D-VLA: A High-Concurrency Distributed Asynchronous Reinforcement Learning Framework for Vision-Language-Action Models

Yucheng Guo[3*†] **Yongjian Guo**[1,3*] **Zhong Guan**[2,3] **Wen Huang**[1,3]
Haodong Yue[1] **Shuai Di**[3] **Xiong Jun Wu**[3] **Yicheng Gong**[3]
[1]Tsinghua Shenzhen International Graduate School, Tsinghua University
[2]Tianjin University [3]JDT AI Infra

Abstract

Reinforcement learning (RL) for Vision-Language-Action (VLA) models is constrained by resource contention between physics simulation and model computation, as well as communication and scheduling overhead. We present *D-VLA* , a distributed asynchronous RL framework that separates high-frequency trajectory transfer over NCCL from lower-frequency weight synchronization over a CPU-side Gloo channel. A four-thread "Swimlane" pipeline overlaps rollout, weight reception, training, and parameter distribution, while dual-pool GPU memory management and an in-process zero-copy observation path support high-concurrency simulation. We evaluate $\pi_{0.5}$ and OpenVLA-OFT on GPU-intensive ManiSkill and CPU-intensive LIBERO. Mechanism-level experiments isolate improvements in synchronization stability, memory capacity, and observation-transfer latency. On LIBERO, asynchronous scheduling improves throughput despite the absence of the same GPU-physics contention. We further evaluate both GRPO and PPO on ManiSkill. In a separate scaling study with $\pi_{0.5}$, *D-VLA* retains 86.70% throughput scaling efficiency from 8 to 256 GPUs.

1 Introduction

Embodied AI, regarded as a pivotal pathway toward Artificial General Intelligence (AGI), is undergoing a profound paradigm shift driven by the emergence of Vision-Language-Action (VLA) models [1, 2, 3, 4, 5] such as OpenVLA [4], π_0 [2], and GR00T [6]. These models achieve a significant transition from manually designed explicit models to data-driven implicit models by integrating visual perception, language understanding, and action generation into a unified end-to-end framework [7, 8, 9]. Through continuous scaling across massive computational resources and datasets, VLA models have demonstrated unprecedented potential in cross-task and cross-morphology adaptation. However, despite these notable advancements, current training paradigms remain heavily reliant on imitation learning based on Supervised Fine-Tuning (SFT). Existing frameworks like LeRobot [10] and

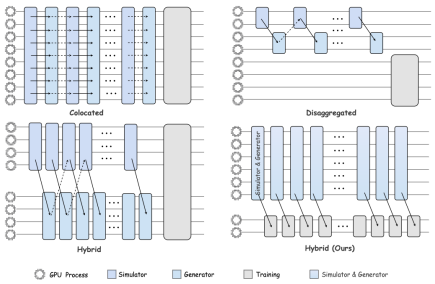

Figure 1: Placement Strategies across Different Training Frameworks

*Equal contributions.
†Corresponding author.
The source code is available at `https://github.com/Aurora24Thu/D-VLA`.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

GR00T [6] primarily utilize expert demonstration data to fine-tune policies via behavior cloning. This SFT-centric path faces multiple severe challenges in practical applications: first, large-scale human-collected robot trajectory data is both costly and difficult to obtain, which strictly limits the scaling of models [11, 12]; second, processing high-dimensional data and complex multimodal architectures imposes a heavy burden on training cycles and inference latency. Furthermore, constrained by the limited diversity of offline datasets, SFT models often exhibit weak generalization capabilities when encountering distribution shifts and unseen tasks [13, 14]. Unlike human learning mechanisms, SFT cannot support agents in discovering new action patterns through autonomous exploration beyond demonstration data. Consequently, the research community is increasingly pivoting toward Reinforcement Learning (RL) frameworks to break through these limitations via online interaction.

In response to the deficiencies of SFT, several RL frameworks have emerged, optimizing different dimensions of the training pipeline. For instance, RLinf [15] provides a universal distributed training and evaluation interface, offering a unified framework for multimodal agents and VLAs. RL-VLA^3 [16] implements a three-stage asynchronous pipeline to decouple data collection, policy inference, and model updates, thereby maximizing hardware utilization and training throughput. SimpleVLA-RL [17] designs rule-based outcome rewards and interactive trajectory sampling for VLA models, proving that performance can surpass SFT even with minimal demonstrations. Additionally, Dexbotix [18] focuses on integrating tactile feedback for high-degree-of-freedom dexterous hands, while Vlab [19] is dedicated to building specialized sim-to-real transfer environments.

Unlike traditional online RL, Embodied AI training involves deep coupling between high-fidelity physics simulation and large-scale deep learning models. The former features high-frequency, fragmented occupation of computational resources, while the latter demands extreme throughput for GPU memory capacity and communication bandwidth. Existing distributed training frameworks, such as RLinf-VLA [15] and RL-VLA^3 [16], introduce hybrid resource allocation and fine-grained asynchronous mechanisms to alleviate computational pressure. However, they have yet to fundamentally resolve the resource contention and execution conflicts between simulation tasks and model optimization at the underlying architecture level. As a result, the overall system throughput remains bottlenecked by the slowest physics stepping or synchronization overhead.

The current performance bottlenecks in embodied RL systems primarily stem from the high degree of coupling between simulation logic and learning logic on the execution plane. On one hand, frequent memory allocation and deallocation by physics engines easily lead to severe memory fragmentation in deep learning frameworks. On the other hand, the frequent transfer of massive multimodal environment data (e.g., high-resolution images) between sampling and inference components introduces significant serialization overhead and communication latency. This systemic "blocking" effect is particularly severe when handling long-sequence interaction tasks, restricting the sample acquisition efficiency of agents in complex scenarios.

