WAMPTQ: BEYOND WEIGHT QUANTIZATION VIA TASK-UTILITY-GUIDED PRECISION ALLOCATION FOR EFFICIENT WORLD MODEL DEPLOYMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

World action models (WAMs) enable action-conditioned prediction and planning, but efficient deployment faces two challenges. Low-bit sensitivity varies across tasks, while reconstruction proxies do not directly measure native task utility. Prediction and context processing also create large transient tensors, so smaller packed artifacts need not reduce peak memory or latency. We introduce WAM post-training quantization (WAMPTQ), which couples task-utility-guided precision selection with liveness-aware execution. WAMPTQ constructs legal 4- and 8-bit weight maps under a matched packed-size budget and selects the final map using native task utility. An optional large language model (LLM) Agent narrows the legal proposal region before scoring. WAMPTQ then freezes the artifact and applies quantized epilogue fusion (QEF) and virtual-context consumer fusion (VCF) to remove avoidable rescaling and context-assembly intermediates. We separate within-pool selection error from candidate-coverage loss and derive an exact live-set criterion for when local savings reduce the global memory peak. Across 15 tasks from four WAM families, the largest observed quality drop relative to FP32 is 2.22 percentage points, with median end-to-end speedup and peak-memory reduction of $2.36\times$ and 71.5%, respectively. The code will be released.

1 INTRODUCTION

WAMs support action-conditioned prediction, latent planning, and control, enabling agents to assess candidate actions before execution (Zhou et al., 2025, Assran et al., 2025, Maes et al., 2026, Yuan et al., 2026). Larger models and repeated planning make persistent storage and transient runtime state key deployment costs. Post-training quantization (PTQ) reduces the former, while kernel fusion and memory-aware execution reduce the latter. Yet WAM tasks respond differently to the same precision assignment, and smaller packed artifacts need not lower peak memory when prediction and context intermediates remain live. Repeated rollouts amplify quantization-induced decision errors and temporary allocations, undermining layerwise objectives.

Existing PTQ methods choose or calibrate low-precision representations (Shao et al., 2024, Ashkboos et al., 2024, Liu et al., 2025, Li et al., 2026). Post-training quantization for generative pretrained transformers (GPTQ) uses approximate second-order reconstruction, activation-aware weight quantization (AWQ) exploits scaling, and SmoothQuant redistributes activation outliers (Frantar et al., 2023, Lin et al., 2024, Xiao et al., 2023). Hardware-aware automated quantization (HAQ), quantized bidirectional encoder representations from transformers (Q-BERT), and Hessian-aware quantization version 2 (HAWQ-V2) allocate bits using hardware feedback or sensitivity proxies (Wang et al., 2019, Shen et al., 2020, Dong et al., 2020), while task-aware approaches use downstream information in quantization objectives or sensitivity estimation (Niu et al., 2023, Xiao et al., 2025, Wang et al., 2026). WAM-specific studies report task- and granularity-dependent sensitivity and motivate rollout-aware calibration (Fu et al., 2026, Zhou et al., 2026). However, reconstruction error and local sensitivity need not preserve planning and control: small perturbations may change an action decision, while larger errors may not. We therefore target task utility at a fixed packed-size budget.

After precision selection, memory movement and tensor materialization can dominate inference cost, as shown by efficient attention, paging, low-bit kernels, and compiler fusion (Dao et al., 2022, Kwon et al., 2023, NVIDIA, 2026, Frantar et al., 2025, Guo et al., 2024, PyTorch Contributors, 2026, Li et al., 2025). Removing a local intermediate lowers the workload peak only if its saving reaches

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.

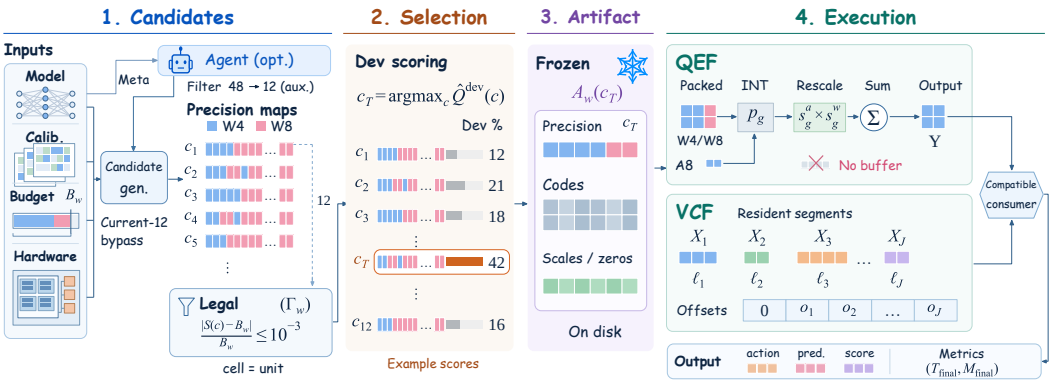

Figure 1: **WAMPTQ overview.** Native development tasks score legal W4/W8 maps. The selected artifact is frozen for QEF/VCF execution. A8 means 8-bit activations, and INT means integer arithmetic. Colored cells are eligible units. The optional Agent contracts unscored proposals, while Current-12 bypasses it. Labels abbreviate calibration (Calib.), development (Dev), generation (gen.), metadata (Meta), optional (opt.), auxiliary (aux.), and prediction (pred.).

the global maximum live set. Added workspace, extended lifetimes, or unchanged secondary peaks can absorb the reduction. Precision allocation therefore needs feedback from native tasks, while execution changes require information about live tensors across the complete workload.

WAMPTQ combines task-utility-guided precision selection with liveness-aware execution. As shown in Figure 1, it constructs legal 4- and 8-bit weight (W4/W8) maps under a matched packed-size budget and selects by native development utility. An optional LLM Agent reduces scoring cost by contracting proposals using only architecture metadata, calibration statistics, and quantization/hardware constraints, without development or final outcomes. After freezing the artifact, QEF rescales groups inside the packed producer and writes the required output data type (dtype), avoiding a full-sized intermediate. VCF exposes resident context segments directly to compatible consumers, avoiding contiguous concatenation. Execution can be tested independently of precision, preserving weight codes, scales, zero points, the map, task interface, and artifact.

Across 15 native-setting tasks from four WAM families, the complete route achieves 2.00–4.30× end-to-end speedup and 63.1–95.3% peak-memory reduction relative to FP32, while the largest observed quality drop is 2.22 percentage points. With the same 12 candidate maps, task feedback improves the frozen six-planning estimate by 1.38 points over Hessian selection, and the selection-aware aggregate remains positive. Fixed-artifact tests show that QEF and VCF reduce the global peak only when net local savings reach the peak live set, including a predicted null case. **Our contributions are:**

- We formulate budget-matched WAM mixed-precision selection using native task utility and give a finite-pool guarantee explaining why proxy-optimal maps need not be task-optimal.
- We introduce optional Agent-assisted proposal contraction before task scoring, preserving the selector and separating coverage loss from within-pool selection error.
- We introduce fixed-artifact QEF and VCF execution and an exact live-set criterion connecting local intermediate elimination to end-to-end peak-memory reduction.

2 Related Work

Post-training and mixed-precision quantization. PTQ reduces model cost without retraining, using reconstruction or sensitivity proxies to preserve low-bit fidelity (Nagel et al., 2020, Li et al., 2021, Wei et al., 2022, Yuan et al., 2022). These criteria improve efficiency–quality trade-offs without measuring downstream decisions directly. Object-detection PTQ (DetPTQ), Task-Circuit Quantization, and task-aware sensitivity analysis (TASA) introduce task information into quantization objectives or sensitivity estimation (Niu et al., 2023, Xiao et al., 2025, Wang et al., 2026). WAMPTQ selects budget-matched legal W4/W8 maps using source-disjoint native-task feedback. The optional Agent narrows the unscored candidate pool while keeping the native-task selector fixed.

