

How Fast Can I Run My VLA?

A critical read of VLA-Perf (Jiang et al., NVIDIA Research, 2026)

Marcus Lam and Pulak Mehrotra

Source markers: [paper] VLA-Perf [π_0] the π_0 paper [n] reference n unmarked: our arithmetic

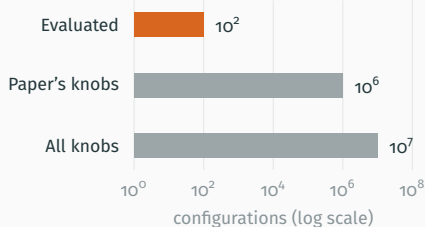
Find VLA designs that run in real time; the space is too big to measure

The paper's question: which combinations of model and system meet a latency target?^[paper]

THE KNOBS

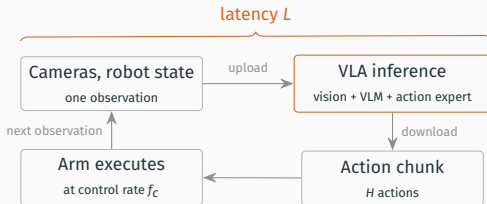
Model	System	Held fixed
Scale	GPU	Precision
Action generation	Placement	Software
Denoising steps	Network	Power
Chunk size		Network variance
Action DoF		Tenancy
Context		Accuracy
Cameras		
Sync / async		
Dual-system		

HOW MANY CONFIGURATIONS?



Building each system is “cost- and time-prohibitive”,^[paper] so the paper models instead, and evaluates about 10^2 of the 10^6 points.

The target: inference as fast as the camera



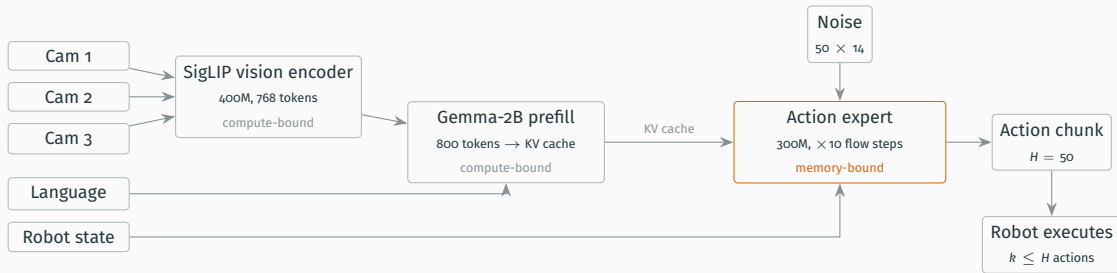
10 Hz “acceptable”

100 Hz “high-performance”^[paper]

Bracketing 24–60 fps cameras. Task success is out of scope.

Objective: latency. Target: camera rate. Accuracy: assumed held fixed.

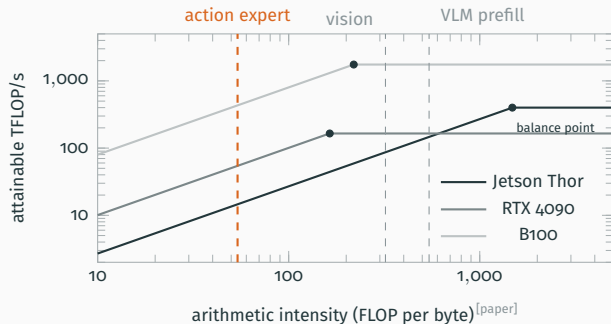
π_0 : the prefix runs once, compute-bound; the expert runs $10\times$, memory-bound



The prefix KV cache is computed once. Each of the 10 flow steps re-runs only the 300M expert, which was “downsized ... to speed up inference.”^[π_0]

As deployed: 50 Hz robots run 25 of the 50 actions open-loop, then re-infer. Measured on an RTX 4090: **73 ms** on-board, **86 ms** over Wi-Fi.^[π_0]

