Parameter Update	1.31 (param update)	0	13.82
Load Balancing	0.0336 (gating stats)	0.000002	13.85

GPU managing roughly 7–8 routing experts and portions of shared experts. Pipeline parallelism (PP) divides the model into 16 stages, each handling approximately 12.51 GB of parameters. DP provides the input data that feeds into EP's expert allocation and PP's stage computations, with EP's expert parameters embedded within PP stages. Memory is shared across DP's parameter shards, PP's stage parameters, and EP's expert parameters, creating a dependency where DP's sharding precedes EP and PP initialization. This stage involves 12.51 GB of parameter computation, 65.5 KB of data computation, negligible communication, and 12.51 GB of storage. Optimizations such as ZeRO-1 reduce memory redundancy, while FP8 mixed precision lowers parameter storage requirements, facilitating large-scale parallelism.

3.2 Forward Propagation

Forward propagation is a critical stage where pipeline parallelism (PP) processes micro-batches sequentially through 16 stages, with each stage's computation depending on the activation outputs from the previous stage, transmitted via Send/Recv operations over InfiniBand and NVLink. Within MoE layers, expert parallelism (EP) employs a gating function to select the top-4 routing experts for each token, using All-to-All communication to distribute tokens and collect outputs across GPUs. Data parallelism (DP) supplies local batch data as the input for PP's first stage and EP's gating function. DP's data sharding initiates the process, followed by PP's stage computations, which provide inputs to EP's MoE layers. EP's All-to-All communication, embedded within PP stages, competes with PP's Send/Recv for InfiniBand bandwidth, creating a significant communication bottleneck. Memory is shared between PP's activation storage (4.28 GB), EP's token buffers (1.07 GB), and key-value caches (1.34 GB). The stage requires 5.36 GB of computation (1.07 GB input/output + 4.29 GB MoE), 11.2 GB of communication (10.7 GB EP + 0.502 GB PP), and 19.2 GB of storage. Optimizations include DualPipe, which employs bidirectional scheduling to reduce pipeline bubbles, efficient All-to-All kernels for computation-communication overlap, node-constrained routing to limit EP's cross-node communication, and FP8 to minimize memory and communication demands.

3.3 Loss Computation

During loss computation, pipeline parallelism (PP) generates logits from the final stage (stage 16) to compute the loss, typically cross-entropy, using local batch labels provided by data parallelism (DP). Expert parallelism (EP) contributes gating values for a sequence-level auxiliary loss to balance expert load across GPUs. PP's logits serve as the primary input for DP's loss computation, while EP's gating values influence the auxiliary loss, creating a dependency where PP and EP outputs precede DP's calculations. The auxiliary loss computation relies on EP's gating distribution, indirectly affecting DP's load balancing. Memory demands are significant, with PP's logits (6.55 GB), forward propagation storage (19.2 GB), and EP's gating values (33.55 MB) totaling 25.78 GB. This stage involves 6.58 GB of computation (6.55 GB output/loss + 0.0336 GB gating), no communication, and 25.78 GB of storage. Optimizations include a minimal balancing factor α in the auxiliary loss to reduce EP load imbalance and a no-token-dropping policy to ensure all tokens contribute to the loss, enhancing stability.

3.4 Backward Propagation

Backward propagation computes gradients, starting from the last pipeline stage (stage 16) and proceeding to the first, with pipeline parallelism (PP) relying on gradient inputs from subsequent stages transmitted via Send/Recv. In MoE layers, expert parallelism (EP) calculates gradients for selected experts, using All-to-All communication to distribute gradient inputs across GPUs. Data parallelism (DP) provides the local batch environment for gradient computations. PP's gradient calculations provide inputs for EP's MoE gradient computations, with EP's All-to-All communication embedded within PP stages, competing with Send/Recv for bandwidth. Memory is shared between PP's activations (recomputed or stored, 4.28 GB) and EP's gradient buffers (1.07 GB), with storage ranging from 19.2 GB (recomputation) to 23.48 GB (stored activations). The stage requires 5.36 GB of computation, 11.2 GB of communication (10.7 GB EP + 0.502 GB PP), and up to 23.48 GB of storage. Optimizations such as DualPipe reduce Send/Recv latency, efficient All-to-All kernels enable computation-communication overlap, activation recomputation minimizes storage needs, and FP8 reduces memory and communication demands.

