WHEN FORESIGHT DRIFTS: PROGRESS ALIGNED RESYNCHRONIZATION FOR CLOUD–EDGE ROBOTIC CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Deploying billion-parameter vision-language-action (VLA) policies on mobile robots creates a systems tension: semantic reasoning benefits from cloud GPUs, whereas closed-loop control requires responsive local execution. We identify two sources of foresight drift in cloud–edge robotic control. First, uniform temporal prediction allocates representational capacity by elapsed duration rather than semantic significance, creating a pace–progress mismatch that underrepresents critical procedural transitions. Second, physical changes can invalidate predicted continuations even when the semantic stage remains relevant, creating a state-foresight mismatch. We propose Foresight Resync (ForRes), which integrates semantic-target-conditioned cloud prediction, progress-based edge cache addressing, and value-guided asynchronous refresh. In the cloud, a JEPA-based predictor forecasts the next semantic milestone from causal execution history, and a lightweight adapter conditions the world action model on this target while preserving its native prediction budget. On the edge, recurrent addressing matches observed execution to cached semantic anchors, retrieving latent guidance according to procedural progress rather than elapsed time. Target-support gating suppresses incompatible guidance, while an alignment-conditioned critic requests refresh when its expected benefit exceeds querying costs; invalid or exhausted caches trigger refresh directly. Edge policy maintains observation-only closed-loop control while awaiting cloud responses. Evaluated with two cloud backbones and an X-VLA edge policy, ForRes achieves up to 78.73% average success on RoboTwin 2.0, improving over Latent-to-Action by 4.20 percentage points with 31.11% fewer cloud calls, and reaches 76.00% success on real-world dual-arm tasks.

1 INTRODUCTION

World Action Models provide crucial predictive foresight for robotic control by capturing environmental dynamics and long-horizon scene evolution, enabling general language-conditioned manipulation Black et al. (2024); Bi et al. (2025); Cen et al. (2025); Chen et al. (2025b). Their growing scale, however, makes deployment on power- and weight-constrained robots challenging, while dynamic interactions require timely control updates grounded in the latest local observations Xu et al. (2024); Sendai et al. (2025); Wang et al. (2026e). This tension motivates an edge–cloud embodied architecture: powerful cloud WAMs provide long-horizon predictive guidance, while lightweight edge policies translate this guidance into actions and continually adjust execution using fresh observations, coupling cloud foresight with responsive local feedback Chen et al. (2025a).

Transferring future-state latents allows a single cloud prediction to guide multiple local action updates, amortizing cloud computation and communication across execution Kehoe et al. (2015); Zheng et al. (2026a). However, as shown in Figure 1, cloud foresight can remain misaligned with the semantic structure and physical conditions of edge execution. We term this phenomenon foresight drift and distinguish two sources. First, cloud predictions typically sample the future at uniform temporal intervals, whereas manipulation is understood and organized around semantic subtasks whose importance is not proportional to their duration. Approaching a cup may take five time steps, while establishing a grasp may occur in less than one. Yet this brief event determines whether the robot

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

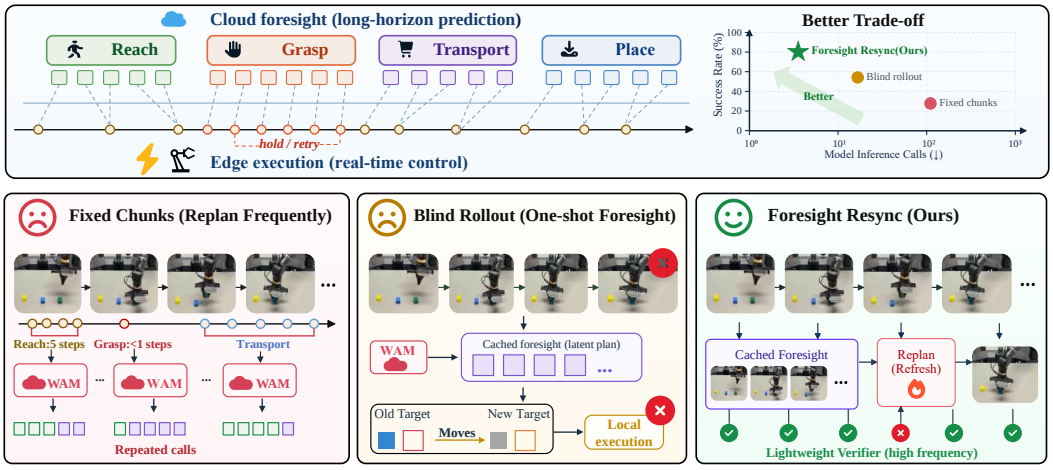


Figure 1: Foresight drift in edge–cloud control. Execution may diverge from the predicted temporal schedule, while physical changes can invalidate foresight even after progress alignment.

can proceed to transport the cup and receives particular attention in human descriptions of the task. Uniform temporal sampling can devote multiple predictions to the approach while failing to explicitly capture grasp establishment. This constitutes a *pace–progress mismatch*: temporal coverage does not ensure coverage of the semantic events that govern task progress. Cloud foresight should therefore be directed toward these events to better support successful edge execution. Second, even when foresight captures the relevant semantic event, execution deviations or environmental changes can make the predicted continuation incompatible with the current physical state. For example, a relocated placement target can invalidate a predicted continuation while the robot remains in the same transport stage. This constitutes a *state–foresight mismatch*. Inference and communication delays increase opportunities for physical divergence, but prediction age alone does not determine whether foresight remains usable.

Existing methods largely follow two directions.As shown in Figure 1, execution-oriented approaches adapt action horizons or compensate for delayed guidance, but do not explicitly allocate cloud prediction capacity around critical semantic events Wang et al. (2026d); Ye et al. (2026); Peng et al. (2026). Event-oriented approaches encode historical progress or verify imagined task events, but do not jointly address progress-aligned cache retrieval and cost-aware asynchronous refresh Wang et al. (2026b;c); Chen et al. (2025a). These limitations expose a gap between predicting relevant futures and maintaining their utility during execution. Adjusting execution timing cannot recover milestones omitted from prediction, while capturing an event does not ensure continued physical compatibility Liu et al. (2026); Ni et al. (2026). The two drift modes therefore require distinct, complementary mechanisms. Semantic-target-conditioned prediction anchors cloud rollouts to critical events that uniform sampling may miss, while progress-based addressing connects the resulting guidance to ongoing execution. Physical incompatibility requires continually checking whether cached guidance remains valid and refreshing or suppressing it when necessary.