To address these challenges, we propose *D-VLA* , a **d**istributed **VLA** RL framework. Its “Plane Decoupling” design separates high-frequency trajectory communication from lower-frequency weight synchronization to reduce interference with simulation. A four-thread “Swimlane” pipeline overlaps sampling, weight reception, gradient training, and parameter distribution; the achievable overlap depends on the balance between rollout and actor workloads. Dual-pool memory management and a zero-copy observation path support co-located, separated, and hybrid placements (Figure 1). Local topology replication preserves the locality of environment–inference communication as GPU count grows. We evaluate the framework with Group Relative Policy Optimization (GRPO) [20] and Proximal Policy Optimization (PPO), and distinguish the GPU-contention regime of ManiSkill from the host-side scheduling bottlenecks of LIBERO.

The primary contributions of this paper are summarized as follows:

- **Decoupled communication and asynchronous execution:** We separate NCCL trajectory transfer from Gloo weight synchronization and overlap four execution threads, reducing synchronization stalls between simulation and learning.

- **Memory capacity and communication locality:** We combine dual-pool GPU memory management, in-process zero-copy observation access, and local topology replication. Targeted experiments quantify the capacity benefit of memory partitioning and the latency benefit of zero-copy access.

- **Evaluation across workloads and scales:** We benchmark 3B and 7B VLA models on ManiSkill and LIBERO, evaluate GRPO and PPO, and report multi-node weak scaling up to 256 GPUs.

2 Related Work

Evolution of Vision-Language-Action Models In the field of Embodied AI, the evolution of Vision-Language-Action (VLA) models is undergoing a paradigm shift from fundamental Supervised Fine-Tuning (SFT) [21] toward Reinforcement Learning (RL) [19] frameworks with enhanced generalization capabilities. Early foundational VLA models such as RT-2 [1], OpenVLA [4], and π_0 [2] achieved preliminary robotic manipulation capabilities by training on large-scale datasets like Open-X Embodiment [22]. Subsequently, models like $\pi_{0.5}$ [23] and SmolVLA [5] further optimized parameter efficiency and inference speed while maintaining performance. Concurrently, general-purpose humanoid control models such as Gr00t N1.5 [6] have demonstrated broad application prospects. However, due to the limitations of SFT methods in handling out-of-distribution data, research focus is gradually shifting toward utilizing RL to enable continuous dynamic environmental adaptation in complex benchmarks such as ManiSkill [24] and LIBERO [25].

Frameworks for VLA Training and Optimization To support the efficient training of large-scale VLAs, the academic community has developed a series of optimization frameworks ranging from low-level control to high-level learning. Frameworks such as LeRobot [10] and DexBotix [18] provide vertically integrated toolchains for end-to-end learning, covering the entire process from data processing to policy fine-tuning for algorithms like ACT and Diffusion Policy. Early attempts such as SimpleVLA-RL [17] and RLinf-VLA [15] sought to migrate RLHF workflows from Large Language Models (LLMs) to embodied tasks. Nevertheless, the high latency introduced by physics simulators causes synchronous training pipelines to face severe throughput bottlenecks. To address this, RL-VLA³ [16] proposed a fully asynchronous distributed architecture that decouples simulation, inference, and training processes. By drawing on the design philosophies of LLM training systems like veRL [11], OpenRLHF [12], and ROLLART [26], it significantly enhances hardware utilization and training efficiency for policy optimization.

System Challenges in Embodied AI Embodied AI systems face system-level challenges distinct from those of traditional language models during training, primarily stemming from the uncertainties inherent in physics simulators [21]. Unlike the relatively stable neural reward models used in LLM training [27], VLA training requires frequent calls to simulation environments like RoboCasa [28]. These environments exhibit high and fluctuating CPU and GPU resource consumption, which can easily lead to computational idling and resource waste [24, 28]. To address these pain points, frontier research such as RL-VLA^3 [16] has introduced dynamic batch scheduling and fine-grained environment sharding techniques, aiming to resolve the asynchronous bottlenecks caused by simulators while maximizing both sample efficiency and system throughput.

3 System Design of *D-VLA*

3.1 *D-VLA* Framework Overview

The *D-VLA* framework (Figure 2) targets high-concurrency, low-latency VLA training. Its "Plane Decoupling" design separates high-frequency data exchange from lower-frequency weight-control logic. This separation reduces interference between communication and simulator synchronization. A four-thread asynchronous pipeline enables overlap between rollout (simulation and inference), weight reception, gradient computation, and parameter distribution.

3.2 Flexible Deployment and Placement Strategies

To address different resource requirements, *D-VLA* supports co-located, separated, and hybrid deployments. Environment simulation involves frequent memory operations, while model inference demands memory bandwidth. Component placement and rollout:actor GPU ratios are fixed for each run according to hardware topology and workload. In hybrid mode, environments and rollout

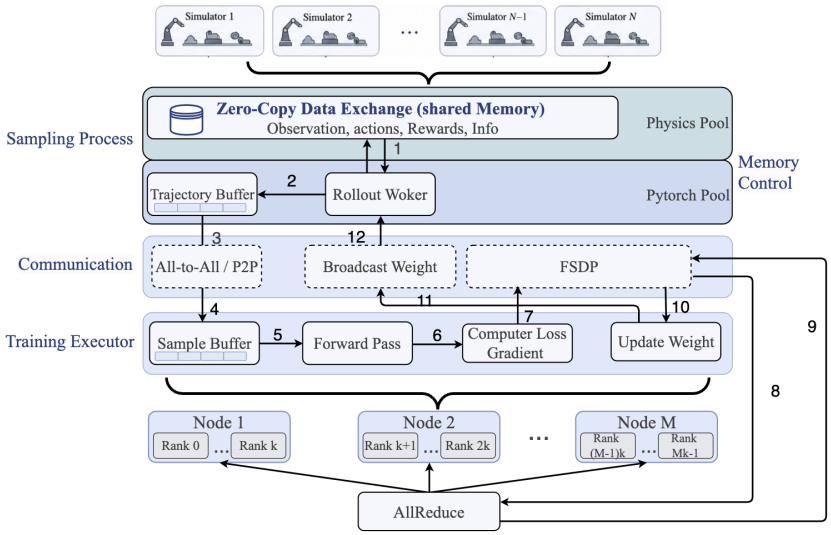

Figure 2: Overview of *D-VLA* . Rollout GPUs co-locate PhysX environments with an inference policy, enabling in-process observation access. Trajectories are dispatched to actor GPUs over NCCL for training under FSDP; updated weights return through a background Gloo channel. The four-thread pipeline overlaps these stages, with the achievable benefit depending on rollout–actor balance.

inference share a device group to minimize observation-transfer overhead, while actor training uses a separate group.