Roofline: left of a GPU's balance point is memory-bound; the expert always is



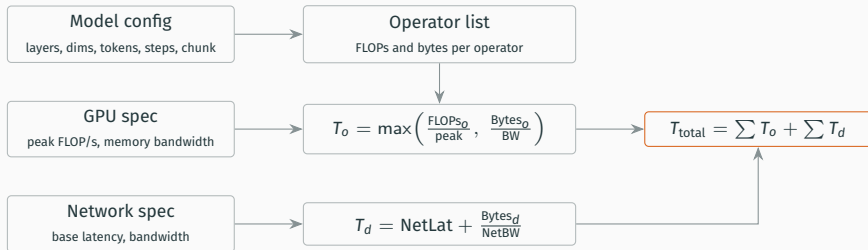
Per operator,

$$T = \max\left(\frac{\text{FLOPs}}{\text{peak}}, \frac{\text{bytes}}{\text{bandwidth}}\right).$$

- Vision, VLM: many tokens → compute-bound
- Action expert: few tokens → **memory-bound on every GPU**
- Thor: all memory-bound

Memory-bound means bytes decide latency. Precision cuts bytes; the paper never varies it.

VLA-Perf predicts latency from datasheet numbers alone; nothing is measured ^[paper]



No box for: precision software efficiency latency variance task success

Datasheet in, latency out: instant, free, and it covers hardware and models that do not exist yet.

Claimed contribution: a model for “arbitrary combinations” of VLA and system, 15 design rules, an open tool.

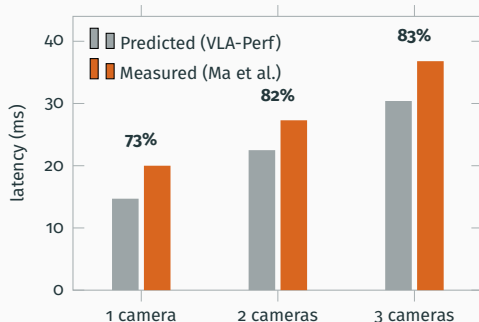
Validated once: one fast π_0 , one GPU, one precision^[1]

THE ONLY MEASURED SETUP

Model	π_0
GPU	RTX 4090
Precision	BF16
Code	hand-tuned Triton (fastest known)
Varied	cameras: 1, 2, 3

Fidelity: how close VLA-Perf's prediction comes to the real setup's latency.

= predicted $\div$ measured



All of the paper's ground truth: the best case, at three points.

What the paper predicts, bluntly^[paper]

MODEL

$\propto$ steps

Denoising steps drive latency; chunk size is nearly free.

$\propto$ size

Bigger model, proportionally slower.

10–100×

Diffusion beats autoregressive decoding.

1K vs. 100

Context steps that fit: datacenter vs. edge GPU.

DEPLOYMENT (π_0)

19 Hz

On-device (Thor): 10 Hz yes, 100 Hz no.

26 Hz

A 4090 over Wi-Fi beats on-device.

120 Hz

100 Hz needs a B100 and WiFi 7.

13.8×

Async multiplies cloud rate (4G: 3.7 → 50.5 Hz).

To go fast: fewer steps, diffusion, a fast GPU nearby, and async.

Each design rule rests on one held-fixed input: six inputs, six critiques

Every design rule passes through the same six steps. Each critique targets one.

	The paper...	We ask...	Critique
1	predicts latency with a roofline	Is it close to real systems?	C1 Fidelity
2	models only BF16, ideal software	Do other settings change the answer?	C2 Precision
3	adds one fixed network latency	Do real deployments behave that way?	C3 Environment
4	calls ≥ 10 Hz “real-time”	Is Hz the right measure of reacting?	C4 Reaction
5	recommends the faster option	What does it cost in success?	C5 Accuracy
6	generalizes from π_0	Do other architectures agree?	C6 Architecture

C1 Fidelity: 80% is the best case; real stacks land at 27–43%

PAPER CLAIMS

“Exceeding 80%” fidelity is sufficient for insights.^[paper]

ASSUMES

The tuned 4090 result stands for every stack and every GPU.