We investigate synchronizing cloud foresight with robot execution along two axes: semantic task progress and physical state consistency. We propose *ForRes*, which combines semantic-target-conditioned prediction, progress-aligned cache addressing, and value-guided asynchronous refresh. In the cloud, a JEPA-based predictor forecasts the next semantic milestone from causal execution history. Lightweight adaptation conditions a frozen WAM on this target, while event-weighted sampling concentrates its prediction budget around critical procedural transitions. Future latents are cached alongside semantic keys for edge retrieval. On the edge, recurrent addressing integrates observation history, proprioception, and alignment memory to retrieve guidance matching actual procedural progress rather than elapsed time. Target-support gating assesses whether the cached target and guidance remain physically compatible with ongoing execution. An alignment-conditioned continuation critic compares the expected returns of cache reuse and delayed refresh against querying costs, while invalid or exhausted caches trigger replanning directly. Training with varied cache ages and guidance dropout enables X-VLA to maintain observation-only closed-loop control during

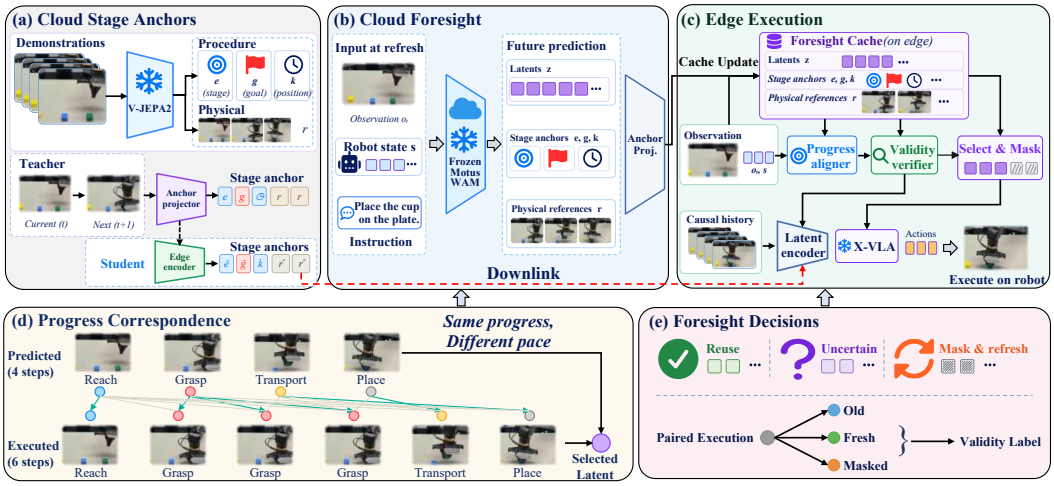


Figure 2: **Overview of ForRes.** A JEPA-based milestone predictor conditions cloud WAM foresight on semantic targets. Progress-aligned addressing retrieves cached latents, while target-support gating and a continuation critic regulate guidance reuse and asynchronous refresh. Guidance conditions X-VLA for closed-loop control, with observation-only execution during refresh.

cloud refresh. On RoboTwin 2.0, ForRes reaches 78.73% average success, outperforming Latent-to-Action by 4.20 percentage points with 31.11% fewer cloud calls.

2 Related Work

2.1 Edge–Cloud Embodied Control

Deployment-oriented research examines hardware scaling and optimizes inference partitioning across edge and cloud resources Taherin et al. (2026); Zheng et al. (2026a;b). Hierarchical and dual-system architectures organize planning and reactive execution, providing architectural foundations for cloud–edge coordination Chen et al. (2026); Gao et al. (2026); Hirose et al. (2026). Complementary representation-learning approaches strengthen spatiotemporal understanding and visual–physical grounding, offering capabilities relevant to local control rather than distributed execution itself Ni et al. (2026); Feng et al. (2026). Latency-resilient methods address delayed guidance through spatial correction, delay-robust features, or speculative waypoint buffering with execution verification Yang et al. (2026); Peng et al. (2026); Nguyen et al. (2026). These directions address deployment efficiency, control representations, and delayed-output adaptation.

2.2 Adaptive Execution and Task Progress

Adaptive execution methods replace fixed action horizons with decisions informed by motion phases, attention, or uncertainty Nie et al. (2026); Wang et al. (2026a); Liang et al. (2026). Complementary state representations support long-horizon control through progress-supervised event memory and spatial grounding, respectively Wang et al. (2026b); Li et al. (2026a). Verification-based approaches assess predicted event consistency, future–reality agreement, or observed task progress to guide action selection and replanning Wang et al. (2026c;d); Ye et al. (2026). Efficient future-conditioning methods reuse predictive context across asynchronous action updates or denoising steps, reducing repeated world-model computation Cai et al. (2026); Zhao et al. (2026). Building on these directions, ForRes addresses synchronization of reusable cloud foresight with asynchronous edge execution: semantic milestones structure prediction, procedural correspondence governs cache retrieval, and physical compatibility together with continuation value determines reuse and refresh.