Low-bit execution and memory-efficient inference. Efficient deployment also depends on runtime materialization. FlashAttention reduces attention intermediates, and PagedAttention separates logical context from physical storage (Dao et al., 2022, Kwon et al., 2023). Compute Unified Device

Architecture (CUDA) templates for linear algebra subroutines and solvers (CUTLASS), mixed-precision auto-regressive parallel inference on large language models (MARLIN), the flexible lookup table engine (FLUTE), compiler systems, and singular value decomposition-based quantization (SVDQuant) specialize low-bit kernels or fuse execution (NVIDIA, 2026, Frantar et al., 2025, Guo et al., 2024, PyTorch Contributors, 2026, Li et al., 2025, Chen et al., 2018). Workspace, extended lifetimes, or secondary peaks can offset local savings. With the artifact fixed, QEF/VCF remove intermediates to improve complete-workload time and memory.

3 METHOD

For a pretrained WAM and workload w, WAMPTQ selects a budget-matched map by native development utility, optionally after Agent contraction, then freezes it for QEF/VCF execution (Figure 1). Sections 3.1–3.3 present task selection and its within-pool guarantee, optional contraction and its coverage loss, and fixed-artifact execution governed by the global live-set criterion.

3.1 TASK-UTILITY-GUIDED PRECISION SELECTION

Calibration proxies need not preserve native task utility. We select maps using native development feedback at a matched packed size, separating selector effects from model size.

Legal maps and source splits. Let $\mathcal{U}_w = \{u_1, \ldots, u_{N_w}\}$ be the N_w eligible quantization units of workload w. A precision map $c = (b_1, \ldots, b_{N_w})$ assigns $b_i \in \{4, 8\}$, while protected units retain their prescribed precision. The artifact $A_w(c)$ stores weight codes, scales, zero points, the map, and metadata. Its serialized size $S_w(c)$ must match the target budget $B_w > 0$ within a relative tolerance:

$$|S_w(c) - B_w|/B_w \le \varepsilon_B, \qquad \varepsilon_B = 10^{-3}. \tag{1}$$

Before development scoring, the legal generator rejects constraint violations and duplicates to produce an ordered pool $\mathcal{C}_w$ of $K = 12$ maps. Quantizer, activation precision, grouping, protected operations, and index tie-breaking are fixed across selectors. Disjoint source sets $\mathcal{D}_{\text{cal}}$, $\mathcal{D}_{\text{dev}}$, and $\mathcal{D}_{\text{eval}}$ serve quantization/proxy scoring, task feedback, and frozen final evaluation, respectively.

Native-task selection and proxy controls. Task selection maximizes native development utility:

$$c_T = \arg\max_{c \in \mathcal{C}_w} \widehat{Q}_w^{\text{dev}}(c). \tag{2}$$

Planning uses source-averaged success, Franka Ranking uses episode-clustered pairwise accuracy, and RoboCasa Windows uses demonstration-clustered recorded-window quality. This feedback captures effects absent from calibration proxies without model training.

Reconstruction minimizes normalized distortion at the adapter's quantization target boundary:

$$E_R(c) = \frac{1}{|\mathcal{D}_{\text{cal}}|} \sum_{x \in \mathcal{D}_{\text{cal}}} \frac{\|h_w(x) - \widehat{h}_{w,c}(x)\|_2^2}{\|h_w(x)\|_2^2 + \varepsilon_{\text{rec}}}, \quad c_R = \arg\min_{c \in \mathcal{C}_w} E_R(c). \tag{3}$$

Outputs $h_w(x)$ and $\widehat{h}_{w,c}(x)$ have matching shapes and are flattened before comparison. The positive constant ε_{rec} stabilizes the ratio. For comparison, the trace-weighted Hessian proxy is

$$E_H(c) = \sum_{i=1}^{N_w} \text{Tr}(H_i)\|W_i - \widehat{W}_i(c)\|_F^2, \quad c_H = \arg\min_{c \in \mathcal{C}_w} E_H(c), \tag{4}$$

where W_i and $\widehat{W}_i(c)$ are original and reconstructed unit weights, and $H_i \in \mathbb{R}^{d_i \times d_i}$ is the calibration Hessian approximation for $d_i = \dim \text{vec}(W_i)$. Protected-unit terms are constant and omitted.

3.1.1 WITHIN-POOL SELECTION ERROR

We bound within-pool selection error and analyze contraction coverage loss separately (Section 3.2).

Assumption 1 (Pre-development construction and source sampling)**.** Let $\mathscr{F}_w$ contain all pre-development information and randomness, including calibration, Agent proposals, and candidate construction. Conditional on $\mathscr{F}_w$, the ordered candidate pools and artifacts are fixed, and development source clusters $Z_1, \ldots, Z_n$ are independent and identically distributed from $\mathcal{P}_w$, with $n \ge 1$. Each source utility $v(c, Z)$ is measurable and lies in $[0, 1]$. Repeated executions or windows from a source belong to that cluster. The task selector maximizes its empirical mean with fixed-index tie-breaking.

Condition on $\mathscr{F}_w$ and define $q(c) = \mathbb{E}[v(c, Z)]$ and $\widehat{q}_n(c) = n^{-1}\sum_{j=1}^{n} v(c, Z_j)$. Equal-source selection uses $\widehat{Q}_w^{\mathrm{dev}} = 100\widehat{q}_n$. Candidate scores may be dependent through shared sources.

Theorem 1 (Finite-pool task-selection guarantee)**.** *Under Assumption 1, let $c^\star \in \arg\max_{c\in\mathcal{C}_w} q(c)$ for a pool of size $K \geq 1$, and let c_T maximize $\widehat{q}_n$. For $0 < \delta < 1$, with probability at least $1-\delta$,*

$$q(c^\star) - q(c_T) \leq \min\{1, 2\eta_K\}, \qquad \eta_K = \sqrt{\frac{\log(2K/\delta)}{2n}}. \tag{5}$$

Proof sketch. Bounded-source concentration and a union bound give $|\widehat{q}_n(c) - q(c)| \leq \eta_K$ for all candidates. Comparing $c^\star$ and c_T gives two deviations and a nonpositive empirical difference by optimality. Regret is at most one. Appendix A.2 proves the concentration step and extends the result to independent sources with fixed, possibly unequal weights. □

The bound concerns the best map in the realized pool. It requires no assumption that reconstruction or Hessian scores preserve the ordering of maps by task utility.

3.2 Optional Agent-assisted contraction and candidate coverage

To reduce scoring cost, the optional Agent narrows legal proposals while retaining native-task selection (Section 3.1). Let Θ_w be the preset legal proposal region, and let I_w collect architecture metadata, calibration statistics, B_w, quantization constraints, and hardware constraints. With fixed model and prompt configuration ϕ and proposal randomness ξ, the Agent returns a restricted region:

$$\widetilde{\Theta}_w = \mathcal{A}_\phi(I_w, \xi), \qquad \mathcal{C}_w^A = \Gamma_w(\widetilde{\Theta}_w, I_w), \qquad |\mathcal{C}_w^A| = 12. \tag{6}$$

The deterministic generator Γ_w uses a fixed seed and enforces Equation 1 with unchanged activation, group-size, protected-unit, and hardware constraints. The Agent narrows assignments without changing the quantization format or selector. Its inputs exclude development/final sources and candidate scores. The pool and ordering are frozen before development evaluation. Below, we analyze candidate-coverage loss separately from within-pool selection error. Appendix D gives the model, prompt, contraction trace, pool audit, repeats, sensitivity baseline, and cost.

3.2.1 Candidate coverage under contraction

Contraction reduces native evaluations but may exclude high-utility maps. We separate this coverage loss from selection error relative to a broader pre-development pool.