3.5 Gradient Synchronization and Parameter Updates

Gradient synchronization and parameter updates primarily involve data parallelism (DP), which synchronizes gradients across 512 GPUs using All-Reduce communication and updates parameters, likely with an optimizer like Adam. Pipeline parallelism (PP) supplies stage gradients, and expert parallelism (EP) provides expert gradients from MoE layers, both of which must complete before DP's All-Reduce begins. The dependency chain requires PP and EP to finish gradient computations, with EP's All-to-All and PP's Send/Recv communications preceding DP's All-Reduce, creating a serial communication bottleneck. Memory is shared among DP's gradient buffers (1.31 GB), PP's stage gradients, and EP's expert gradients, with storage demands of 13.82 GB (if activations are released) or 24.79 GB (if stored). This stage involves 1.31 GB of computation, 2.62 GB of communication (All-Reduce), and up to 24.79 GB of storage. Optimizations include efficient All-Reduce kernels for computation-communication overlap, ZeRO-1 to reduce parameter communication, and FP8 to lower gradient storage and communication requirements.

3.6 Load Balancing and Scheduling

Load balancing and scheduling ensure efficient resource utilization. Expert parallelism (EP) dynamically adjusts expert biases to balance token allocation across 64 GPUs, using sequence-level auxiliary loss and gating statistics. Pipeline parallelism (PP) employs DualPipe to schedule micro-batches, optimizing pipeline utilization. Data parallelism (DP) is indirectly affected, as load balancing influences the efficiency of All-Reduce synchronization. EP's load balancing precedes PP's scheduling, which in turn impacts DP's synchronization. The stage requires minimal resources: 33.55 MB of computation for gating statistics, 2 KB of communication, and 13.85 GB of storage. Optimizations include auxiliary loss-independent load balancing to minimize EP imbalance, DualPipe to reduce pipeline bubbles, and a no-token-dropping policy to stabilize execution, ensuring consistent performance across PP and DP.

4 Comprehensive Dependency Analysis

4.1 Dependency Matrix

4.2 Execution Flow

The training process follows a structured execution flow. Initially, data parallelism (DP) shards data and parameters, followed by expert parallelism (EP) allocating experts and pipeline parallelism (PP) assigning stages. In forward propagation, DP provides local batches, PP computes through stages, EP handles MoE layers (gating, All-to-All, expert computation), and PP passes activations. Loss computation involves PP generating logits, EP contributing auxiliary loss, and DP calculating local loss. Backward propagation sees PP computing stage gradients, EP handling MoE gradients, and PP passing gradient inputs. Gradient synchronization is performed by DP via All-Reduce, followed by parameter updates. Load balancing involves EP adjusting biases, PP scheduling via DualPipe, and DP synchronizing under balanced conditions.

4.3 Key Bottlenecks

Several bottlenecks arise during training. Communication bottlenecks occur as EP's All-to-All (10.7 GB) and PP's Send/Recv (0.502 GB) compete for InfiniBand bandwidth, while DP's All-Reduce (2.62 GB) waits for their