3 Problem Formulation

A language instruction ℓ specifies the manipulation task. At control step t, the robot receives visual observations o_t and proprioceptive state s_t. The causal history $\mathcal{H}_t$ contains observations and

states available through t, together with actions issued before t. At each action-update boundary, a pretrained edge VLA generates a chunk $A_t = a_{t:t+h-1} \sim \pi_{\theta_0}(\cdot \mid o_t, s_t, \ell)$, where a_t is a robot action, h is the local action horizon, and θ_0 denotes the pretrained edge-policy parameters. The pretrained cloud WAM exposes a future-state prediction interface. For a request issued at step τ, a target predictor and a conditioned WAM produce

$$\hat{g}_\tau = F_\psi(\mathcal{H}_\tau, \ell), \quad \hat{Z}^f_\tau = W_{\omega_0,\Delta\omega}\left(o_\tau, s_\tau, \ell, P_\rho(\hat{g}_\tau)\right) = \left[z^f_{\tau,1}, \ldots, z^f_{\tau,N}\right]. \tag{1}$$

F_ψ predicts the next semantic-event representation $\hat{g}_\tau$, and P_ρ maps it into WAM conditioning features. Pretrained parameters ω_0 remain frozen; $\Delta\omega$ denotes trainable lightweight adaptation. Each $z^f_{\tau,i}$ is a predicted future-state latent, and N is the fixed number of prediction slots per request. Prediction slots follow chronological, event-weighted progress coordinates rather than fixed physical-time intervals. Training targets include the event boundary, but predictions need not realize the intended event. The response arrives after d_τ control steps for consideration at the next action-update boundary. The delay d_τ is distinguished from its causal estimate $\hat{d}_t$ available when issuing a request. Let $\mathcal{C}_\tau$ denote the latest accepted cache, which is empty before initialization. At each action-update boundary, the edge-side addressing, selection, and refresh mechanism Γ_ϑ uses $\mathcal{H}_t$, ℓ, $\mathcal{C}_\tau$, the request status ν_t, and the estimated delay $\hat{d}_t$ to produce an ordered list of candidate guidance indices $\mathcal{S}_t$ and a binary request decision $r_t \in \{0, 1\}$. Here, ϑ collects the mechanism's trainable parameters; Γ_ϑ is not an additional network. The indicator $\nu_t \in \{0, 1\}$ equals one while a cloud request is pending, and $r_t = 1$ initiates a request only when $\nu_t = 0$. The adapted policy generates

$$A_t \sim \pi_{\theta_0,\Delta\theta}\left(\cdot \mid o_t, s_t, \ell, U_t\right), \tag{2}$$

where $\Delta\theta$ denotes trainable edge-policy adapters. The tokens U_t encode selected guidance when the cache is used and are empty during observation-only control. The system-level objective is

$$\max_{\Theta} \mathbb{E}_{\xi\sim P_\Theta}\left[R_{\text{task}}(\xi) - c_{\text{req}} \sum_{t\in\mathcal{T}_{\text{upd}}} r_t\right], \tag{3}$$

where Θ collects all trainable parameters, P_Θ is the trajectory distribution induced by the system and environment, and $R_{\text{task}}(\xi)$ is the task return of executed trajectory ξ. The set $\mathcal{T}_{\text{upd}}$ contains action-update steps. The request penalty c_{req} represents communication and computation cost in return units under a fixed per-request generation budget. Response delay affects execution and task return without incurring another penalty. This objective motivates modular supervised training without implying end-to-end optimization or global optimality. Deployment uses causal inputs only.

4 METHOD

As shown in Figure 4, ForRes couples JEPA-conditioned cloud foresight with progress-aligned edge retrieval. Target gating and a continuation critic regulate guidance reuse and refresh, while X-VLA executes locally with observation-only fallback during replanning.

4.1 CLOUD: SEMANTIC-TARGET-CONDITIONED FORESIGHT

At cloud query step τ, JEPA-based predictor predicts the representation of the next procedural milestone, $\hat{g}_\tau = F_\psi(\mathcal{H}_\tau, \ell)$, conditioned on causal execution history $\mathcal{H}_\tau$ and language instruction ℓ. During training, demonstration interaction signals identify the next unachieved semantic event and its completion boundary $e(t) > t$. A frozen target encoder constructs $g^\star_t = E_{\text{tar}}(o_{e(t)})$, with features pooled over a temporal boundary window when state transitions are continuous. The target predictor is optimized in a normalized feature space:

$$\mathcal{L}_{\text{stage}} = \mathbb{E}_{(\mathcal{H}_t,\ell,g^\star_t)}\left[\left\|F_\psi(\mathcal{H}_t, \ell) - \text{sg}(g^\star_t)\right\|_2^2\right], \tag{4}$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operator. Observations associated with the same upcoming event instance share its target representation, while trajectory segments without a reliable completion boundary are excluded Feng et al. (2026). Privileged predicates and future horizons are used only for offline supervision. An auxiliary head sharing representations with F_ψ predicts event types,

interacting object roles, and task-schema stages, using cross-entropy supervision from demonstration descriptors Guo et al. (2026). The predicted descriptor is included in metadata μ_τ, enabling semantic eligibility checks independently of latent similarity. Uniform temporal sampling can underrepresent brief but critical contact phases. Let $\mathcal{B}_t \subseteq \{t+1, \dots, e(t)\}$ contain frames within b control steps of an annotated physical transition, such as contact initiation, grasping, or release. For $\mathcal{B}_t \neq \varnothing$, we define event-weighted sampling over N generation slots:

$$w_{t,u} = \frac{1-\alpha}{e(t)-t} + \frac{\alpha\,\mathbf{1}[u \in \mathcal{B}_t]}{|\mathcal{B}_t|}, \quad u_{t,i} = \min\left\{ u \in \{t+1, \dots, e(t)\} : \sum_{v=t+1}^{u} w_{t,v} \geq \frac{i}{N} \right\}. \tag{5}$$