Corollary 1 (Contraction separates coverage from selection)**.** *Under Assumption 1, for nonempty pre-development pools $\mathcal{C}_w^B$ and $\mathcal{C}_w^A$, define $q_B^\star = \max_{c\in\mathcal{C}_w^B} q(c)$, $q_A^\star = \max_{c\in\mathcal{C}_w^A} q(c)$, and $\rho_A = [q_B^\star - q_A^\star]_+$, where $[x]_+ = \max(x, 0)$. If c_T^A is selected from the K-map Agent pool, then with probability at least $1-\delta$,*

$$q_B^\star - q(c_T^A) \leq \min\{1, \rho_A + 2\eta_K\}. \tag{7}$$

Proof sketch. Add and subtract $q_A^\star$, bound $q_B^\star - q_A^\star$ by ρ_A, and use Theorem 1 on the Agent pool. □

Smaller K reduces the selection penalty but does not bound ρ_A, so contraction is compared with broad and random pools. These comparisons assess frozen search routes rather than directly measuring ρ_A (Appendix A.2). Agent randomness must be fixed before development scoring. Source independence is the statistical condition. Disjoint source identifiers prevent overlap but do not by themselves establish independent sampling. Appendix A.2 also gives a gap-dependent recovery bound.

3.3 Fixed-artifact execution and live-set control

Smaller packed artifacts need not lower peak memory: rescaling and context assembly can still create large transient tensors. We remove these avoidable intermediates after precision selection. Each execution intervention fixes $A_w(c_T)$, inputs, normalization, action generation, stopping policy, and output interpretation. Controls disable only the tested module, without precision search (Figure 2).

Deployment objective. WAMPTQ freezes $A_w(c_T)$ and changes only its compatible materialization route. Let F denote FP32, P a packed module-off control, and Q the final QEF+VCF-enabled route. For measured time $T_r > 0$ and peak allocated bytes $M_r > 0$ of route r,

$$S_{\mathrm{total}} = T_{\mathsf{F}}/T_{\mathsf{Q}}, \qquad R_{\mathrm{total}} = 1 - M_{\mathsf{Q}}/M_{\mathsf{F}}. \tag{8}$$

Ratios use the same full warmed workload, excluding offline map construction and scoring.

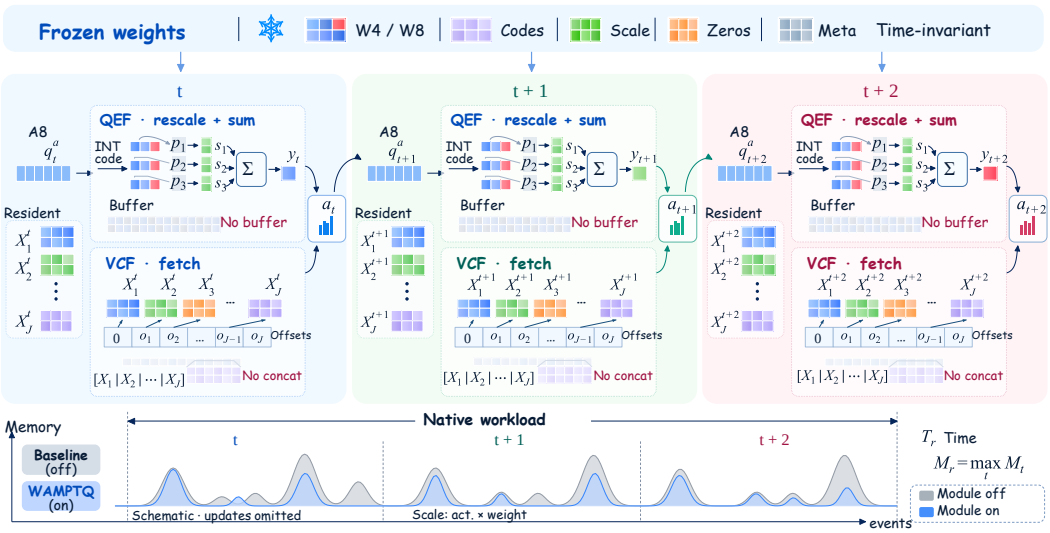

Figure 2: **Fixed-artifact execution over a native workload.** The precision map and packed weight metadata stay fixed as activations and resident context evolve. QEF avoids a full-sized rescaling intermediate, and VCF avoids contiguous context assembly. Schematic curves compare module-off/on execution of the same artifact. Labels use act. for activations and concat for concatenation. T_r measures elapsed time and M_r measures peak allocation over the complete warmed workload.

QEF. For a quantized linear operator with m rows, n_{out} output channels, and reduction width d_{in}, let $\mathcal{G}$ partition the reduction indices. For row i, channel j, and $k \in g$, centered integer activation and weight codes are $\bar{q}^a_{ik} = q^a_{ik} - z^a_{ig}$ and $\bar{q}^w_{jk} = q^w_{jk} - z^w_{jg}$. With positive groupwise scales and bias β_j,

$$y_{ij} = \sum_{g \in \mathcal{G}} s^a_{ig} s^w_{jg} \underbrace{\sum_{k \in g} \bar{q}^a_{ik} \bar{q}^w_{jk}}_{p_{ijg}} + \beta_j. \tag{9}$$

QEF retains p_{ijg} inside the producer, applies group-scale products, accumulates scaled contributions, and writes the required dtype without a full-sized rescaling/conversion intermediate. Proposition 2 proves real-arithmetic equivalence. Appendix A.3 gives overflow and floating-point bounds. Groups with different scales must be rescaled separately before their contributions are combined. This equivalence concerns the same quantized operands. Floating-point execution is checked against the corresponding unfused quantized route using the reported error tolerances.

VCF. Resident tensors $X_1, \ldots, X_J$ share non-context dimensions and have positive context lengths ℓ_j. Define $o_0 = 0$ and $o_j = \sum_{r=1}^{j} \ell_r$. Instead of allocating $X = \text{concat}_{d_{\text{ctx}}}(X_1, \ldots, X_J)$, VCF maps logical context index p to segment j with $o_{j-1} \le p < o_j$ and reads local index $p - o_{j-1}$. Positions, masks, quantization metadata, and logical order remain unchanged, and segments remain valid until the final read. Proposition 4 establishes equivalence for compatible consumers. Any extension of segment lifetimes is included as added allocation in the live-set memory analysis.

3.3.1 When local savings lower the workload peak

Let $\mathcal{T}$ be a nonempty finite set of aligned logical allocation events covering all peak-relevant live sets in both routes. At event t, write $M_t \ge 0$ for baseline live bytes, $D_t \ge 0$ for removed bytes, and $H_t \ge 0$ for bytes added by workspace or extended lifetimes. Count shared storage once and account for every change, so $M'_t = M_t - D_t + H_t \ge 0$. Define the net saving $d_t = D_t - H_t$, global baseline peak $P = \max_t M_t$, and per-event headroom $g_t = P - M_t$.

Theorem 2 (Exact global-peak criterion)**.** *Under complete aligned accounting (Assumption 4), the transformed peak $P' = \max_t M'_t$ satisfies*

$$P - P' = \min_{t \in \mathcal{T}} (g_t + d_t). \tag{10}$$

Thus $P' < P$ exactly when $d_t > -g_t$ at every event. Every baseline peak must have positive net saving, and no added allocation may exhaust another event's headroom.

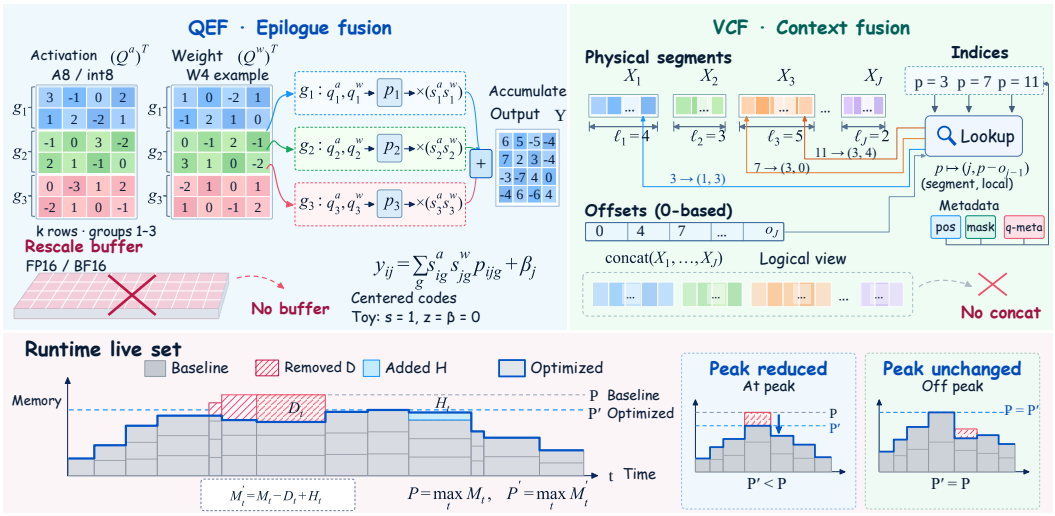

Figure 3: **Execution mechanisms and peak-memory accounting.** QEF rescales group partials, and VCF indexes resident segments. Formats are 16-bit floating point (FP16), 16-bit brain floating point (BF16), and signed 8-bit integers (int8). Labels pos and q-meta mean positions and quantization metadata. Removing D_t bytes and adding H_t workspace/lifetime bytes gives $M'_t = M_t - D_t + H_t$ at each aligned allocation event in the module-off/on comparison (Equation 10).