TRUE DELTA

A ceiling: typical code runs 2–4× slower than predicted.

The validation: hand-tuned Triton, RTX 4090

3 cameras

1 camera

Same RTX 4090, other code

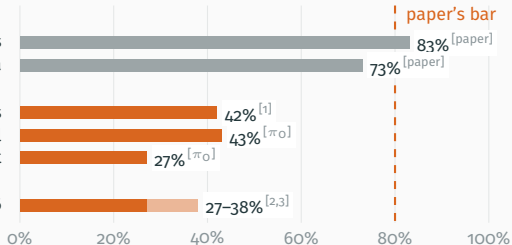
stock openpi, 2 cameras

π_0 's own code, whole model

π_0 's own code, action expert

Other GPU

Jetson Thor, $\pi_{0.5}$, BF16



Every untested knob (code, drivers, harness, GPU) only adds time: the predictions oversell.

C2 Precision: the server wins only at BF16; with FP8, Thor roughly ties

PAPER CLAIMS

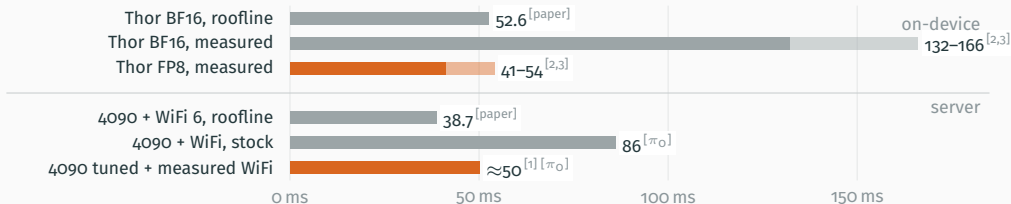
Server beats on-device; 100 Hz on Thor needs model-level changes. ^[paper]

ASSUMES

BF16 and ideal software, on a GPU where everything is memory-bound.

TRUE DELTA

With the best stack on each side, it is roughly a tie.



FP8 alone takes Thor from 132–166 ms to 41–54 ms: a precision change, not a model change.

C3 Network: one fixed number in the model; real Wi-Fi has a tail

PAPER CLAIMS

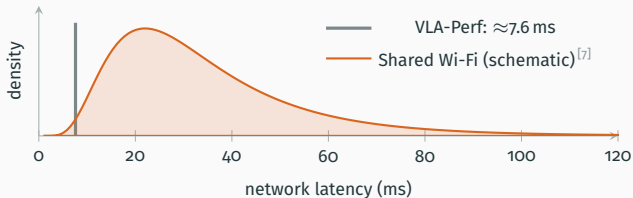
Network quality decides placement; 100 Hz needs Ethernet or WiFi 7. ^[paper]

ASSUMES

Fixed latency: no jitter, no contention, one dedicated GPU per robot.

TRUE DELTA

Rankings hold for the median of an idle network. Deployment risk lives in the p99.



Shared Wi-Fi: ~25 ms mean, up to 50 ms sd. ^[7] Rankings use the mean; robots live in the tail.

C4 Reaction (1/2): async keeps the arm moving by acting on old observations

PAPER CLAIMS

≥ 10 Hz is real-time; async multiplies Hz ($13.8\times$).^[paper]

ASSUMES

Higher Hz means a faster-reacting robot.

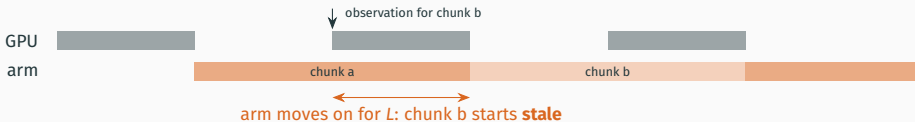
TRUE DELTA

Async raises Hz, not reaction time; latency L is the floor.