To optimize memory overhead, we designed a zero-copy data exchange mechanism for co-located deployments. Traditional distributed communication involves multiple rounds of data serialization, which creates significant overhead when handling high-resolution images. By maintaining different execution threads within the same process space, *D-VLA* allows observation results from the simulation environment to be accessed directly by inference components, achieving "seamless" data flow at the memory level and reducing bandwidth consumption. For tasks requiring massive parallel simulation, *D-VLA* utilizes a separated deployment strategy with pipeline communication, allowing the environment side to perform physical stepping while the inference side concurrently processes the previous decision logic.

3.3 Decoupled Communication Architecture

A key component of *D-VLA* is the separation of the "Training Data Plane" from the "Weight Control Plane." Rollout trajectories constitute the bulk data-plane traffic and are transferred using NCCL. Communication patterns are configured according to resource ratios: symmetric placements can use point-to-point transfer, while asymmetric placements distribute batches across actor workers. Metadata and control signaling are kept separate from bulk tensor transfer.

Model weight synchronization uses a CPU-side Gloo backend with contiguous host buffers. Weight updates are less frequent than environment interaction. Moving communication to a background host channel avoids coupling its progress to CUDA synchronization in the PhysX reset path; staging and applying weights still incur data-movement costs. The stability experiment in Section 4.4 compares the weight-synchronization backends under this scheduling path: the NCCL variant deadlocks, whereas the Gloo configuration completes training.

3.4 End-to-End Asynchronous Pipeline and Data Flow

Rollout workers accumulate trajectories and send a batch once the training batch size is reached. An actor-side host-memory queue buffers differences between data production and consumption. Successful episodes are reset and replenished without waiting for a single slow trajectory to finish. For M rollout and N actor workers, the rollout batch is chosen as a multiple of N to distribute training data evenly. These mechanisms reduce sensitivity to individual long episodes.

The "Swimlane" model consists of a sampling thread, a weight-receiving thread, a training thread, and a weight-distribution thread, coordinated by lightweight semaphores. Their overlap reduces idle intervals relative to synchronous alternation. The queue interface and communication backend are independent of the policy objective; the evaluated training paths include GRPO with micro-batching and PPO. Episodes have a bounded horizon, limiting trajectory-length variation: the deployment described in our experiments typically needs roughly 40 actions for short successful trajectories and truncates by about 100 actions; the GRPO/PPO comparison specifically uses an 80-step horizon.

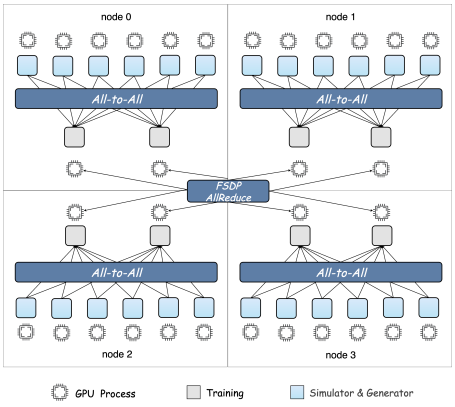

Figure 3: Schematic of Multi-Node Communication in *D-VLA*

3.5 Memory Management and Large-Scale Node Scaling

To mitigate memory pressure from non-framework components such as physics engines, *D-VLA* partitions memory into a "Model Computation Pool," managed by Torch's caching allocator, and an "Environment Auxiliary Pool" for physics-engine temporary objects (Figure 2). This partition limits competition between cached model allocations and simulator buffers. It raises the measured concurrency ceiling.

For cross-node scaling, *D-VLA* replicates a local sampling–inference topology (Figure 3), preserving the locality of high-frequency observations. In the evaluated topology, environment–rollout traffic remains within each node, while cross-node model synchronization is handled among actor workers using Fully Sharded Data Parallel (FSDP). Background weight distribution separates communication progress from the simulator's CUDA synchronization path. This locality reduces communication overhead as the cluster grows.

Portability. The communication and scheduling layers interact with environments through queues carrying observations, actions, and rewards. Adapting a new physics engine therefore requires an environment data adapter rather than a redesign of Plane Decoupling or Swimlane scheduling. Our experiments cover single-agent tasks in ManiSkill and LIBERO.

4 Experiments

We evaluate end-to-end throughput, component-level behavior, learning outcomes, and multi-node scaling. ManiSkill stresses GPU simulation and model execution, whereas LIBERO provides a complementary CPU-intensive workload. The system comparison, algorithm comparison, and scaling study use distinct configurations, specified below rather than treated as interchangeable measurements.

4.1 Experimental Setup

Hardware and models. All experiments are conducted on NVIDIA H200 GPUs, each with 141 GB of memory. Multi-node experiments run on a cluster with InfiniBand interconnects. We evaluate two VLA models: $\pi_{0.5}$ (3B parameters), which generates action chunks via iterative flow matching, and OpenVLA-OFT (7B parameters), which uses parallel decoding with continuous action regression. Figures 4 and 5 report 8-GPU experiments, Table 1 reports 16-GPU experiments, and Table 4 extends to 256 GPUs.

Benchmarks and algorithms. ManiSkill uses GPU-accelerated physics and rendering, creating competition with model inference for device memory and compute. LIBERO is CPU-intensive and stresses a different execution path. The system benchmarks use GRPO; Section 4.6 additionally compares GRPO and PPO on ManiSkill PutOnPlateInScene25Main-v3.