Algorithm 1 WAMPTQ: optional contraction, task selection, fixed-artifact execution

Require: Model, workload w, source-disjoint $\mathcal{D}_{\text{cal}}, \mathcal{D}_{\text{dev}}, \mathcal{D}_{\text{eval}}$, budget B_w and constraints.
Ensure: Frozen artifact $A_w(c_T)$ and final execution route Q.

1: Compute calibration statistics and optionally contract proposals using only I_w.
2: Construct and order $K = 12$ unique legal maps and freeze the pool before scoring.
3: **for** each $c \in \mathcal{C}_w$ **do**
4: Build $A_w(c)$ with the shared quantizer and calibration statistics.
5: Evaluate native development quality $\widehat{Q}_w^{\text{dev}}(c)$.
6: **end for**
7: Select c_T by Equation 2, break ties by index, and freeze $A_w(c_T)$.
8: Enable compatible QEF/VCF boundaries while preserving the artifact and task interface.
9: Evaluate on $\mathcal{D}_{\text{eval}}$ and measure warmed-workload time and peak allocation.

Proof. Substitute $M'_t = P - (g_t + d_t)$ and maximize. Subtraction from P proves both claims. □

For one modified stage with baseline bytes A and saving $0 \le d \le A$, let C be the largest unchanged stage. Then $P' = \max(A - d, C)$ and, when $A \ge C$, the global saving is $\min(d, A - C)$. If $A < C$, the peak is unchanged. This saturation predicts the positive and null VCF cases. Figure 3 illustrates this accounting. Appendix A.5 covers tied peaks, combined transforms, and uncertain allocations.

The criterion concerns peak allocated memory, not latency. Savings from separate interventions need not add because each intervention may change which event sets the global peak. Algorithm 1 summarizes contraction, task selection, and fixed-artifact execution.

4 Experiments

4.1 Evaluation design

We test (i) end-to-end WAM efficiency and quality, (ii) budget-controlled selection and contraction, and (iii) fixed-artifact QEF/VCF liveness predictions and numerical agreement.

The end-to-end study covers 15 native-setting workloads from Fast-WAM, LeWorldModel (LeWM), the DINO world model (DINO-WM), and action-conditioned video joint-embedding predictive architecture 2 (V-JEPA2-AC). DINO denotes self-distillation with no labels. Thirteen workloads report planning/control success. Franka Ranking and RoboCasa Windows are recorded-data endpoints,

Table 1: **End-to-end deployment: FP32 → final WAMPTQ.** Time is for complete warmed workloads, in seconds (s). Peak is allocated device memory in gibibytes (GiB, 2^{30} bytes). Quality is $100\times$ successes/trials. †: recorded-data endpoints. W4 share counts eligible modules. Table 26 also reports the parameter- and byte-weighted composition of the selected artifacts.

Task	Quality (%)	Time (s)	Speedup	Peak (GiB)	Memory ↓	W4 share
Fast-WAM						
Spatial	98.0 → 96.4	143.450 → 33.398	4.30×	23.256 → 5.103	78.06%	100.00%
Object	99.6 → 99.4	149.432 → 38.642	3.87×	23.256 → 5.103	78.06%	100.00%
Goal	97.2 → 96.4	145.039 → 33.948	4.27×	23.256 → 5.117	78.00%	99.35%
Long	95.4 → 93.4	13.166 → 6.569	2.00×	23.151 → 6.608	71.46%	60.10%
LeWM						
Wall	78.0 → 94.0	23.126 → 10.705	2.16×	0.439 → 0.162	63.12%	66.67%
Push-T	91.3 → 90.0	10.694 → 3.317	3.22×	0.439 → 0.155	64.77%	85.19%
TwoRoom	87.0 → 86.0	23.202 → 10.738	2.16×	0.439 → 0.162	63.12%	66.67%
Cube	78.0 → 84.0	11.863 → 4.169	2.85×	0.440 → 0.153	65.26%	92.59%
Reacher	94.0 → 92.0	11.340 → 3.069	3.70×	0.439 → 0.133	69.74%	100.00%
DINO-WM						
Wall	94.0 → 98.0	10.349 → 5.030	2.06×	4.126 → 0.811	80.33%	100.00%
PointMaze	100.0 → 100.0	14.161 → 5.992	2.36×	18.254 → 0.857	95.30%	100.00%
Push-T	92.0 → 92.0	41.243 → 19.918	2.07×	18.254 → 0.863	95.27%	63.64%
MetaWorld Reach	90.0 → 88.0	138.758 → 58.910	2.36×	2.460 → 0.848	65.51%	100.00%
V-JEPA2-AC						
Ranking†	93.3 → 91.1	37.507 → 12.236	3.07×	5.370 → 1.415	73.66%	61.30%
Windows†	35.7 → 37.5	114.990 → 53.734	2.14×	6.022 → 2.049	65.97%	55.60%

with 45 comparisons from 15 episodes and 56 windows from 14 demonstrations. Their percentages remain separate from the six-planning aggregate. The controlled study uses Fast-WAM Long, LeWM Wall/Push-T/TwoRoom/Cube, DINO-WM Push-T, and both auxiliary endpoints. Task suffixes L/D denote LeWM/DINO-WM. Planning sources have three executions, averaged within and then across sources. Resampling preserves these clusters. Deployment quality uses final binary-success counts. All selectors share the original $K = 12$ pool. Contraction changes only pool construction. The selector, splits, constraints, and budget tolerance (10^{-3}) remain fixed.

Primary systems measurements use a driver-reported NVIDIA GeForce RTX 4090. Timing covers the complete warmed workload, excluding loading, packing, compilation, and initialization. We disable result-dependent early stopping and synchronize boundaries. Peak memory is allocated device memory over the same workload. Module timing uses 10 alternating control/intervention and intervention/control (AB/BA) pairs. Appendices B and F give details.

4.2 End-to-end deployment across 15 tasks

Across all 15 workloads, the complete route delivers 2.00–$4.30\times$ speedup and 63.1–95.3% peak-memory reduction relative to FP32 (Table 1). Ratios use recorded endpoints before display rounding. Workload medians are $2.36\times$ speedup and 71.46% memory reduction, without pooling times. These are equally weighted summaries across native workloads with different execution lengths. They should not be read as a single speedup or memory reduction for a pooled deployment workload.

Fast-WAM Spatial uses all-W4-eligible modules at $4.30\times$ speedup, while Long uses 60.10% W4 at $2.00\times$. W4 module share is not a monotonic runtime proxy: latency and peak allocation also include activations, context, and workspace. Module counts also differ from parameter and serialized-byte fractions: units vary in parameter count, and packed artifacts include scales, zero points, padding, and other metadata. DINO-WM PointMaze retains $50/50$ successes at $18.254 \to 0.857$ GiB peak allocation. Appendix E gives weighted composition and artifact sizes.