Sync arm waits for every inference: $\text{Hz} = 1/L$



Async next inference runs while the arm moves: Hz up to the GPU/network limit



Async buys Hz by running each chunk on an observation that is already L old.

C4 Reaction (2/2): in async, Hz no longer says how fast the robot reacts

Cloud B100 over 4G: async lifts π_o from 3.7 to 50.5 Hz,^[paper] but every action uses a view **273 ms** old.

FAILURE 1: STALENESS

Each chunk is planned for where the arm was L ago: Lf_c steps out of date (≈ 14 at 50 Hz). Naive chunk switching jerks out of distribution.^[8]

FAILURE 2: REACTION TIME

A new event waits for the next observation, then L more: Lf_c to $k + Lf_c$ steps. Async shortens neither term.

$\pi_o, f_c = 50 \text{ Hz}$	Paper's metric	Stale steps	Fastest reaction ($= L$)
RTX 4090 on-device	32 Hz	≈ 2	31 ms
B100 cloud over 4G, async	50.5 Hz	≈ 14	273 ms

Hz picks the cloud; reaction time picks the robot. Blanket “use async” hides both failures.

C5 Accuracy: assumed, so fewer steps and smaller models come for free

PAPER CLAIMS

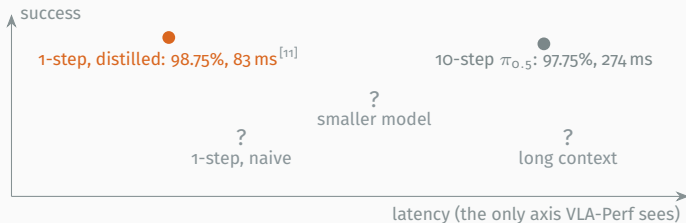
Reach 100 Hz with fewer steps or smaller models.^[paper]

ASSUMES

Models “meet the necessary accuracy thresholds” (§3).

TRUE DELTA

A latency menu: each option needs a success number.



Naive 1-step breaks success; distilled 1-step keeps it, after retraining the model cannot see.

C6 Architecture: the split verdict is a property of π_o , not of VLAs

PAPER CLAIMS

Device-server split is “unattractive”; dual-system is hypothetical. ^[paper]

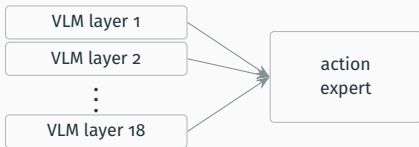
ASSUMES

Every action expert reads all 18 layers of VLM KV cache.

TRUE DELTA

Holds only for full-KV experts; an open dual-system exists.

π_o (VLA-Perf’s assumption): ≈ 14.7 MB



GRooT N1^[12]: ≈ 3.3 MB



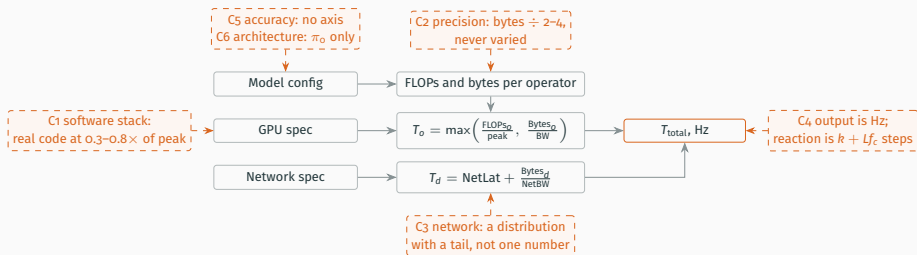
One-layer conditioning ships $4.5\times$ fewer bytes; the open dual-system was never checked.