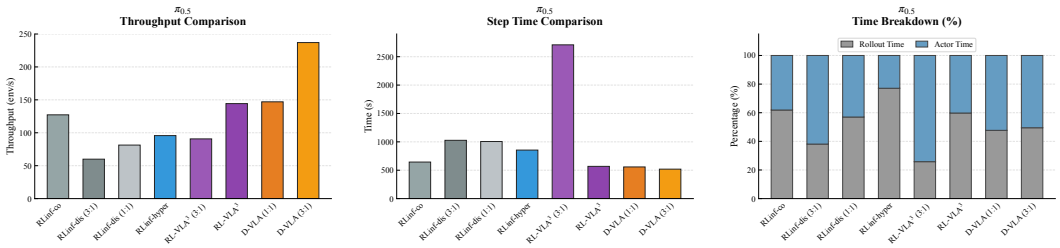

Figure 4: $\pi_{0.5}$ on ManiSkill with 8 H200 GPUs. Left: throughput (env/s). Middle: end-to-end step time (seconds per training iteration). Right: rollout/actor timing breakdown; absolute times are seconds per iteration. Ratios denote rollout-side:actor GPU allocation.

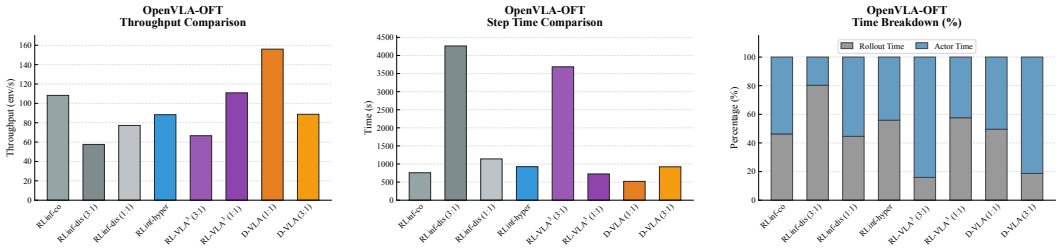

Figure 5: OpenVLA-OFT on ManiSkill with 8 H200 GPUs. Throughput is in env/s; step, rollout, and actor times are seconds per training iteration, not milliseconds per action. Ratios denote rollout-side:actor GPU allocation.

Baselines and placement. We compare RLinf-VLA [15] in colocated (RLinf-co), disaggregated (RLinf-dis), and hybrid (RLinf-hyper) modes, and the asynchronous RL-VLA^3 framework [16]. The evaluated RL-VLA^3 deployment separates environment, rollout, and actor components, whereas *D-VLA* co-locates environment and rollout inference and isolates actor training. Ratios denote the GPU allocation between rollout inference and policy training. With 8 GPUs, 1:1 assigns 4 GPUs to rollout and 4 to training, while 3:1 assigns 6 GPUs to rollout and 2 to training. The same proportions apply at larger GPU counts. The ratios describe the nominal resource split, not identical internal placements across frameworks. The hybrid configuration uses two simulator sub-stages (`pipe=2`); this pipeline-depth setting is aligned across the compared pipelined implementations. Ratios are selected per experiment, not adjusted online.

Workload and statistics. The 16-GPU comparison uses the reported settings of 128 environments/GPU, four rollout epochs, and a global training batch size of 2,048. LIBERO uses a global batch size of 1,024 and a mini-batch size of 128; its per-placement environment counts appear in Table 3. Shared rollout and training settings are matched within each comparison. Throughput is measured in env/s (environment state transitions per second), not action-chunk inference calls per second. Step, rollout, and actor times are seconds per training iteration; step time denotes end-to-end iteration time, not single-action latency. The 16-GPU and LIBERO tables report means and standard deviations across six random seeds; LIBERO explicitly uses sample standard deviation. Statistics for the algorithm comparison have a separate definition in Table 5. Appendix A provides additional configuration details.

4.2 ManiSkill Throughput and Resource Balance

8-GPU and 16-GPU results. For $\pi_{0.5}$ on 8 GPUs (Figure 4), *D-VLA* (1:1) achieves 147.0 env/s, outperforming RLinf-co (127.24 env/s) by 15.53%. *D-VLA* (3:1) further increases throughput to 237.0 env/s, an 86.26% improvement over RLinf-co. On 16 GPUs, *D-VLA* (1:1) achieves 336.04 env/s (Table 1). For OpenVLA-OFT on 8 GPUs, *D-VLA* achieves 156.0 env/s, outperforming RLinf-co (108.24 env/s) and RL-VLA^3 (110.88 env/s) by 44.12% and 40.69%, respectively.

Iteration latency and overlap. The 8-GPU $\pi_{0.5}$ results report a *D-VLA* iteration time of 566.41 s versus 1,006.8 s for RLinf-dis, a 43.74% reduction. Asynchronous execution overlaps rollout and training rather than adding their durations sequentially. In the OpenVLA-OFT 3:1 configuration,

Table 1: ManiSkill system comparison on 16 GPUs. Throughput is mean ± standard deviation over six seeds (env/s). The highest throughput for each model is bold. Full iteration, rollout, and actor timings appear in Table 2.

Framework	$\pi_{0.5}$	OpenVLA-OFT
RLinf-co	232.23 ± 2.32	87.20 ± 0.87
RLinf-dis (1:1)	150.58 ± 1.51	107.23 ± 1.07
RLinf-dis (3:1)	175.29 ± 1.75	99.33 ± 4.99
RLinf-hyper	171.74 ± 1.72	77.23 ± 2.77
RL-VLA^3 (1:1)	244.61 ± 2.45	170.48 ± 2.70
RL-VLA^3 (3:1)	250.77 ± 2.51	152.98 ± 3.53
D-VLA (1:1)	336.04 ± 3.36	$\mathbf{250.90 \pm 2.51}$
D-VLA (3:1)	$\mathbf{376.00 \pm 3.76}$	154.23 ± 1.54

actor-side model updates become the bottleneck. Changing the configured resource ratio toward 1:1 allocates more capacity to training and improves overlap.

Table 2: Detailed ManiSkill timing results on 16 GPUs. All timings are reported as mean ± standard deviation across six seeds, in seconds per training iteration. Step denotes the end-to-end iteration time. Rollout and actor execution may overlap.