When $\mathcal{B}_t = \varnothing$, we use uniform weights $w_{t,u} = 1/(e(t)-t)$. The mixing coefficient $\alpha \in (0,1)$ and boundary radius b are selected on validation data. The positive, normalized weights ensure that the final training sample lies at the event boundary, $u_{t,N} = e(t)$; this sampling property does not guarantee successful event completion in generated rollouts. Each coordinate i/N therefore indexes an event-weighted progress quantile rather than a fixed elapsed time. A lightweight adapter conditions the cloud WAM on the predicted milestone:

$$\hat{Z}^f_\tau = W_{\omega_0, \Delta\omega}\left(o_\tau, s_\tau, \ell, P_\rho(\hat{g}_\tau)\right), \tag{6}$$

where o_τ and s_τ denote the current observation and robot state, respectively, and $\hat{Z}^f_\tau = (z^f_{\tau,i})_{i=1}^N$ is the predicted future latent sequence. Continuous progress coordinates i/N replace conventional time embeddings. The adapter P_ρ, coordinate embeddings, and low-rank parameters $\Delta\omega$ are trained using the backbone's native generative loss, while the base parameters ω_0 remain frozen. During training, conditioning gradually shifts from teacher-forced targets $g^\star_t$ to predictions $\hat{g}_t = F_\psi(\mathcal{H}_t, \ell)$. Each future latent $z^f_{\tau,i}$ is paired with an indexing key $k_{\tau,i} = P_\kappa(z^f_{\tau,i}, \hat{g}_\tau)$. The semantic foresight cache $\mathcal{C}_\tau$ stores these N ordered key–latent pairs, the predicted milestone $\hat{g}_\tau$, the root anchor $k_{\tau,0} = E_{\text{ref}}(o_\tau)$, and metadata μ_τ. The reference visual encoder E_{ref} maps observations into the same embedding space as the projected keys. Alongside the predicted semantic descriptor, μ_τ records the dispatch time, coordinate bounds, and request identifiers. Exposing latent representations and indexing keys allows the edge policy to use cloud foresight without image decoding, while maintaining local control at its own update rate.

4.2 Edge: Semantic Addressing of Cached Futures

To align asynchronous predictions with execution without additional cloud rollouts, a recurrent query module combines causal observation history, current proprioception, instruction ℓ, memory m_t, and cached anchor features. The memory summarizes preceding causal execution and addressing updates. The module predicts a correspondence distribution p_t over the source anchor, N future slots, and a no-match outcome, indexed from 0 to $N+1$. A confident match selects an ordered list $\mathcal{S}_t$ of up to K subsequent latent indices for conditioning. Matching the source anchor begins selection at slot 1; ambiguous correspondence or exhausted coverage yields $\mathcal{S}_t = \emptyset$. Demonstration augmentation uses temporal compression and stationary repetitions, preserving phase labels. Generated caches receive supervision from offline-decoded sequences rather than inheriting labels from demonstration step $u_{t,i}$. Annotations distinguish approach, contact, grasp stability, transport, release, object roles, and event occurrence. The target distribution y^{addr}_t assigns equal probability to anchors matching the query's annotated phase and event occurrence. When no anchor matches, it assigns unit probability to the no-match class; uncertain pairs are excluded. Let $\mathcal{D}_{\text{addr}}$ denote the distribution of annotated query-cache pairs. The addressing module minimizes

$$\mathcal{L}_{\text{addr}} = -\mathbb{E}_{\mathcal{D}_{\text{addr}}}\left[\sum_{i=0}^{N+1} y^{\text{addr}}_t(i) \log p_t(i)\right]. \tag{7}$$

Physical displacement preserves procedural correspondence when the semantic phase remains unchanged. Key projection regularization minimizes the expected squared Euclidean distance between keys computed by P_κ from $z^f_{\tau,i}$ and $\hat{g}_\tau$, and stop-gradient reference embeddings extracted by E_{ref} from offline-decoded observations $\tilde{o}_{\tau,i}$. A lightweight target-support gate verifies the physical compatibility of retrieved guidance:

$$b_t = B_\zeta(\mathcal{H}_t, \ell, \mathcal{C}_\tau), \qquad \mathcal{L}_{\text{sup}} = \mathbb{E}\left[\text{BCE}(b_t, y^{\text{sup}}_t)\right]. \tag{8}$$

The binary label y_t^{sup} indicates whether the cached target matches the next uncompleted event and its guidance supports physically feasible completion under the edge policy. Labels are constructed offline using semantic-event annotations and execution outcomes under cached guidance. When b_t falls below a calibrated safety threshold, the edge sets $\mathcal{S}_t = \emptyset$, revoking invalid guidance and alerting the refresh scheduler. A parallel head $C_\eta(\mathcal{H}_t, \ell, \hat{g}_\tau)$, parameterized by η, uses binary event-completion supervision to estimate milestone completion. Completion retires the cache; subsequent requests target the next uncompleted event unless the task is finished.

4.3 Edge: Continuation Value and Asynchronous Refresh

With valid guidance $\mathcal{S}_t$, the edge makes a continuation-or-refresh decision by comparing finite-horizon execution utilities. An alignment-conditioned critic estimates the execution return under both choices:

$$(Q_t^c, Q_t^r) = Q_\beta \left(\mathcal{H}_t, \ell, \mathcal{C}_\tau, p_t, m_t, \hat{d}_t \right), \tag{9}$$

where Q_t^c and Q_t^r denote the expected returns of continuing with the current cache and requesting fresh cloud foresight, respectively, and $\hat{d}_t$ is the causally estimated round-trip delay. To train Q_β, we generate counterfactual paired rollouts initialized from identical simulator checkpoints, execution histories, and cache contexts. Branch c continues execution with cached guidance, whereas branch r initiates an asynchronous cloud request and uses observation-only control during the simulated delay window d. Both branches terminate upon goal completion, critical failure, or horizon exhaustion. For each branch $b \in \{c, r\}$, the realized return J_t^b balances milestone attainment y_{complete}^b, continuous progress ΔP_t^b, failure penalty y_{fail}^b, and duration T_t^b via non-negative weights w_{comp}, w_{prog}, w_{fail}, and w_{time}. The critic is supervised by the realized returns of the paired branches:

$$\mathcal{L}_{\text{critic}} = \mathbb{E}_{\mathcal{D}_{\text{branch}}} \left[(Q_t^c - J_t^c)^2 + (Q_t^r - J_t^r)^2 \right]. \tag{10}$$

Here, Q_t^c and Q_t^r are finite-horizon continuation utilities rather than global Markovian state values. At each action-chunk boundary, the edge requests refresh only when the predicted improvement, $Q_t^r - Q_t^c$, exceeds the request cost c_{req}, which accounts for communication and cloud computation. If the task is unfinished and the cache is absent, invalid, or exhausted, refresh bypasses value gating. All requests require $\nu_t = 0$. During round-trip latency, the robot continues closed-loop observation-only control while suppressing duplicate requests. A newly arrived cache is verified against the latest causal history $\mathcal{H}_t$ before activation, so responses that have already become obsolete are discarded.