Every gap has been measured; all but the network one already have a fix

	The gap, measured	The fix, where one exists
C1	Stock openpi π_0 on the same 4090: 53.7 ms, 42% of predicted ^[1]	CUDA graphs and kernel fusion bring it to 27.3 ms ^[1]
C2	Thor $\pi_{0.5}$ at BF16: 132–166 ms, about $3\times$ the roofline ^[2,3,6]	FP8 on Thor: 41–54 ms ^[2–4] ; 3-bit weights keep 94.8% success ^[5]
C3	A shared GPU adds 75% latency; $\mathcal{N}(10, 15)$ ms of injected delay costs 10 points of success ^[7]	Open: no p99-latency-vs-success curve for VLAs over 5G or cloud
C4	Naive async chunk switching gives jerky, out-of-distribution motion ^[8]	Inpaint the committed actions ^[8,9] ; report reaction latency directly, 54% lower ^[10]
C5	Naive 1-step $\pi_{0.5}$ degrades success on most tasks ^[11]	Distilled 1-step: 98.75% success at 83 ms, vs. 97.75% at 274 ms ^[11]
C6	An open dual-system already exists: GROOT N1, 10 Hz VLM, 120 Hz actions, 63.9 ms on an L40 ^[12]	Condition on one VLM layer: 3.3 MB across a split instead of 14.7 MB ^[12]

Most of the correction is already published. The model needs to take it as input.

The mechanisms hold; the latencies and verdicts built on them don't



HOLDS: THE MECHANISMS

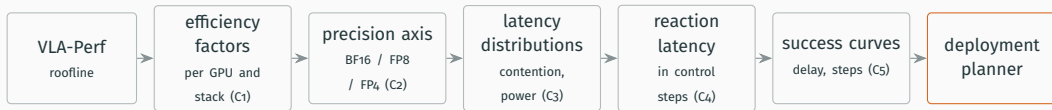
The expert is memory-bound, so steps cost latency and chunk size is free. Latency grows with parameters. Diffusion beats autoregressive decoding. Payload bytes decide splits.

DOESN'T HOLD AS STATED: THE VERDICTS

Every latency and Hz (best case). “Server beats on-device” (precision, tails). “100 Hz needs WiFi 7” (p99). “Async makes the cloud real-time” (reaction). “Splits are unattractive” (π_o only).

Use VLA-Perf to rule designs out and to read trends, not to quote a deployment latency.

VLA-Perf says what's impossible; five additions say what's deployable



VLA-Perf tells you what's *impossible*.
To know what's *deployable*, you need the five boxes it leaves out.

References i

1. Y. Ma, Y. Zhou, Y. Yang, T. Wang, H. Fan. *Running VLAs at Real-time Speed*. arXiv 2510.26742, 2025. <https://arxiv.org/abs/2510.26742>
2. NVIDIA Jetson AI Lab. *OpenPi $\pi_0.5$ on Jetson Thor* (measured latencies; vendor page). https://www.jetson-ai-lab.com/tutorials/openpi_on_thor/
3. Infinigence. *ApXInf v0.1 release notes* (Thor $\pi_0.5$ latency; project page). <https://github.com/infinigence/ApXInf/releases/tag/v0.1>
4. NVIDIA Developer Forums. *Real-Time Inference on Thor & RTX: P10.5/GROOT...* (community post, May 2026; least reliable). <https://forums.developer.nvidia.com/t/real-time-inference-on-thor-rtx-p10-5-groot-n1-6-1-7-thor-23-hz-rtx-5090-50-80hz/368788>
5. A. Akbari et al. *ActQuant: Sub-4-bit Action-Guided Quantization for VLA Models*. arXiv 2605.24011, 2026. <https://arxiv.org/abs/2605.24011>
6. A. Taherin et al. *Cross-Platform Scaling of Vision-Language-Action Models from Edge to Cloud GPUs*. arXiv 2509.11480, 2025. <https://arxiv.org/abs/2509.11480>
7. S. Pohland, X. Foukas, G. Ananthanarayanan, A. Kolobov, S. Mehrotra, B. Radunovic, A. Verma. *Offload or Overload: A Platform Measurement Study of Mobile Robotic Manipulation Workloads*. arXiv 2603.18284, 2026. <https://arxiv.org/abs/2603.18284>
8. K. Black, M. Y. Galliker, S. Levine. *Real-Time Execution of Action Chunking Flow Policies*. NeurIPS 2025. <https://arxiv.org/abs/2506.07339>
9. A. Agouzoul. *Understanding Asynchronous Inference Methods for Vision-Language-Action Models*. arXiv 2605.08168, 2026. <https://arxiv.org/abs/2605.08168>
10. Y. Guo, B. Liu. *Reflex: Real-Time VLA Control through Streaming Inference*. ICML 2026. <https://arxiv.org/abs/2607.14695>
11. W. Luan, J. Li, W. Zhao, W. Zhang, T. Wu, R. Ma. *SnapFlow: One-Step Action Generation for Flow-Matching VLAs via Progressive Self-Distillation*. arXiv 2604.05656, 2026. <https://arxiv.org/abs/2604.05656>
12. J. Bjorck et al. (NVIDIA). *GROOT N1: An Open Foundation Model for Generalist Humanoid Robots*. arXiv 2503.14734, 2025. <https://arxiv.org/abs/2503.14734>
13. K. Black et al. π_0 : A Vision-Language-Action Flow Model for General Robot Control. arXiv 2410.24164, 2024. <https://arxiv.org/abs/2410.24164>
14. W. Jiang, J. Clemons, K. Sankaralingam, C. Kozyrakis. *How Fast Can I Run My VLA? Demystifying VLA Inference Performance with VLA-Perf*. arXiv 2602.18397, 2026. <https://arxiv.org/abs/2602.18397>