Framework	Step	Rollout	Actor
$\pi_{0.5}$ (3B)			
RLinf-co	705.50 ± 8.47	503.50 ± 6.04	200.50 ± 2.41
RLinf-dis (1:1)	1088.00 ± 13.06	672.60 ± 8.07	383.60 ± 4.60
RLinf-dis (3:1)	2840.00 ± 34.08	1662.02 ± 19.94	1142.30 ± 13.71
RLinf-hyper	954.00 ± 11.45	751.20 ± 9.01	199.00 ± 2.39
RL-VLA^3 (1:1)	669.80 ± 8.04	525.66 ± 6.31	383.00 ± 4.60
RL-VLA^3 (3:1)	1960.00 ± 23.52	1580.24 ± 18.96	1420.33 ± 17.04
D-VLA (1:1)	488.32 ± 5.86	460.65 ± 5.53	452.33 ± 5.43
D-VLA (3:1)	1307.23 ± 15.69	1299.20 ± 15.59	1288.02 ± 15.46
OpenVLA-OFT (7B)			
RLinf-co	1877.00 ± 22.52	259.00 ± 3.11	1607.50 ± 19.29
RLinf-dis (1:1)	1528.00 ± 18.34	508.52 ± 6.10	1001.80 ± 12.02
RLinf-dis (3:1)	1628.00 ± 59.38	896.60 ± 10.76	1030.23 ± 48.36
RLinf-hyper	2123.50 ± 25.48	432.75 ± 5.19	1675.00 ± 20.10
RL-VLA^3 (1:1)	961.03 ± 11.53	354.20 ± 4.25	814.40 ± 9.77
RL-VLA^3 (3:1)	1058.35 ± 12.70	761.60 ± 9.14	816.40 ± 9.80
D-VLA (1:1)	653.23 ± 7.84	508.23 ± 6.10	568.24 ± 6.82
D-VLA (3:1)	1062.30 ± 12.75	708.23 ± 8.50	868.24 ± 10.42

Structural comparison with RL-VLA^3. Both frameworks overlap rollout and training, but differ in observation locality. In the tested RL-VLA^3 placement, observations cross GPU boundaries between environment and inference workers; *D-VLA* accesses them within the same process. On 16 GPUs with OpenVLA-OFT and a 3:1 split, throughput is close (154.23 versus 152.98 env/s). At 1:1, *D-VLA* reaches 250.90 versus 170.48 env/s, a 47.17% improvement. Section 4.4 reports isolated transfer-latency measurements for the data path.

Compare with RLinf-hyper. RLinf-hybrid placement is not uniformly preferable in baseline. For OpenVLA-OFT at 16 GPUs, RLinf-hyper is below both RLinf-co and RLinf-dis, while for $\pi_{0.5}$ it exceeds RLinf-dis (1:1). The former ordering agrees with the corresponding 16-GPU, `pipe=2` ManiSkill configuration reported for RLinf-VLA [15].

Learning behavior. The ManiSkill success-rate curves in Figure 6 show competitive learning behavior under the evaluated asynchronous configuration. Section 4.6 reports a separate task-level comparison for both GRPO and PPO.

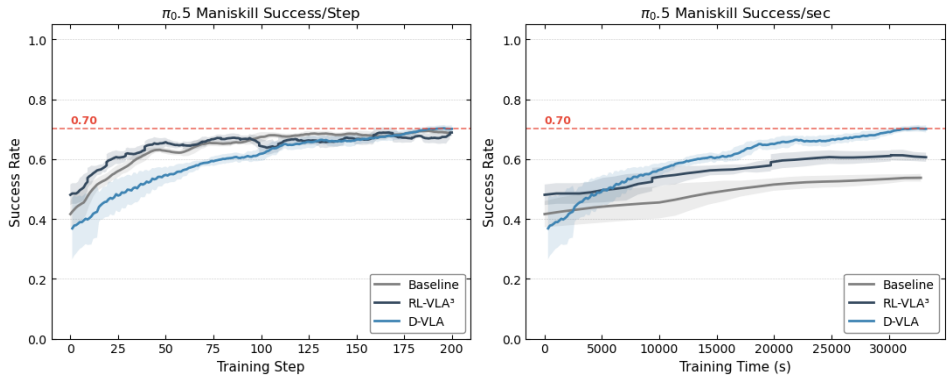

Figure 6: Training success rate on ManiSkill with $\pi_{0.5}$ under the evaluated system configurations.

Table 3: LIBERO steady-state throughput on 8 GPUs (env/s, mean $\pm$ sample standard deviation over six seeds). Ratios 6:2 and 4:4 correspond to 3:1 and 1:1.

Model	Roll:act	Env/GPU	Total env	RLinf	RL-VLA^3	*D-VLA*
$\pi_{0.5}$	6:2	32	192	168.63 ± 0.78	227.05 ± 2.10	298.45 ± 3.41
$\pi_{0.5}$	6:2	64	384	160.10 ± 0.87	208.99 ± 2.80	268.75 ± 5.22
$\pi_{0.5}$	4:4	32	128	165.30 ± 2.01	184.80 ± 2.90	208.63 ± 3.64
$\pi_{0.5}$	4:4	64	256	158.50 ± 1.10	173.36 ± 2.50	191.52 ± 3.72
OpenVLA-OFT	6:2	32	192	161.60 ± 0.02	191.66 ± 0.40	228.40 ± 0.54
OpenVLA-OFT	6:2	32	128	145.40 ± 2.75	173.26 ± 1.60	207.30 ± 0.75
OpenVLA-OFT	4:4	32	128	181.22 ± 0.62	214.11 ± 0.50	254.30 ± 0.12
OpenVLA-OFT	4:4	64	256	174.58 ± 0.39	196.99 ± 0.40	224.38 ± 0.45

4.3 CPU-Intensive LIBERO Evaluation

Table 3 complements the GPU-intensive ManiSkill experiments with 8-GPU LIBERO measurements. *D-VLA* achieves the highest throughput in every row, with gains over RLinf ranging from 20.83% to 76.99%. On this CPU-intensive simulator, the source of benefit differs from the GPU-physics contention motivating our design: the asynchronous Swimlane pipeline overlaps rollout with actor updates and reduces scheduling and serialization stalls, rather than resolving the same on-GPU physics contention.