Quality changes span -2.22 to $+16.0$ percentage points: nine declines, two ties, and four gains. The extremes are Ranking ($42/45 \to 41/45$) and LeWM Wall ($39/50 \to 47/50$). With checkpoint, planner, states, and scoring fixed, five paired Wall runs average 79.2% for FP32 versus 93.6% for the quantized route ($+14.4$ points). The audit reproduces the improvement, although these matched runs do not establish the cause of the observed quality improvement (Appendices E and C.6.1).

Table 2: **Controlled precision selection and selection-aware robustness.** All selectors share 12 legal budget-matched maps. Differences are reported in percentage points (pp), with selection rerun over 10 calibration/development resamples. Confidence intervals (CIs) use 20,000 nested source-bootstrap replicates. Recon. denotes reconstruction. †: recorded-data endpoints. Frozen aggregates use displayed estimates, and resampled aggregates use procedure-level summaries.

	Frozen final quality (%)			Task−Hessian (pp)		
Workload	Recon.	Hessian	Task	Frozen	Resampled	95% nested CI
Long	92.4	92.8	**93.4**	+0.6	+0.5	[0.0,+1.0]
Wall-L	90.0	92.0	**94.0**	+2.0	+1.6	[-1.0,+4.2]
Push-L	89.3	89.3	**90.0**	+0.7	+0.6	[-0.4,+1.6]
TwoRoom	84.0	85.0	**86.0**	+1.0	+0.8	[-1.0,+2.6]
Cube	80.0	82.0	**84.0**	+2.0	+1.6	[-1.2,+4.5]
Push-D	90.0	90.0	**92.0**	+2.0	+1.6	[-1.1,+4.3]
Ranking†	88.9	88.9	**91.1**	+2.2	+1.9	[-2.4,+6.0]
Windows†	35.7	35.7	**37.5**	+1.8	+1.4	[-3.1,+5.8]
Six planning				**+1.38**	**+1.11**	**[+0.28,+1.93]**

Table 3: **Auxiliary candidate contraction.** Frozen final quality (%) under a shared task selector, quantization constraints, and budget tolerance. Δ_B and Δ_R are Agent minus Broad and Random, in percentage points. Planning means using unrounded execution counts.

Workload	Broad-48	Random-12	**Agent-12**	Δ_B	Δ_R
Long	93.53	92.47	93.40	-0.13	$+0.93$
Wall-L	94.00	91.33	94.00	0.00	$+2.67$
Push-L	90.22	88.89	90.00	-0.22	$+1.11$
TwoRoom	86.33	84.67	86.00	-0.33	$+1.33$
Cube	84.00	81.33	84.00	0.00	$+2.67$
Push-D	92.00	89.33	92.00	0.00	$+2.67$
Ranking†	91.11	88.89	91.11	0.00	$+2.22$
Windows†	37.50	35.71	37.50	0.00	$+1.79$
Six planning mean	90.01	88.00	89.90	-0.11	$+1.90$
Dev-scored maps	48	12	12		

4.3 Task feedback selects stronger precision maps

With quantization fixed across 12 legal budget-matched maps, task selection leads on all eight frozen final endpoints and gains $+1.38$/$+2.28$ six-planning points over Hessian/reconstruction (Table 2).

The aggregate pattern persists over ten source-level calibration/development resamples that rerun selection over the fixed pool. A further 20,000 nested bootstrap replicates also resample final sources. The planning Task−Hessian difference is $+1.11$ points with a 95% interval of $[+0.28, +1.93]$. Modal task maps recur in 70–90% of resamples. Per-workload intervals touch or span zero, so the aggregate provides the clearest evidence of an advantage for task selection (Appendix C).

Task utility and reconstruction can disagree: small distortions may cross a decision boundary while larger ones need not (Appendix A.2). In the controlled pool, the largest planning Task−Hessian gains occur on Wall-L, Cube, and Push-D ($+2.0$ points each). A separate matched-budget deployment-count comparison points in the same direction: on Cube, the task-selected route records $42/50$ successes versus GPTQ's $39/50$ and round-to-nearest (RTN) quantization's $38/50$ (Appendix C). These counts use a different estimator and are not part of the selector bootstrap.

4.4 Agent contraction retains near-Broad-search quality

Broad-48 evaluates 48 legal maps, Random-12 samples 12 from that pool, and Agent-12 scores 12 maps from its contracted region. Current-12 denotes the original deployment route. All routes use the same native-task selector and freeze the selected map before final evaluation.

Agent-12 reduces native development candidate evaluations by 75% ($384 \to 96$ across eight workloads) while keeping the six-planning mean close to Broad-48: 89.90% versus 90.01% (Table 3). The largest planning gap to Broad is 0.33 points. Relative to the equal-sized Random-12 subset, Agent-12

Table 4: **Fixed-artifact module effects.** Peak allocation (MiB) uses the final endpoint when on and endpoint alignment when off. Timing: 10-pair median relative change with a 95% bootstrap interval.

Workload	Module	Peak off→on	Memory ↓	ΔT	95% timing CI
Push-L	QEF	$182.17 \to 158.49$	13.0%	$+2.10\%$	$[+1.69, +2.50]\%$
Cube	QEF	$172.32 \to 156.64$	9.1%	$+2.50\%$	$[+1.98, +3.01]\%$
Long	QEF	$7074.59 \to 6766.84$	4.35%	$+1.95\%$	$[+1.50, +2.40]\%$
Wall-D	VCF	$830.91 \to 830.91$	0.0%	0.00%	$[-0.31, +0.40]\%$
PointMaze	VCF	$898.80 \to 878.04$	2.31%	-5.60%	$[-6.20, -5.05]\%$

Table 5: **Live-set accounting (MiB).** After equals Before minus Removed plus Added, up to rounding. The endpoint-aligned liveness summary gives allocations and module-off → module-on global peaks.

Workload / module	Before	Removed	Added	After	Global peak
Push-L / QEF	182.17	31.68	8.00	158.49	$182.17 \to 158.49$
Cube / QEF	172.32	25.68	10.00	156.64	$172.32 \to 156.64$
Long / QEF	7074.59	367.74	60.00	6766.84	$7074.59 \to 6766.84$
Wall-D / VCF	760.00	32.00	4.00	732.00	$830.91 \to 830.91$
PointMaze / VCF	898.80	64.00	8.00	842.80	$898.80 \to 878.04$

reaches 89.90% versus 88.00% and is higher on every recorded endpoint in this comparison. These frozen search estimates provide neither a final-data oracle nor a non-inferiority test.

Across 12 Agent and 20 Random-12 proposal/subset runs, the six-planning mean is $89.72 \pm 0.17\%$ versus $88.01 \pm 0.28\%$, a $+1.71$-point mean difference with a run-level percentile-bootstrap 95% interval of $[+1.56, +1.86]$. Agent-12 also outperforms a calibration-sensitivity 12-map contraction on all eight endpoints, with a $+0.98$-point six-planning difference. Appendix D reports repeats, overlap, model and prompt settings, and costs. Agent-assisted offline time totals 2099.5 s versus 6842.0 s for Broad-48, a 69.3% reduction. Deployment speedups exclude search.

4.5 Fixed-artifact execution and live-set mechanisms

For each module, P and Q differ only in whether it is enabled on the same artifact. Table 4 reports memory reduction and relative time change (Equation 41). Memory controls align the recorded endpoints. Timing intervals are calculated from paired control and intervention runs.

On three workloads, QEF further reduces memory by 4.35–13.0% at a 1.95–2.50% timing cost. All timing intervals exceed zero. On Push-L, 31.68 MiB removed minus 8.00 MiB added lowers the control peak by 23.68 MiB. The FP32/control/final sequence is $449.85 \to 182.17 \to 158.49$ MiB, a 64.77% total reduction. This incremental effect differs from full-route speedup.

VCF illustrates secondary peaks (Table 5). On Wall-D, removing 32 MiB and adding 4 MiB lowers the context stage from 760 to 732 MiB, below the unchanged 830.91 MiB global peak. On PointMaze, removing 64 MiB and adding 8 MiB lowers the stage from 898.80 to 842.80 MiB. An unchanged 878.04 MiB stage then sets the peak, so 56 MiB saved locally yields 20.76 MiB globally. Equation 10 predicts both cases. Push-L converts its full 23.68 MiB net stage saving into a global reduction.