4.4 Latent-Conditioned Control and Policy Adaptation

A lightweight future-latent encoder E_ϕ maps the ordered cached latents indexed by $\mathcal{S}_t$ into conditioning tokens U_t. When guidance is suppressed or invalid, or a refresh request is pending, we set $U_t = \varnothing$. The edge policy is trained on expert action chunks with optional foresight conditioning:

$$\mathcal{L}_{\text{act}} = \mathbb{E}_{(\mathcal{H}_t, o_t, s_t, \ell, A_t^\star) \sim \mathcal{D},\, U_t \sim \mathcal{Q}(\cdot \mid \mathcal{H}_t, \ell)} \left[\ell_{\text{base}} \left(\pi_{\theta_0, \Delta\theta}(\cdot \mid o_t, s_t, \ell, U_t), A_t^\star \right) \right], \tag{11}$$

where $\mathcal{D}$ is the demonstration distribution, $A_t^\star$ is the expert action chunk beginning at t, and ℓ_{base} is the policy's native action-training loss. The distribution $\mathcal{Q}$ samples conditioning tokens through E_ϕ from causal cache contexts, including empty guidance. We adapt the policy through a two-stage curriculum: we first train E_ϕ with the base policy backbone frozen, and subsequently optimize E_ϕ jointly with lightweight action adapter parameters $\Delta\theta$, preserving the pretrained representations of the multimodal visual encoder. The sampling distribution $\mathcal{Q}$ exposes the policy to varied cache ages and stochastic guidance dropout, enforcing stable observation-only fallback when $U_t = \varnothing$. Crucially, robot actions remain strictly supervised by high-frequency teleoperated demonstrations paired with native sensory observations. World-model forecasting runs asynchronously in the cloud, while addressing, gating, and value evaluation occur at local action-update boundaries. This separation amortizes cloud computation without blocking observation-driven edge control.

5 Experiments

We evaluate ForRes across simulated and physical manipulation environments, focusing on task execution, cloud foresight synchronization, runtime efficiency, and the individual contributions of semantic target generation, progress cache addressing, and continuation value refresh.

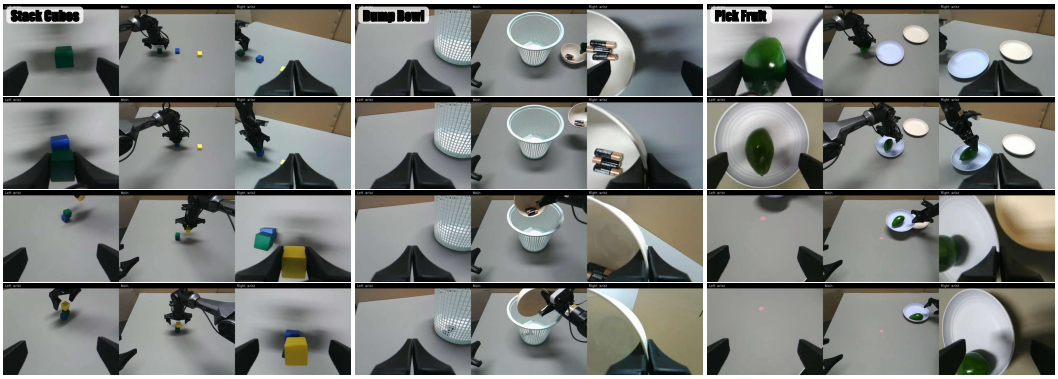


Figure 3: Real-world manipulation tasks on the Aloha-AgileX platform. From top to bottom: Dump Bowl, Pick Fruit, and Stack Cubes.

5.1 Experimental Setup

Benchmarks. We report task-averaged success on RoboTwin 2.0 Chen et al. (2025b) under clean and randomized settings, and assess foresight synchronization under execution-rate changes, network delays, and object disturbances. Real-world evaluation uses an Aloha-AgileX dual-arm platform with one overhead and two wrist-mounted RGB cameras, plus proprioceptive sensing.

Cloud WAMs and edge policy. We employ LingBot-VA Li et al. (2026b) and Motus Bi et al. (2026) as cloud WAM backbones, with X-VLA serving as the unified edge execution policy. Each WAM uses an independent future-latent encoder to project its native latent representations into conditioning tokens for X-VLA. We preserve the native temporal sampling and local action-chunk horizons of each cloud model: LingBot-VA yields one latent covering 16 action steps, while Motus yields one latent covering 8 action steps. Both models generate two future latents per refresh request, providing 32 steps of nominal coverage for LingBot-VA and 16 steps for Motus. To isolate synchronization capability without introducing confounding policy variations, all collaborative methods share the identical lightweight X-VLA backbone. We also report standalone baselines from our prior study, including RDT, π_0, and direct cloud execution.

Metrics. We evaluate system performance across five dimensions: task success rate, cloud utilization, local control efficiency, network transmission, and onboard deployment footprint. Task performance is quantified by success rates across clean, randomized, and real-world trials, alongside the recovered performance gap between standalone edge policies and cloud models. Cloud utilization is assessed by average requests per episode and the percentage reduction relative to periodic refresh:

$$R_{\text{call}} = \left(1 - \frac{\overline{N}_{\text{method}}}{\overline{N}_{\text{fixed}}}\right) \times 100\%, \tag{12}$$

where $\overline{N}$ aggregates initialization and subsequent queries across both completed and interrupted trials. We also record total cloud GPU seconds per episode. Control efficiency is tracked via onboard action chunk generation latency, end-to-end response latency, and amortized inference compute time. Communication profiling measures payload volume per request and cumulative bidirectional bandwidth. Edge deployment cost includes peak GPU memory, deployed parameter counts, and trainable parameter fractions.