4.4 Mechanism-Level Ablations

We evaluate three mechanisms separately: the weight-sync backend for synchronization stability, dual-pool memory management for maximum stable environment concurrency, and in-process observation access for transfer latency. Appendix B provides the complete configurations and measurements.

Weight-sync backend. Replacing Gloo with NCCL in the tested $\pi_{0.5}$ production configuration produces a first-step deadlock at `cudaDeviceSynchronize` when communication shares the reset synchronization path. The process times out with exit code 124 before completing the first step. With Gloo, training completes with exit code 0; the delayed-peer synchronization probe takes about 0.1 ms, and synchronization overhead before `env.reset` is approximately 0.02–0.16 ms per step. The result establishes stability for this schedule.

Dual-pool memory. With partitioning enabled, Torch-reserved memory at rollout end decreases from 39,286 to 33,388 MiB, and free device memory increases from 19,588 to 25,486 MiB (about 5.8 GiB). Maximum stable concurrency increases from 4,352 to 4,608 environments, or from 544 to 576 per GPU. Without partitioning, 4,480 environments fail during PhysX buffer creation; with partitioning, 4,864 environments reach a PyTorch out-of-memory limit. Thus the partition raises the capacity ceiling.

In-process observation access. For batches of 32, 64, and 128 observations (4.6, 9.2, and 18.4 MiB), zero-copy access takes 7.26, 7.48, and 7.51 μs, compared with 691, 697, and 706 μs for the measured

Table 4: $\pi_{0.5}$ scaling on ManiSkill with GRPO, a 3:1 split, and the reported 192-environment/GPU budget. Throughput (Thr.) is in env/s.

GPUs	Global batch	*D-VLA*		RL-VLA^3		RLinf	
		Thr.	Eff.	Thr.	Eff.	Thr.	Eff.
8	1536	192.94	100.00%	130.64	100.00%	94.03	100.00%
16	3072	376.23	97.50%	248.22	95.00%	174.07	92.56%
32	6144	714.84	92.62%	446.80	85.50%	296.34	78.79%
64	12288	1406.65	91.13%	854.27	81.74%	550.20	73.14%
128	24576	2720.29	88.12%	1608.27	76.94%	1002.92	66.66%
256	49152	5352.70	86.70%	3058.47	73.16%	1866.03	62.02%

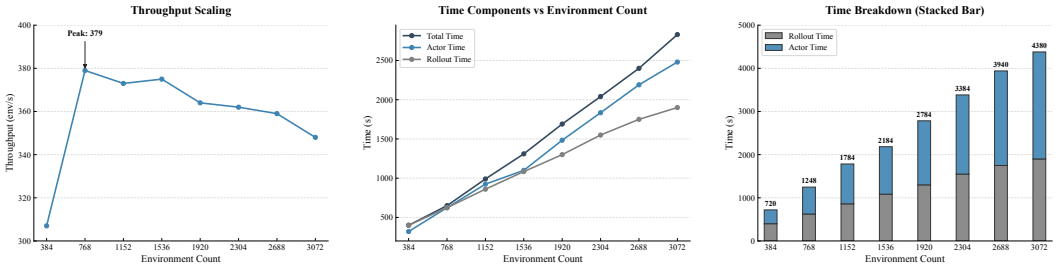

Figure 7: Fixed-hardware environment-count sweep for $\pi_{0.5}$ with a 3:1 rollout-side:actor allocation. Throughput is in env/s; step, rollout, and actor times are seconds per training iteration.

NCCL transfer path. The zero-copy path costs approximately 1.1% of the transfer latency and changes little with batch size. These measurements quantify observation-access latency in the co-located data path.

4.5 GPU-Count Scaling and Environment-Count Saturation

Multi-node scaling. Table 4 evaluates $\pi_{0.5}$ with GRPO and a 3:1 split from 8 to 256 GPUs. The reported environment setting is 192 per GPU, the global batch size is $192G$ for G total GPUs, and the mini-batch size is 192. Scaling efficiency relative to 8 GPUs is defined in Equation 1. With a $32\times$ increase in GPU count, *D-VLA* retains 86.70% efficiency, compared with 73.16% for RL-VLA^3 and 62.02% for RLinf. The evaluated systems differ in communication locality and scheduling: *D-VLA* keeps environment–rollout traffic within each node and synchronizes the model across actor workers; the fully disaggregated RL-VLA^3 layout permits observation traffic across nodes, while synchronous RLinf exposes stage waiting.

$$\mathrm{Eff}(G) = \frac{\mathrm{Thr}(G)}{\mathrm{Thr}(8)\,(G/8)} \times 100\%. \tag{1}$$

Environment-count operating range on fixed hardware. A separate $\pi_{0.5}$ sweep increases environments from 384 to 3,072 while holding hardware and the 3:1 resource allocation fixed (Figure 7). Throughput peaks near 379 env/s at 768 environments, then decreases toward 360 env/s at 3,072. Around the peak, rollout and actor execution times are approximately balanced under the fixed GPU allocation. The sweep identifies the throughput peak for this configuration and shows saturation as the environment count increases beyond it.

4.6 GRPO and PPO Comparison

We compare both objectives on ManiSkill PutOnPlateInScene25Main-v3 using $\pi_{0.5}$, 384 training environments, a global batch size of 2,048, and 300 training iterations. Runs start from the same SFT checkpoint, with shared model and optimization settings aligned across frameworks. Appendix C lists the shared settings and objective-specific differences.

Table 5: GRPO/PPO comparison on ManiSkill PutOnPlateInScene25Main-v3 with $\pi_{0.5}$. Throughput is reported as mean $\pm$ standard deviation (env/s). Success is training-trajectory success, with mean $\pm$ standard deviation over the final 50 training steps in the reported converged segment near steps 250–300.