Appendix

Backup (C4): what Hz misses, in three conditions

π_o : $f_c = 50$ Hz, $k = 25$, $L = 73$ ms.^[π_o] The paper's Hz metric checks only row 1.

Condition	In words	Formula	π_o
1 Keep up	next chunk arrives before this one runs out	$\text{rate} \geq f_c/k$	2 Hz, not 10
2 Staleness	chunk was planned for where the arm was L ago	Lf_c steps	≈ 4 steps
3 Reaction	wait for next observation, then inference	Lf_c to $k + Lf_c$	4–29 steps

Same $f_c = 50$ Hz	Paper's metric	Stale steps	Fastest reaction (= L)
4090 on-device (Tab. 3)	32 Hz	≈ 2	31 ms
B100 cloud over 4G, async (Tab. 8)	50.5 Hz	≈ 14	273 ms

Async raises row 1, not rows 2 and 3, so Hz and reaction time disagree on the ranking.

Backup: from a configuration to a prediction (π_0 on an RTX 4090)

RTX 4090 spec: 165 TFLOP/s (BF16), 1008 GB/s. ^[paper] Each component is either compute-bound or memory-bound, so one term dominates:

Component	Bound	Dominant term (back of envelope)	Ours	VLA-Perf ^[paper]
Vision encoder	compute	$2 \times 0.4\text{B params} \times 768 \text{ tokens} / 165 \text{ TFLOP/s}$	3.7 ms	4.0 ms
VLM prefill	compute	$2 \times 2\text{B params} \times 800 \text{ tokens} / 165 \text{ TFLOP/s}$	19.4 ms	19.8 ms
Action expert	memory	$10 \text{ steps} \times (0.3\text{B params} \times 2 \text{ B}) / 1008 \text{ GB/s}$	6.0 ms	7.3 ms
Total			29 ms	31.1 ms

A prediction is the configuration's FLOPs and bytes divided by the GPU's datasheet numbers.

Backup: how each knob enters the equations

Every result in §4 reruns the same equations with one input changed:

Knob	Input it changes	Effect on the π_0 prediction
GPU (4090 $\rightarrow$ Thor)	peak FLOP/s, bandwidth	memory-bound terms $\times 3.7$ (action 7.3 $\rightarrow$ ~ 27 ms)
Denoising steps (10 $\rightarrow$ 5)	action-expert passes	action term halves
Chunk size (50 $\rightarrow$ 250)	action-expert tokens	nearly free: weights read once per step
Cameras, model size	tokens, parameters	scales with tokens $\times$ params
Placement, network	adds NetLat + bytes/NetBW	image upload + action download

Measured times are never reused; every prediction assumes the same $\sim 80\%$ fidelity.