On the stock 24 GB GPU, QEF reduces peak allocation by 12.9% on Push-L and 9.0% on Cube, while VCF gives 0.0% on Wall-D and 2.3% on PointMaze. All 448 fixed-artifact operator checks pass numerical, contract, and fallback criteria. Appendix F details tolerances, replication times, fixed-work replay, and applicability. QEF supports $3700/3786$ (97.7%) eligible packed producers, and VCF supports $10/11$ context-assembly boundaries observed across 7 of 15 tasks.

5 Conclusion

WAMPTQ separates WAM precision selection from runtime materialization control. Native-task feedback selects budget-matched W4/W8 maps, optional Agent contraction reduces evaluations while retaining near-broad-search quality, and QEF/VCF remove avoidable intermediates according to global live-set behavior. Across 15 tasks from four WAM families, it achieves 2.00–$4.30\times$ speedup and 63.1–95.3% peak-allocation reduction over FP32, with at most 2.22 percentage points of observed quality loss. Evaluation is limited to these native-task settings. Testing broader WAM families and deployment distributions is an important direction for future work.

ARTIFICIAL INTELLIGENCE USE STATEMENT

Generative artificial intelligence (AI) tools were used to polish the writing. The authors are responsible for validating the claims, proofs, experiments, and final presentation.

ETHICS STATEMENT

This work studies efficient deployment of world action models. Deployment should include application-specific safety testing and human oversight, while respecting the licenses and privacy requirements of the underlying models and data.

REPRODUCIBILITY STATEMENT

The code will be released. The appendices document the experimental settings, quantization conventions, evaluation protocols, Agent configuration and prompts, and complete proofs.

REFERENCES

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-free 4-bit inference in rotated LLMs. In *Advances in Neural Information Processing Systems*, 2024.

Mido Assran, Adrien Bardes, David Fan, et al. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, Carlos Guestrin, and Arvind Krishnamurthy. TVM: An automated end-to-end optimizing compiler for deep learning. In *USENIX Symposium on Operating Systems Design and Implementation*, pp. 578–594, 2018.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022.

Zhen Dong, Zhewei Yao, Daiyaan Arfeen, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. HAWQ-V2: Hessian aware trace-weighted quantization of neural networks. In *Advances in Neural Information Processing Systems*, 2020.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023.

Elias Frantar, Roberto L. Castro, Jiale Chen, Torsten Hoefler, and Dan Alistarh. MARLIN: Mixed-precision auto-regressive parallel inference on large language models. In *ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 2025.

Zhongqian Fu, Tianyi Zhao, Kai Han, Hang Zhou, Xinghao Chen, and Yunhe Wang. An empirical study of world model quantization. *arXiv preprint arXiv:2602.02110*, 2026.

Han Guo, William Brandon, Radostin Cholakov, Jonathan Ragan-Kelley, Eric P. Xing, and Yoon Kim. Fast matrix multiplications for lookup table-quantized LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 12419–12433, 2024.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *ACM Symposium on Operating Systems Principles*, 2023.

Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. BRECQ: Pushing the limit of post-training quantization by block reconstruction. In *International Conference on Learning Representations*, 2021.

Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. SVDQuant: Absorbing outliers by low-rank component for 4-bit diffusion models. In *International Conference on Learning Representations*, 2025.

Tianqi Li, Wenyu Fang, Xin He, Xue Geng, Xu Cheng, and Yun Liu. When W4A4 breaks camouflaged object detection: Token-group dual-constraint activation quantization. In *European Conference on Computer Vision*, pp. 77–95, 2026.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, 2024.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. SpinQuant: LLM quantization with learned rotations. In *International Conference on Learning Representations*, 2025.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorldModel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

Markus Nagel, Rana Ali Amjad, Mart van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? Adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pp. 7197–7206, 2020.

Lin Niu, Jiawei Liu, Zhihang Yuan, Dawei Yang, Xinggang Wang, and Wenyu Liu. Improving post-training quantization on object detection with task loss-guided L_p metric. *arXiv preprint arXiv:2304.09785*, 2023.

NVIDIA. Efficient GEMM in CUDA. *CUTLASS Documentation*, 2026. Accessed September 25, 2026. `https://docs.nvidia.com/cutlass/latest/media/docs/cpp/efficient_gemm.html`.

PyTorch Contributors. torch.compiler. *PyTorch Compiler Documentation*, 2026. Accessed September 25, 2026. `https://docs.pytorch.org/docs/stable/user_guide/torch_compiler/torch.compiler.html`.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. OmniQuant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations*, 2024.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. Q-BERT: Hessian based ultra low precision quantization of BERT. In *AAAI Conference on Artificial Intelligence*, 2020.

Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. HAQ: Hardware-aware automated quantization with mixed precision. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

Fei Wang, Chao Xue, Taoran Liu, Li Shen, Ye Liu, and ChangXing Ding. Beyond activation alignment: The alignment-diversity tradeoff in task-aware LLM quantization. *arXiv preprint arXiv:2607.00908*, 2026.

Xiuying Wei, Ruihao Gong, Yuhang Li, Xianglong Liu, and Fengwei Yu. QDrop: Randomly dropping quantization for extremely low-bit post-training quantization. In *International Conference on Learning Representations*, 2022.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pp. 38087–38099, 2023.

Hanqi Xiao, Yi-Lin Sung, Elias Stengel-Eskin, and Mohit Bansal. Task-Circuit Quantization: Leveraging knowledge localization and interpretability for compression. In *Conference on Language Modeling*, 2025.

Zhihang Yuan, Chenhao Xue, Yiqi Chen, Qiang Wu, and Guangyu Sun. PTQ4ViT: Post-training quantization for vision transformers with twin uniform quantization. In *European Conference on Computer Vision*, pp. 191–207, 2022.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning*, PMLR 267:79115–79135, 2025.

Jiacheng Zhou, Jinfan Lv, Ruixuan Li, Longtai Zhang, Yan Wang, Wenqiang Zhang, and Lizhe Qi. QuantWAMs: Calibrating at the right granularity for world action models. *arXiv preprint arXiv:2607.28405*, 2026.

SUPPLEMENTARY MATERIAL

A DEFINITIONS, ASSUMPTIONS, AND COMPLETE PROOFS

The analysis concerns selection from a finite legal pool and execution of a fixed quantized artifact. The concentration results control population task utility on a specified source distribution. The operator identities concern the same quantized operands. Live-set accounting concerns allocated device memory, not runtime or allocator-reserved capacity.

A.1 NOTATION AND PROBABILITY SPACE

Take a probability space $(\Omega, \mathscr{F}, \mathbb{P})$ carrying the calibration data, proposal randomness, and development clusters, and let $\mathscr{F}_w \subseteq \mathscr{F}$ be the sigma-field in Assumption 1. All pools are nonempty finite subsets of the finite legal W4/W8 map space and are measurable functions of $\mathscr{F}_w$. Candidate artifacts include fixed quantization statistics and any packing randomness. Conditional on $\mathscr{F}_w$, Z_j takes values in a measurable source space $\mathcal{Z}_w$ and $v(c, \cdot) : \mathcal{Z}_w \to [0, 1]$ is fixed. For compactness, $q(c)$ denotes $\mathbb{E}[v(c, Z) \mid \mathscr{F}_w]$. This quantity can vary with the realized calibration and artifact.

Source-level execution randomness is included in Z_j. Within-source utilities, including repeated executions and different candidates, can be dependent. Independence is required across clusters. Conditioning on a source-shared random variable must preserve the stated conditional independence. Disjoint source identifiers prevent overlap but do not prove independent sampling from $\mathcal{P}_w$.

The Hessian proxy in Equation 4 is the recorded trace-weighted criterion, not an assumption about task utility. Theorem 1 imposes no positivity or task-loss interpretation on H_i. Its estimator and the numerical stabilizer value belong to the implementation metadata summarized in Appendix G.