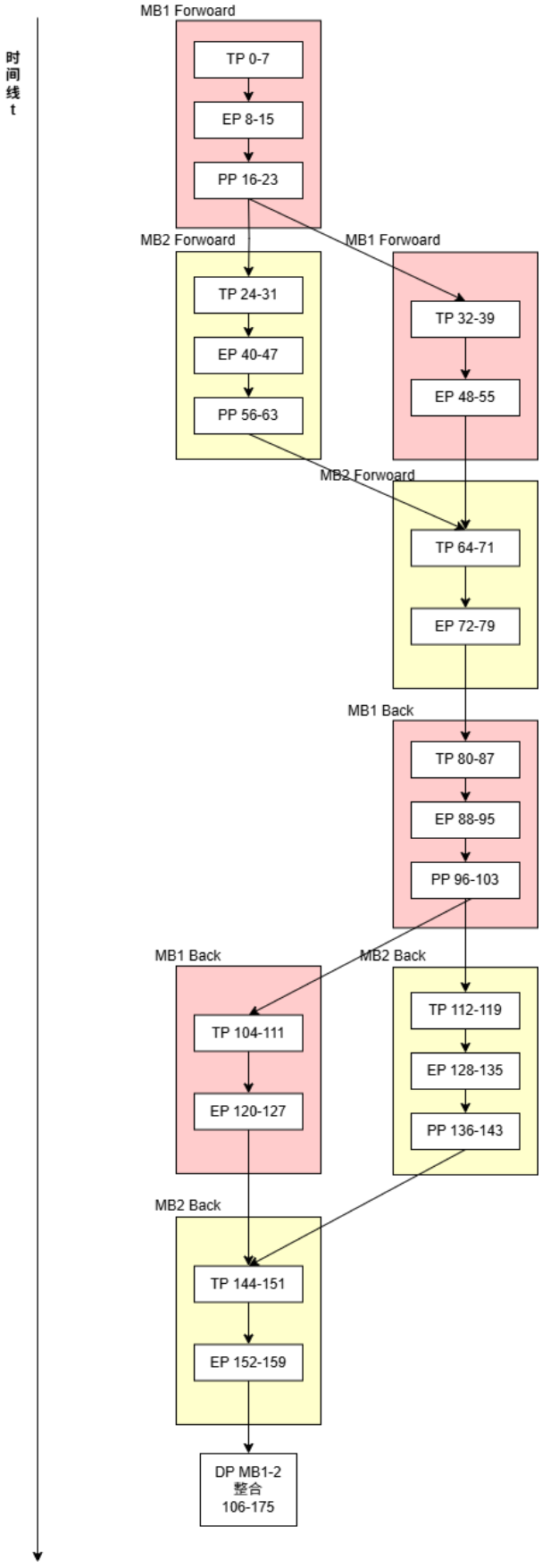

Figure 1: Overview of the research work

Table 2: Parallel Strategy Dependencies and Execution Order

Stage	EP Role	DP Role	PP Role	Dependencies	Execution Order
Data Sharding	Allocate experts	Shard data/params	Allocate stages	DP → EP/PP	DP → EP → PP
Forward Prop.	Gating, All-to-All, experts	Local batches	Stage comp., Send/Recv	DP → PP/EP, PP → EP, EP ↔ PP	DP → PP → EP → PP
Loss Comp.	Gating values	Local loss	Final output	PP → DP/EP, EP → DP	PP → EP → DP
Backward Prop.	Gating/expert grads	Local grad env.	Stage grads	PP → EP, EP ↔ PP	PP → EP → PP
Gradient Sync.	Expert grads	All-Reduce	Stage grads	PP/EP → DP	PP → EP → DP
Param. Update	None	Update params	None	PP/EP → DP	DP
Load Balancing	Adjust biases	Affected by balance	DualPipe sched.	EP → DP/PP, PP → EP	EP → PP → DP

completion, creating serial delays. Synchronization bottlenecks emerge as PP depends on EP's All-to-All completion, and DP waits for PP and EP gradients, exacerbated by load imbalances. Memory bottlenecks result from competition among PP's activations (4.28 GB), EP's buffers (1.07 GB), and DP's parameter shards (12.51 GB), with peak storage reaching 25.78 GB during loss computation.

5 Optimization Measures

Optimizations play a crucial role in mitigating these bottlenecks. DualPipe employs bidirectional scheduling to reduce pipeline bubbles, minimizing PP's Send/Recv latency and easing EP's communication demands. Efficient communication kernels achieve near-100% computation-communication overlap for EP's All-to-All, PP's Send/Recv, and DP's All-Reduce, reducing bandwidth competition. Auxiliary loss-independent load balancing and a no-token-dropping policy stabilize EP's token allocation, enhancing DP and PP efficiency. Node-constrained routing limits EP's All-to-All to fewer nodes, alleviating bandwidth pressure. FP8 mixed precision reduces memory (e.g., 19.2 GB in forward propagation) and communication demands (e.g., 11.2 GB), while ZeRO-1 minimizes DP's memory redundancy (12.51 GB/GPU), supporting large-scale parallelism.

6 Conclusion

DeepSeek-V3's training process orchestrates 16-way pipeline parallelism, 64-way expert parallelism, and ZeRO-1 data parallelism to manage a 671B parameter MoE model across 512 GPUs. The strategies exhibit complex dependencies, with EP's All-to-All communication embedded within PP stages, DP relying on PP and EP for gradient synchronization, and shared memory demands peaking at 25.78 GB. Strict execution ordering, visualized through an imported dependency tree, ensures coordinated operation, while optimizations such as DualPipe, FP8, efficient communication kernels, and load balancing mitigate communication bottlenecks (e.g., 11.2 GB in forward/backward), synchronization delays, and memory pressures, achieving stable and efficient training.