Benchmark Performance on RoboTwin 2.0. Table 1 compares cloud-only WAMs, edge-only VLAs, foresight transfer paradigms, and ForRes across cloud backbones with X-VLA as the common edge policy. ForRes outperforms standalone edge policies and our Latent-to-Action framework with either LingBot-VA or Motus as the cloud WAM. Motus with X-VLA increases the success rate from 54.50% to 78.73%, closing 72.69% of the performance gap between standalone X-VLA and cloud-only Motus. Under the randomized setting where execution pace and physical states drift from planned trajectories, it improves the success rate from 39.00% to 71.36%, surpassing Latent-to-Action by 6.32 percentage points. LingBot-VA with X-VLA elevates average task success from 54.50% to 75.48% and randomized success from 39.00% to 66.84%, recovering 55.65% of the performance gap. ForRes reduces remote queries from 4.5 to 3.1 calls per episode on LingBot-VA and

Method	Cloud	Edge	Clean	Rand.	Avg.	Gap Cl.	Calls/Ep.
Standalone Baselines from Prior Work							
Cloud-only WAM	LingBot-VA	–	92.90%	91.50%	92.20%	–	9.0
	Motus	–	88.66%	87.02%	87.84%	–	18.0
	X-WAM	–	89.80%	90.70%	90.25%	–	14.4
Edge-only VLA	–	RDT	34.50%	13.72%	24.11%	–	0.0
	–	π_0	46.42%	16.34%	31.38%	–	0.0
	–	X-VLA	70.00%	39.00%	54.50%	–	0.0
Edge–Cloud Collaborative Foresight Transfer							
Latent-to-Language	LingBot-VA	X-VLA	75.94%	42.88%	59.41%	13.02%	4.5
	Motus	X-VLA	77.18%	41.28%	59.23%	14.19%	9.0
Latent-to-Action	LingBot-VA	X-VLA	81.98%	60.32%	71.15%	44.16%	4.5
	Motus	X-VLA	84.02%	65.04%	74.53%	60.08%	9.0
ForRes (Ours)	LingBot-VA	X-VLA	**84.12%**	**66.84%**	**75.48%**	**55.65%**	**3.1**
	Motus	X-VLA	**86.10%**	**71.36%**	**78.73%**	**72.69%**	**6.2**

Table 1: Task success rates on RoboTwin2.0. Edge Params denotes the number of parameters deployed on the edge. Avg. Gain denotes the relative improvement in average success rate over the corresponding edge-only VLA.

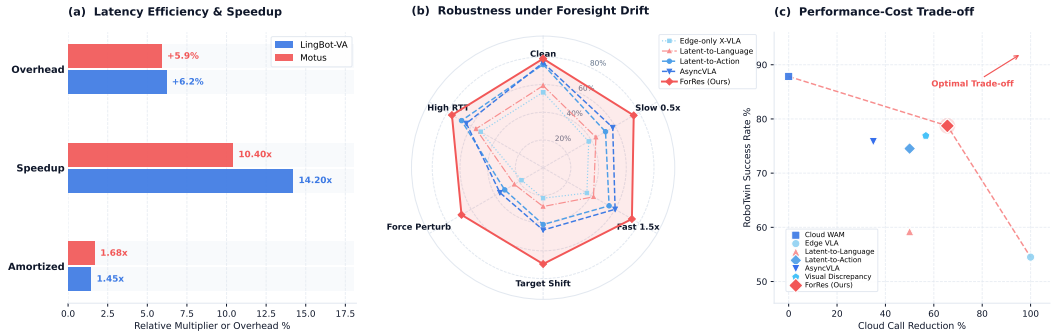


Figure 4: **Efficiency and robustness of ForRes.** (a) Speedup and overhead. (b) Success under execution disturbances and latency. (c) Success versus cloud-call reduction relative to cloud-only execution, showing improved performance–cost trade-offs.

from 9.0 to 6.2 calls on Motus, cutting communication frequency by 31.11% on both backbones. These results show progress-aligned resynchronization and value-guided refresh resolve foresight drift while preserving responsive closed-loop control on the edge.

Qualitative Real-World Evaluation. Figure 3 shows three manipulation tasks through successive observations from an external camera and two wrist-mounted cameras. In *Stack Cubes*, the robot sequentially grasps and places colored cubes to form a three-cube tower, requiring precise alignment during placement. In *Dump Bowl*, it lifts a bowl, positions it above a basket, and tilts it to empty the contents, coordinating grasp maintenance with orientation changes. In *Pick Fruit*, it picks up a green fruit, places it into a bowl, and subsequently lifts and transports the bowl. These sequences illustrate grasping, placement, and object transport across stages with changing object configurations and visibility.

Physical Dual-Arm Manipulation Results. As shown in Table 2, ForRes achieves an average success rate of 76.00%, outperforming edge-only X-VLA by 28.00 percentage points. Notably, the benefit of progress aligned foresight synchronization becomes more pronounced as task complexity increases. ForRes improves over X-VLA by 12.0 percentage points on *Dump Bowl*, 28.0 percentage points on *Pick Fruit*, and 44.0 percentage points on the most challenging *Stack Cubes* task. It matches Motus on *Dump Bowl*, exceeds it by 2.0 percentage points on *Pick Fruit*, and completely matches the 76.00% average success rate of the cloud-only model. These results demonstrate that

ForRes consistently improves real-world manipulation performance, with particularly substantial gains on complex multi-stage tasks.

5.2 Ablation Studies

We examine component contributions using Motus as the cloud WAM and X-VLA as the edge policy. Table 3 reports clean and randomized success rates alongside average cloud requests per episode. ForRes achieves 78.73% average success with 6.2 requests, improving over Latent-to-Action by 4.20 percentage points while reducing cloud calls by 31.11%.

Task	Motus	X-VLA	ForRes	Gain
Dump Bowl	94.0	82.0	**94.0**	**+12.0**
Pick Fruit	82.0	56.0	**84.0**	**+28.0**
Stack Cubes	52.0	6.0	**50.0**	**+44.0**
Avg.	76.00	48.00	**76.00**	**+28.00**

Table 2: Real-world success rates on the Aloha-AgileX platform. All methods use 50 demonstrations per task and are evaluated over 20 matched physical trials.