A.2 COMPLETE FINITE-POOL AND CONTRACTION PROOFS

All statements below condition on $\mathscr{F}_w$ unless stated otherwise. Assumption 1 fixes the pool before development scoring and requires independent bounded source observations. The Agent and legal generator may use calibration, architecture, and constraints, but cannot use development outcomes.

Lemma 1 (Bounded-source concentration)**.** *For a fixed candidate c and $\eta > 0$,*

$$\mathbb{P}\big(|\widehat{q}_n(c) - q(c)| \geq \eta \mid \mathscr{F}_w\big) \leq 2e^{-2n\eta^2}. \tag{11}$$

Proof. For a random variable $V \in [0, 1]$, define $f(\lambda) = \log \mathbb{E} \exp(\lambda(V - \mathbb{E}V))$ for $\lambda \in \mathbb{R}$. Boundedness permits differentiation under the expectation. Under the exponentially tilted distribution, $f''(\lambda)$ is a variance. For any $U \in [0, 1]$, $\mathrm{Var}(U) \leq \mathbb{E}U - (\mathbb{E}U)^2 \leq 1/4$, since $U^2 \leq U$. Hence $f''(\lambda) \leq 1/4$ and, because $f(0) = f'(0) = 0$, Taylor's integral formula yields $f(\lambda) \leq \lambda^2/8$ for either sign of λ. For $V_j = v(c, Z_j)$ and $\lambda > 0$, independence and Markov's inequality yield

$$\mathbb{P}(\widehat{q}_n(c) - q(c) \geq \eta) \leq e^{-\lambda n\eta} \prod_{j=1}^{n} \mathbb{E}e^{\lambda(V_j - q(c))} \tag{12}$$

$$\leq \exp(-\lambda n\eta + n\lambda^2/8). \tag{13}$$

The choice $\lambda = 4\eta$ gives $e^{-2n\eta^2}$. Apply the argument to $-\sum_j (V_j - q(c))$ and combine the two tails. □

Complete proof of Theorem 1. By Lemma 1 and a union bound over the K fixed candidates,

$$\mathbb{P}\left(\max_{c \in \mathcal{C}_w} |\widehat{q}_n(c) - q(c)| \geq \eta\right) \leq 2Ke^{-2n\eta^2}. \tag{14}$$

Set $\eta = \eta_K = \sqrt{\log(2K/\delta)/(2n)}$. On the complementary event,

$$q(c^\star) - q(c_T) = [q(c^\star) - \widehat{q}_n(c^\star)] + [\widehat{q}_n(c^\star) - \widehat{q}_n(c_T)] + [\widehat{q}_n(c_T) - q(c_T)] \tag{15}$$

$$\leq \eta_K + 0 + \eta_K. \tag{16}$$

The middle term is nonpositive by empirical optimality, including ties. All utilities lie in $[0, 1]$, so the regret is also at most one. This proves Equation 5. The proof does not require independence between the K candidate score vectors. Multiply the utility bound by 100 for its percentage-point form. □

Table 6: Notation. Operator dimensions and indices are local to their operator. Analytical memory quantities are in bytes. Tables explicitly specify display units.

Symbol	Definition
$w, \mathcal{U}_w, N_w$	Workload, eligible units, and eligible-unit count.
$c, b_i, \mathcal{C}_w, K$	Precision map, unit bit width, ordered legal pool, and pool cardinality.
$A_w(c), S_w(c), B_w$	Frozen packed artifact, its serialized bytes, and positive target budget.
$\varepsilon_B, \varepsilon_{\mathrm{rec}}$	Budget tolerance (10^{-3}) and positive reconstruction stabilizer.
$\Theta_w, \widetilde{\Theta}_w, \Gamma_w$	Preset proposal region, contracted region, and fixed legal-pool generator.
$I_w, \phi, \xi, \mathcal{A}_\phi$	Pre-development metadata, Agent configuration, proposal randomness, and Agent mapping.
$\mathscr{F}_w, \mathcal{P}_w, Z_j, n$	Pre-development information sigma-field, source distribution, source cluster, and independent development-cluster count.
$v(c, Z), q(c), \widehat{q}_n(c)$	Bounded source utility, conditional population utility, and empirical mean.
$c_R, c_H, c_T, c^\star$	Reconstruction, Hessian, empirical-task, and population-best selections.
δ, η_K	Failure probability in $(0, 1)$ and uniform concentration radius.
$\mathcal{C}_w^B, \mathcal{C}_w^A$	Broad-reference and Agent-generated legal pools.
$q_B^\star, q_A^\star, \rho_A$	Pool-optimal utilities and positive-part coverage loss relative to Broad.
$a_j, q_a, n_{\mathrm{eff}}$	Fixed source weights, weighted population target, and $1/\sum_j a_j^2$.
$W_i, \widehat{W}_i, H_i, d_i$	Original/reconstructed weights, calibration Hessian approximation, and weight-vector dimension.
$m, n_{\mathrm{out}}, d_{\mathrm{in}}$	Input rows, output channels, and reduction width.
$\mathcal{G}, g, p_{ijg}, \nu$	Group partition, a group, integer group partial, and number of groups.
$q^a, q^w, z^a, z^w, s^a, s^w, \beta$	Activation/weight codes, zero points, positive scales, and bias.
$J, \ell_j, o_j, d_{\mathrm{ctx}}$	Segment count, lengths, prefix offsets, and context axis.
$\mathcal{T}, M_t, D_t, H_t$	Aligned events, baseline live bytes, removed bytes, and added workspace/lifetime bytes.
d_t, g_t, P, P'	Net saving, baseline headroom, and baseline/transformed global peaks.
$\mathsf{F}, \mathsf{P}, \mathsf{Q}, T_r, M_r$	FP32, module-off control, final route, route time, and route peak allocation.

Random pools fixed before scoring. For almost every realization of $\mathscr{F}_w$, the conditional failure probability is at most δ. Taking its expectation gives an unconditional failure probability at most δ. This covers random Agent proposals without a union bound over all proposal histories. The guarantee still concerns the best candidate in the realized pool. Inspecting development scores to choose among pools would violate this conditioning argument, even if the chosen pool is then frozen.

Complete proof of Corollary 1. Conditional on $\mathscr{F}_w$, both reference pools are fixed. Add and subtract $q_A^\star$:

$$q_B^\star - q(c_T^A) = (q_B^\star - q_A^\star) + (q_A^\star - q(c_T^A)) \le \rho_A + (q_A^\star - q(c_T^A)). \tag{17}$$

Apply Theorem 1 to $\mathcal{C}_w^A$, then use $q_B^\star - q(c_T^A) \le 1$. This proves the bound with the same failure probability, without assuming nesting. When $\mathcal{C}_w^A \subseteq \mathcal{C}_w^B$, $\rho_A = q_B^\star - q_A^\star$. For separately generated legal pools, the signed difference may be negative, and the positive part preserves the bound. □

Broad-48 itself uses empirical task selection. If c_T^B is its selected map and $r_s = q_s^\star - q(c_T^s) \ge 0$ for $s \in \{A, B\}$, then

$$q(c_T^B) - q(c_T^A) = (q_B^\star - q_A^\star) + r_A - r_B. \tag{18}$$

A final-panel difference adds evaluation sampling variation to this identity. Consequently, the observed Agent–Broad gap is a comparison of frozen search routes, not a direct measurement of ρ_A. The reported summary also does not establish candidate-by-candidate nesting of Agent-12 inside Broad-48. Equation 7 remains valid for either construction.