Algorithm	Framework	Throughput ↑	Success (%) ↑
GRPO	RLinf-co	230.3 ± 2.1	58.84 ± 5.04
GRPO	RL-VLA^3	264.6 ± 2.9	63.77 ± 3.58
GRPO	*D-VLA*	306.8 ± 3.7	78.69 ± 2.12
PPO	RLinf-co	175.4 ± 1.7	73.66 ± 1.68
PPO	RL-VLA^3	179.5 ± 1.3	80.00 ± 1.59
PPO	*D-VLA*	183.6 ± 0.8	86.34 ± 1.50

Under GRPO, *D-VLA* improves throughput by 33.22% over RLinf-co and training success by 19.85 percentage points. Under PPO, the corresponding improvements are 4.68% and 12.68 percentage points. These results demonstrate throughput and training-success improvements with both GRPO and PPO in *D-VLA* .

5 Conclusion and Future Work

D-VLA separates trajectory transfer and weight synchronization, overlaps execution with a four-thread pipeline, and combines memory partitioning with local observation access. Experiments with 3B and 7B VLA models cover GPU-intensive ManiSkill and CPU-intensive LIBERO. The targeted ablations demonstrate synchronization stability, increased memory capacity, and reduced transfer latency; end-to-end experiments demonstrate throughput benefits with GRPO and PPO. A separate $\pi_{0.5}$ scaling experiment retains 86.70% efficiency at 256 GPUs relative to 8 GPUs.

Future work. We plan to extend *D-VLA* to multi-agent interaction and additional physics engines, and evaluate its behavior under a broader range of policy staleness.

Acknowledgement

This work is supported by the Natural Science Foundation of Guangdong Province (Grant No. 2025A1515011946)

References

[1] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR, 06–09 Nov 2023.

[2] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[3] Gemini Robotics Team. Gemini Robotics 1.5: Pushing the Frontier of Generalist Robots with Advanced Embodied Reasoning, Thinking, and Motion Transfer. *arXiv e-prints*, page arXiv:2510.03342, October 2025.

[4] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[5] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. Smolvla: A vision-language-action model for affordable and efficient robotics, 2025.

[6] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[7] Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision-language-action models for embodied ai. *arXiv preprint arXiv:2405.14093*, 2024.

[8] Yifan Zhong, Fengshuo Bai, Shaofei Cai, Xuchuan Huang, Zhang Chen, Xiaowei Zhang, Yuanfei Wang, Shaoyang Guo, Tianrui Guan, Ka Nam Lui, et al. A survey on vision-language-action models: An action tokenization perspective. *arXiv preprint arXiv:2507.01925*, 2025.

[9] Dapeng Zhang, Jing Sun, Chenghui Hu, Xiaoyan Wu, Zhenlong Yuan, Rui Zhou, Fei Shen, and Qingguo Zhou. Pure vision language action (vla) models: A comprehensive survey. *arXiv preprint arXiv:2509.19012*, 2025.

[10] Remi Cadene, Simon Aliberts, Francesco Capuano, Michel Aractingi, Adil Zouitine, Pepijn Kooijmans, Jade Choghari, Martino Russi, Caroline Pascal, Steven Palma, et al. Lerobot: An open-source library for end-to-end robot learning. *arXiv preprint arXiv:2602.22818*, 2026.

[11] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297, 2025.

[12] Jian Hu, Xibin Wu, Zilin Zhu, Weixun Wang, Dehao Zhang, Yu Cao, et al. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*, 6, 2024.

[13] Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, et al. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint arXiv:2506.13585*, 2025.

[14] Zhenpeng Su, Leiyu Pan, Xue Bai, Dening Liu, Guanting Dong, Jiaming Huang, Minxuan Lv, Wenping Hu, Fuzheng Zhang, Kun Gai, et al. Klear-reasoner: Advancing reasoning capability via gradient-preserving clipping policy optimization. *arXiv preprint arXiv:2508.07629*, 2025.

[15] Hongzhi Zang, Mingjie Wei, Si Xu, Yongji Wu, Zhen Guo, Yuanqing Wang, Hao Lin, Liangzhi Shi, Yuqing Xie, Zhexuan Xu, et al. Rlinf-vla: A unified and efficient framework for vla+ rl training. *arXiv preprint arXiv:2510.06710*, 2025.

[16] Zhong Guan, Haoran Sun, Yongjian Guo, Shuai Di, Xiaodong Bai, Jing Long, Tianyun Zhao, Mingxi Luo, Chen Zhou, Yucheng Guo, et al. Rl-vla3: Reinforcement learning vla accelerating via full asynchronism. *arXiv preprint arXiv:2602.05765*, 2026.

[17] Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, et al. Simplevla-rl: Scaling vla training via reinforcement learning. *arXiv preprint arXiv:2509.09674*, 2025.

[18] Bin Xie, Erjin Zhou, Fan Jia, Hao Shi, Haoqiang Fan, Haowei Zhang, Hebei Li, Jianjian Sun, Jie Bin, Junwen Huang, et al. Dexbotic: Open-source vision-language-action toolbox. *arXiv preprint arXiv:2510.23511*, 2025.

[19] Mustafa Shukor Dana Aubakirova, Jade Cholgari, and Leandro von Werra. Vlab: Your laboratory for pretraining vlas. `https://github.com/huggingface/vlab`, 2025.

[20] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[21] Chen Zhou, Haoran Sun, Hedan Yang, Jing Long, Junwu Xiong, Luqiao Wang, Mingxi Luo, Qiming Yang, Shuai Di, Song Wang, et al. Thousand-gpu large-scale training and optimization recipe for ai-native cloud embodied intelligence infrastructure. *arXiv preprint arXiv:2603.11101*, 2026.

[22] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[23] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. pi 0.5: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[24] T Mu, Z Ling, F Xiang, D Yang, X Li, S Tao, Z Huang, Z Jia, and H Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. In *35th Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[25] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

[26] Wei Gao, Yuheng Zhao, Tianyuan Wu, Shaopan Xiong, Weixun Wang, Dakai An, Lunxi Cao, Dilxat Muhtar, Zichen Liu, Haizhou Zhao, et al. Rollart: Scaling agentic rl training via disaggregated infrastructure. *arXiv preprint arXiv:2512.22560*, 2025.

[27] Rui Yu, Shenghua Wan, Yucen Wang, Chen-Xiao Gao, Le Gan, Zongzhang Zhang, and De-Chuan Zhan. Reward models in deep reinforcement learning: A survey. *arXiv preprint arXiv:2506.15421*, 2025.