Semantic Targets and Event-Weighted Sampling. Replacing the next-event semantic target with a fixed-horizon target reduces average success from 78.73% to 75.65%, with a larger drop under randomized conditions (71.36% to 67.20%) than under clean conditions (86.10% to 84.10%). This indicates that conditioning foresight on the milestone benefits execution beyond predicting a fixed horizon. Removing event-weighted sampling reduces average success to 77.15%, despite an unchanged cloud request count of 6.2. The 1.58-percentage-point difference supports allocating the prediction budget around critical transitions.

Progress-Aligned Semantic Addressing. Replacing semantic addressing with sequential time-indexed retrieval decreases average success from 78.73% to 75.80%. The degradation is more pronounced under randomized conditions, where success falls by 3.96 percentage points, compared with 1.90 percentage points under clean conditions. Meanwhile, cloud calls remain similar at 6.3 versus 6.2 per episode. These results support matching cached guidance to observed execution progress rather than advancing through predictions solely according to elapsed time. Continuation-Value-Guided Refresh. Replacing the continuation critic with periodic refresh lowers average success from 78.73% to 76.10% while increasing cloud calls from 6.2 to 9.0 per episode. The full framework improves randomized success by 4.16 percentage points and uses 31.11% fewer requests than this variant. The results support conditioning refresh decisions on the estimated utility of cached foresight, improving task success while reducing remote query overhead.

Configuration	Clean	Rand.	Avg.	Calls
ForRes (Full)	**86.10%**	**71.36%**	**78.73%**	**6.2**
w/o Stage Target	84.10%	67.20%	75.65%	6.4
w/o Event Sampling	85.30%	69.00%	77.15%	6.2
w/o Semantic Addressing	84.20%	67.40%	75.80%	6.3
w/o Continuation Critic	85.00%	67.20%	76.10%	9.0
Latent-to-Action Base	84.02%	65.04%	74.53%	9.0

Table 3: Ablation results on RoboTwin 2.0 using LingBot-VA and X-VLA. Metrics report task success rate percentages and remote query counts per episode.

Cloud–edge component combination. We evaluate the progression from Latent-to-Action to the full framework. Introducing semantic generation alone improves average success from 74.53% to 76.10% while retaining periodic refresh. Deploying semantic addressing and the continuation critic on unconditioned WAM outputs elevates success to 77.35% while lowering query frequency from 9.0 to 6.4 calls per episode. Unifying components within ForRes delivers 78.73% average success with 6.2 requests per episode. This confirms that milestone-directed prediction and progress-aligned retrieval provide complementary benefits for asynchronous embodied control.

6 Conclusion

ForRes addresses cloud–edge foresight drift through semantic-target-conditioned prediction, progress-aligned retrieval, and value-guided refresh. Simulation and real-world experiments demonstrate improved task success, with fewer cloud queries in simulation, supporting foresight alignment with semantic progress and physical conditions.

AI USE STATEMENT

During the preparation of this manuscript, we used generative AI tools to assist with language editing, proofreading, and improving the clarity and consistency of the written presentation. Generative AI was also used to support selected mathematical proofs in the appendix, including checking algebraic steps and clarifying their exposition. We independently reviewed and verified all AI-assisted text and proofs and take full responsibility for the final content of this work.

ETHICS STATEMENT

Knowledge editing can improve the reliability of large language models but may also introduce risks, including misinformation, biased outputs, and unintended behavioral changes if used carelessly. Our experiments rely on publicly available code, datasets, and models and follow their stated licenses and terms of use. We recommend validating edited facts, assessing side effects, and monitoring model behavior in controlled deployments.

REFERENCES

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. *arXiv preprint arXiv:2512.13030*, 2025.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35101–35113, 2026.

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.

Jisong Cai, Long Ling, Shiwei Chu, Zhongshan Liu, Jiayue Kang, Zhixuan Liang, Wenjie Xu, Yinan Mao, Weinan Zhang, Xiaokang Yang, et al. Aha-wam: Asynchronous horizon-adaptive world-action modeling with observation-guided context routing. *arXiv preprint arXiv:2606.09811*, 2026.

Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, et al. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

Hao Chen, Jiaming Liu, Chenyang Gu, Zhuoyang Liu, Renrui Zhang, Xiaoqi Li, Xiao He, Yandong Guo, Chi-Wing Fu, Shanghang Zhang, et al. Fast-in-slow: a dual-system vla model unifying fast manipulation within slow reasoning. *Advances in Neural Information Processing Systems*, 38: 98049–98083, 2026.

Jiahong Chen, Jing Wang, Long Chen, Chuwei Cai, and Jinghui Lu. Nanovla: Routing decoupled vision-language understanding for nano-sized generalist robotic policies. *arXiv preprint arXiv:2510.25122*, 2025a.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025b.

Yicheng Feng, Wanpeng Zhang, Ye Wang, Hao Luo, Haoqi Yuan, Sipeng Zheng, and Zongqing Lu. Spatial-aware vla pretraining through visual-physical alignment from human videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 712–723, 2026.

Chongkai Gao, Zixuan Liu, Zhenghao Chi, Junshan Huang, Xin Fei, Yiwen Hou, Yuxuan Zhang, Yudi Lin, Zhirui Fang, and Lin Shao. Vla-os: Structuring and dissecting planning representations and paradigms in vision-language-action models. *Advances in Neural Information Processing Systems*, 38:136705–136736, 2026.

Jun Guo, Qiwei Li, Peiyan Li, Zilong Chen, Nan Sun, Yifei Su, Heyun Wang, Yuan Zhang, Xinghang Li, and Huaping Liu. Unified 4d world action modeling from video priors with asynchronous denoising. *arXiv preprint arXiv:2604.26694*, 2026.

Noriaki Hirose, Catherine Glossop, Dhruv Shah, and Sergey Levine. Asyncvla: An asynchronous vla for fast and robust navigation on the edge. *arXiv preprint arXiv:2602.13476*, 2026.

Ben Kehoe, Sachin Patil, Pieter Abbeel, and Ken Goldberg. A survey of research on cloud robotics and automation. *IEEE Transactions on automation science and engineering*, 12(2):398–409, 2015.