Backup: the knobs VLA-Perf can turn, and the ones it can't

MODEL KNOBS

Scale	2.7B, 9.1B, 16.7B, 81B
Action generation	diffusion (2 sizes), AR, AR-parallel
Denoising steps	1, 5, 10, 20, 50
Chunk size	5, 25, 50, 100, 250
Action DoF	7 to 42
Context	1 to 10K timesteps
Cameras	1, 2, 3
Execution	sync, async
Dual-system	off, 5 Hz, 10 Hz cap

SYSTEM KNOBS

GPU	Thor, 4090, A100, H100, B100
Placement	device, edge, cloud, split
Network	Ethernet, WiFi, 4G/5G, cloud

HELD FIXED

Precision	BF16 only
Software	ideal
Power	peak spec
Network	one fixed latency
Tenancy	1 GPU per robot
Accuracy	assumed adequate

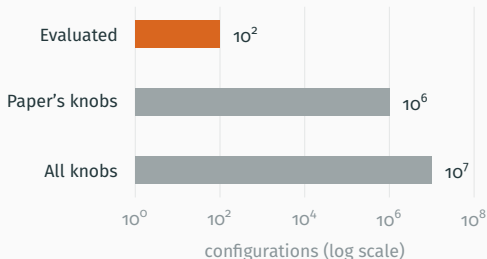
Each of these becomes a critique.

Backup: how big is the search space?

Model knobs ($4 \cdot 5 \cdot 5 \cdot 4 \cdot 6 \cdot 5 \cdot 3 \cdot 2 \cdot 3$)	$\approx 2 \times 10^5$
× valid GPU / placement / network	≈ 15
Paper's own knob space	$\approx 10^6$
× fixed knobs (precision, stack, power)	$\approx 10^7$
Points actually evaluated	$\approx 10^2$

One-factor-at-a-time sweeps around the π_0 default.
Interactions (fewer steps × async × 5G, precision × placement) are never explored.

Some combinations are invalid (AR ignores steps; out of memory), so 10^6 is an upper-order estimate.



Backup: what the paper gets right, by takeaway number

Mechanisms that follow from intensity and bytes moved, not from absolute numbers. No critique touches them.

Takeaway	Holds because...
T2	The action expert is memory-bound (decode-like); vision and VLM are compute-bound (prefill-like).
T6	So denoising steps cost latency, and chunk size is nearly free.
T3	Latency grows with parameters; memory capacity caps model size per GPU.
T7, T8	Diffusion beats autoregressive decoding, and the parallel-decoding baseline makes it a fair fight.
T9, T10, T11	<i>Mechanisms</i> : network time dominates server latency, payload size decides splits, async pipelines it all.

Backup: takeaways that don't hold as stated, by takeaway number

Takeaway	The paper says	Critique	True delta
All numbers	Latency and Hz in Tables 3–9	C1	Best case only: typical code runs $2\text{--}4\times$ slower
T1, T13	Edge is limited; 100 Hz needs model changes	C2	Precision suffices: FP8 Thor runs $\pi_{0.5}$ in 41–54 ms
T9	Server beats on-device	C2, C3	Best vs. best is a tie on the mean; tails favor the device
T14, T15	100 Hz needs Ethernet or WiFi 7	C3	Holds for an idle network's median; p99 decides
T11, T15	Async makes the cloud real-time	C4	Async raises Hz, not reaction time; L is the floor
T4, T5, T13	Scale up; 1K context; cut steps or size	C5	A latency menu: each option needs a success number
T10	Device–server split is unattractive	C6	Only for full-KV experts; one-layer ships $4.5\times$ fewer bytes
T12	Dual-system gives $1.05\text{--}2.26\times$	C6	Upper bound on an unbuilt system; GROOT N1 exists