[28] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

A Experimental Protocol

Reporting scope. The 16-GPU ManiSkill and 8-GPU LIBERO measurements report means and standard deviations across six seeds. LIBERO explicitly reports sample standard deviation: for measurements $x_1, \ldots, x_6$, $\bar{x} = \sum_{i=1}^{6} x_i/6$ and $s = \sqrt{\sum_{i=1}^{6}(x_i - \bar{x})^2/5}$.

Hardware configuration. The reported hardware is H200 (141 GB) with InfiniBand for inter-node communication.

B Mechanism-Level Measurement Details

B.1 Weight Synchronization Stability

Table 6 replaces the weight-synchronization backend while retaining the tested production scheduling path for $\pi_{0.5}$. With NCCL, the shared synchronization path creates a circular dependency between weight synchronization and rollout-data progress at the first reset. The NCCL run fails with a timeout before completing the first step. With Gloo, the host-side communication thread can progress without waiting on the same CUDA path.

Table 6: Weight-sync backend comparison in the tested $\pi_{0.5}$ configuration. The probe measures `cudaDeviceSynchronize` with delayed peer progress; pre-reset overhead is synchronization time before `env.reset`.

Backend	Probe	Exit code	Pre-reset overhead
Gloo	$\sim$0.1 ms	0 (completes)	0.02–0.16 ms/step
NCCL	Deadlock	124 (timeout)	No completed step

B.2 Memory Capacity

The memory experiment targets 4,608 training environments (576 per GPU) and 1,024 evaluation environments (128 per GPU), with `predict_mbs=64`, `force_lazy_eval=true`, and an ON-phase fraction of 0.24. Table 7 reports Torch-reserved memory, free device memory, and exit status for the target configuration, alongside the maximum stable concurrency from the capacity sweep. Torch-reserved memory measures allocator reservation.

Table 7: Dual-pool partitioning. Memory readings are at rollout end (MiB). Exit code is for the target 4,608-environment configuration; maximum stable concurrency and the next reported failing configuration come from the capacity sweep.

Partition	Exit code	Torch reserved	Free	Max env	Env/GPU	Reported failure beyond capacity
OFF	1	39286	19588	4352	544	4480: PhysX buffer creation
ON	0	33388	25486	4608	576	4864: PyTorch OOM

B.3 Observation Transfer Latency

Table 8 measures observation access within a co-located process against the tested NCCL transfer path. The former avoids transporting the observation payload between separate components. These measurements describe the complete measured paths, including their access/transfer overhead.

Table 8: Observed latency of the in-process zero-copy and NCCL observation paths. Ratios are computed from the reported latencies.

Batch	Observation (MiB)	Zero-copy (μs)	NCCL (μs)	Ratio
32	4.6	7.26	691	1.05%
64	9.2	7.48	697	1.07%
128	18.4	7.51	706	1.06%

C GRPO and PPO Configuration

The task is ManiSkill PutOnPlateInScene25Main-v3 with GPU simulation and RGB plus segmentation observations. Shared settings are `max_episode_steps=80`, `num_action_chunks=5` (16 chunk samples per trajectory), and the `pi05` model configured with `flow_noise`, `noise_params =[0.16,0.12,200]`, and `num_steps=4`. All runs start from the same SFT checkpoint, use 300 training steps, and disable evaluation and checkpointing during training. The policy optimizer uses a learning rate of 7.91×10^{-6}, Adam betas $(0.9, 0.95)$, and gradient clipping at 1.0; global batch size is 2,048. Table 9 summarizes the batch configurations for RLinf and *D-VLA* .

Table 9: Reported environment and batch settings for the GRPO/PPO comparison. Objective-specific differences are kept the same between RLinf and *D-VLA* .

Setting	RLinf GRPO	RLinf PPO	*D-VLA* GRPO	*D-VLA* PPO
Training environments	384	384	384	384
Evaluation environments	128	240	128	240
Group size	8 (48 groups)	1	8 (48 groups)	1
Micro-batch size	64	64	64	64
Update epochs	1	2	1	2

Table 10: Objective-specific hyperparameters. These differ between GRPO and PPO, rather than between the frameworks within an objective.

Parameter	GRPO	PPO
Advantage estimator	GRPO	GAE ($\gamma = 0.99$, $\lambda = 0.95$)
Entropy bonus	0.005	0.005
Policy clipping (lower/upper)	0.2 / 0.2	0.2 / 0.2
Value clipping	—	0.2
Huber parameter	—	10
Reward filtering	True, $[0.0, 1.0]$	False
Value learning rate	—	1.55×10^{-4}
Adam epsilon	10^{-8}	10^{-5}

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes],

   Justification: Alignment between Claims and Methodology: The introduction claims to address the systematic bottlenecks in VLA training—specifically the resource contention between physics simulation and large-scale model optimization. This is directly and rigorously addressed in Section 3 through the proposed "Plane Decoupling" architecture and the four-thread "Swimlane" asynchronous pipeline. The technical implementation provides a concrete solution to the high-level problems identified at the beginning of the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations regarding pipeline symmetry at extreme scales, the need for dynamic resource reallocation, and the current scope of single-agent scenarios in the "Discussion" section 5.

   Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: We provide details on architecture, hyperparameters, and data splits in Abstract and the supplemental material."

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [No]

   Justification: We provide the full source code in the supplementary material, including scripts for environment setup, model training, and evaluation. Detailed hyperparameters, hardware configurations (16 GPUs),

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Anonymized code and data are included in the supplemental material

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Full experimental details, including hyperparameters and optimization settings are provided in Experiment Settings.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer:[Yes]

   Justification: All results are reported with mean and standard deviation over 6 random seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: we provide hardware specs and training GPUs in experinment settings.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research follows all NeurIPS ethical guidelines.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Discussed in introduction and conclusion.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our model is for research in simulation and does not pose high misuse risks.

    Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All used datasets (ManiSkill) and codes are properly cited and used under their respective licenses.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new codebase is documented and provided in the supplementary material.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

Justification: [N/A]

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification:[N/A]

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.