Chengmeng Li, Junjie Wen, Yaxin Peng, Yan Peng, and Yichen Zhu. Pointvla: Injecting the 3d world into vision-language-action models. *IEEE Robotics and Automation Letters*, 11(3):2506–2513, 2026a.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026b.

Yuanchang Liang, Xiaobo Wang, Kai Wang, Shuo Wang, Xiaojiang Peng, Haoyu Chen, David Kim Huat Chua, and Prahlad Vadakkepat. Adaptive action chunking at inference-time for vision-language-action models. *arXiv preprint arXiv:2604.04161*, 2026.

Zhen Liu, Xinyu Ning, Zhe Hu, Xinxin Xie, Weize Li, Zhipeng Tang, Chongyu Wang, Zejun Yang, Hanlin Wang, Yitong Liu, et al. Goal2skill: Long-horizon manipulation with adaptive planning and reflection. *arXiv preprint arXiv:2604.13942*, 2026.

Chanh Nguyen, Shutong Jin, Florian T Pokorny, and Erik Elmroth. Speculative policy orchestration: A latency-resilient framework for cloud-robotic manipulation. *arXiv preprint arXiv:2603.19418*, 2026.

Chaojun Ni, Cheng Chen, Xiaofeng Wang, Zheng Zhu, Wenzhao Zheng, Boyuan Wang, Tianrun Chen, Guosheng Zhao, Haoyun Li, Zhehao Dong, et al. Swiftvla: Unlocking spatiotemporal dynamics for lightweight vla models at minimal overhead. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13474–13485, 2026.

Junnan Nie, Jiayi Li, Jiachen Zhang, Junyi Lao, Chenghao Liu, Tianle Zhang, and Songfang Huang. Pace: Phase-aware chunk execution for robot policies with action chunking. *arXiv preprint arXiv:2606.00537*, 2026.

Daojie Peng, Fulong Ma, Bingtao Wang, Sheng Wang, and Jun Ma. Latency-tolerant cloud-edge collaborative vision-language-action models via emergent representational specialization. *arXiv preprint arXiv:2608.00569*, 2026.

Kohei Sendai, Maxime Alvarez, Tatsuya Matsushima, Yutaka Matsuo, and Yusuke Iwasawa. Leave no observation behind: Real-time correction for vla action chunks. *arXiv preprint arXiv:2509.23224*, 2025.

Amir Taherin, Juyi Lin, Arash Akbari, Arman Akbari, Pu Zhao, Weiwei Chen, David Kaeli, and Yanzhi Wang. Cross-platform scaling of vision-language-action models from edge to cloud gpus. In *Proceedings of the Great Lakes Symposium on VLSI 2026*, pp. 234–239, 2026.

Haoxuan Wang, Gengyu Zhang, Yan Yan, Ramana Rao Kompella, and Gaowen Liu. Vla knows its limits: Adaptive execution horizons for robot policies. *arXiv preprint arXiv:2602.21445*, 2026a.

Kai Wang, Zhaopeng Gu, Yixiang Chen, Yuan Xu, Qisen Ma, Jiabing Yang, Zhaowen Li, Yan Huang, Liang Wang, and Peng Su. Dim-wam: World-action modeling with diverse historical event memory. *arXiv preprint arXiv:2606.27677*, 2026b.

Kailin Wang, Haoxiang Jie, Yaoyuan Yan, Jiacheng Zhou, and Zhiyou Heng. Ev-wm: Event-verified world models for long-horizon robotic manipulation. *arXiv e-prints*, pp. arXiv–2606, 2026c.

Rui Wang, Yue Zhang, Jiehong Lin, Kuncheng Luo, Jianan Wang, Zhongrui Wang, and Xiaojuan Qi. When to trust imagination: Adaptive action execution for world action models. *arXiv preprint arXiv:2605.06222*, 2026d.

Zihua Wang, Zhitao Lin, Ruibo Li, Yu Zhang, Xu Yang, Siya Mi, and Xiu-Shen Wei. Open-loop planning, closed-loop verification: Speculative verification for vla. *arXiv preprint arXiv:2604.02965*, 2026e.

Zhiyuan Xu, Kun Wu, Junjie Wen, Jinming Li, Ning Liu, Zhengping Che, and Jian Tang. A survey on robotics with foundation models: toward embodied ai. *arXiv preprint arXiv:2402.02385*, 2024.

Kai Yang, Zedong Chu, Yingnan Guo, Zhengbo Wang, Shichao Xie, Yanfen Shen, Xiaolong Wu, Xing Li, and Mu Xu. Asyncshield: A plug-and-play edge adapter for asynchronous cloud-based vla navigation. *arXiv preprint arXiv:2604.24086*, 2026.

Feng Ye, Yiming Zhao, Yong Yu, Hongxu Zhou, Yong Pan, Yuan Xue, Peng Jia, and Chuanmin Jia. Rethink before you execute: Adaptive execution for world action models. *arXiv preprint arXiv:2608.09492*, 2026.

Weiheng Zhao, Haoyi Jiang, Xin Shi, Liu Liu, Fan Huang, Zhizhong Su, Wei Sui, and Xinggang Wang. Faster-wam: Efficient inference-time future conditioning for robust world action models. *arXiv preprint arXiv:2608.04404*, 2026.

Zihao Zheng, Hangyu Cao, Jiayu Chen, Sicheng Tian, Chenyue Li, Maoliang Li, Xinhao Sun, Guojie Luo, and Xiang Chen. Roboecc: Multi-factor-aware edge-cloud collaborative deployment for vla models. *arXiv preprint arXiv:2603.20711*, 2026a.

Zihao Zheng, Sicheng Tian, Hangyu Cao, Chenyue Li, Jiayu Chen, Maoliang Li, Xinhao Sun, Hailong Zou, Guojie Luo, and Xiang Chen. Rapid: Redundancy-aware and compatibility-optimal edge-cloud partitioned inference for diverse vla models. *arXiv preprint arXiv:2603.07949*, 2026b.