Independent sources with unequal weights. For the same K-map pool $\mathcal{C}_w$, let $a_1, \ldots, a_n \ge 0$ be fixed conditional on $\mathscr{F}_w$ with $\sum_j a_j = 1$, and let $V_j(c) \in [0, 1]$ be independent across j. Allow different means $\mu_j(c) = \mathbb{E}[V_j(c)]$. Define the estimand and estimator together:

$$q_a(c) = \sum_j a_j \mu_j(c), \qquad \widehat{q}_a(c) = \sum_j a_j V_j(c), \qquad n_{\mathrm{eff}} = \frac{1}{\sum_j a_j^2}. \tag{19}$$

The same moment-generating-function argument gives

$$\mathbb{E}\exp\left(\lambda\sum_j a_j(V_j(c)-\mu_j(c))\right)\le\exp\left(\frac{\lambda^2}{8}\sum_j a_j^2\right). \tag{20}$$

Optimizing the exponential bound at $\lambda = 4\eta/\sum_j a_j^2$ yields

$$\mathbb{P}(|\widehat{q}_a(c)-q_a(c)|\ge\eta)\le 2\exp\left(-\frac{2\eta^2}{\sum_j a_j^2}\right). \tag{21}$$

For $c_a^\star \in \arg\max_c q_a(c)$ and $\widehat{c}_a \in \arg\max_c \widehat{q}_a(c)$, a union bound and the same decomposition give

$$q_a(c_a^\star)-q_a(\widehat{c}_a)\le\min\left\{1, 2\sqrt{\frac{\log(2K/\delta)}{2n_{\text{eff}}}}\right\} \tag{22}$$

with probability at least $1-\delta$. Under a common source law, all $\mu_j(c) = q(c)$ and $q_a(c) = q(c)$. Otherwise, the target is the stated fixed mixture, not an unspecified common mean. The result excludes outcome-dependent weights. Weighting cannot make within-demonstration windows independent.

Proposition 1 (Gap-dependent map recovery)**.** *Under Assumption 1, suppose $K \ge 2$ and the pool has a unique population optimum $c^\star$. Write $\Delta_c = q(c^\star) - q(c) > 0$ for $c \ne c^\star$. Then*

$$\mathbb{P}(c_T \ne c^\star \mid \mathscr{F}_w)\le\min\left\{1, \sum_{c\ne c^\star} e^{-n\Delta_c^2/2}\right\}. \tag{23}$$

Proof. For competitor c, independent source differences $U_j = v(c^\star, Z_j) - v(c, Z_j)$ have range $[-1, 1]$ and mean Δ_c. Rescaling the bounded-source proof to range two gives $\mathbb{P}(n^{-1}\sum_j U_j \le 0) \le e^{-n\Delta_c^2/2}$. If $c_T \ne c^\star$, some competitor empirically matches or exceeds $c^\star$. A union bound suffices to control this event across all competing maps. □

For $\Delta_{\min} = \min_{c\ne c^\star}\Delta_c$, it suffices that $n \ge 2\Delta_{\min}^{-2}\log((K-1)/\delta)$ for recovery probability at least $1-\delta$. For $K = 1$, recovery is automatic. Small gaps permit map switching even when regret is small. Stable task utility need not imply a repeatedly selected map. Observed modal frequencies and Agent standard deviations summarize empirical results, not unknown population gaps.

Reconstruction ordering need not equal task ordering. Consider a one-dimensional boundary output $h = a > 0$ and task success exactly when the supplied output is positive. Two candidate outputs $\widehat{h}_1 = -a/2$ and $\widehat{h}_2 = 3a$ have squared reconstruction errors $9a^2/4$ and $4a^2$. Their normalization denominator is the same: $a^2 + \varepsilon_{\text{rec}}$. Reconstruction prefers candidate 1, while task utility prefers candidate 2. This analytical counterexample disproves a general ordering implication. It is not a simulated WAM experiment. Table 2 provides the empirical proxy comparison.

A.3 QEF: GROUPWISE IDENTITY, OVERFLOW, AND ROUNDING

Assumption 2 (Quantized operator contract)**.** The dimensions $m, n_{\text{out}}, d_{\text{in}}$ are positive integers, with $1 \le i \le m$ and $1 \le j \le n_{\text{out}}$. A finite collection $\mathcal{G}$ of nonempty groups partitions $\{1, \dots, d_{\text{in}}\}$. Integer codes and zero points and positive scales define $\widehat{a}_{ik} = s_{ig}^a(q_{ik}^a - z_{ig}^a)$ and $\widehat{w}_{jk} = s_{jg}^w(q_{jk}^w - z_{jg}^w)$ for $k \in g$. The reference and QEF use the same codes, scales, zero points, bias, grouping, and final output dtype. Integer group partials do not overflow. The producer can retain partial accumulators locally rather than materializing an output-sized conversion buffer.

Proposition 2 (QEF preserves the quantized linear operator)**.** *Under Assumption 2, applying each group's scale product inside the producer computes the same real-arithmetic linear operator as multiplying the reconstructed quantized operands. The computation admits a schedule without a globally materialized output-sized rescaling/conversion tensor.*

Proof. For fixed i, j,

$$y_{ij} = \sum_{k=1}^{d_{\text{in}}} \widehat{a}_{ik}\widehat{w}_{jk} + \beta_j \tag{24}$$

$$= \sum_{g\in\mathcal{G}} \sum_{k\in g} s^a_{ig}(q^a_{ik} - z^a_{ig}) s^w_{jg}(q^w_{jk} - z^w_{jg}) + \beta_j \tag{25}$$

$$= \sum_{g\in\mathcal{G}} s^a_{ig} s^w_{jg} \sum_{k\in g} \bar{q}^a_{ik}\bar{q}^w_{jk} + \beta_j. \tag{26}$$

Each scale is constant inside its group, giving Equation 9. A constructive schedule computes a group partial p_{ijg}, scales it, adds it to the local output accumulator, and reuses the partial storage for the next group. It then adds β_j and writes the output dtype. Repeating this schedule by output element or supported output tile needs no separate output-sized conversion tensor. Register/shared-memory usage, and any other workspace remain subject to the live-set accounting. □

Group boundaries and overflow. For arbitrary partials p_1, p_2, the identity $s_1p_1 + s_2p_2 = s(p_1 + p_2)$ holds for all partials only if $s_1 = s_2 = s$. Thus, a single rescaling after combining groups is generally invalid. For a signed integer accumulator of $b_{\text{acc}} \geq 2$ bits, define $A_g = \max_{i,k\in g} |\bar{q}^a_{ik}|$ and $W_g = \max_{j,k\in g} |\bar{q}^w_{jk}|$. A sufficient no-overflow condition, for products and all accumulation prefixes, is

$$|g| A_g W_g \leq 2^{b_{\text{acc}}-1} - 1. \tag{27}$$

It follows by bounding any prefix by the sum of absolute products in the entire group. If this condition fails, a wider accumulator or split accumulation must ensure a valid integer range.

Proposition 3 (A forward-error bound for scaled accumulation)**.** *Fix one output (i, j) and write $\nu = |\mathcal{G}|$ and $t_g = s^a_{ig}s^w_{jg}p_{ijg}$. Assume exact integer partials and exact conversion to the accumulation format. Use the same represented scale and bias constants on both routes. Form each t_g by two rounded multiplications, then sequentially sum the ν terms and bias. Each operation obeys the relative-error model with unit roundoff $0 < u_{\text{fp}} < 1$, without overflow or underflow. For $(\nu + 2)u_{\text{fp}} < 1$, define $\gamma_h = hu_{\text{fp}}/(1 - hu_{\text{fp}})$, $h = 0, \ldots, \nu + 2$. The pre-cast $\widetilde{y}_{ij}$ satisfies*

$$|\widetilde{y}_{ij} - y_{ij}| \leq \gamma_{\nu+2} \left(\sum_{g\in\mathcal{G}} |t_g| + |\beta_j| \right). \tag{28}$$

If the final dtype conversion obeys the same model with roundoff $0 < u_{\text{out}} < 1$, its result y^{out}_{ij} obeys

$$|y^{\text{out}}_{ij} - y_{ij}| \leq (1 + u_{\text{out}})|\widetilde{y}_{ij} - y_{ij}| + u_{\text{out}}|y_{ij}|. \tag{29}$$

Proof. A product of h rounding factors $(1 + \epsilon_r)$, $|\epsilon_r| \leq u_{\text{fp}}$, differs from one by at most γ_h when $hu_{\text{fp}} < 1$. For the upper deviation, expand $(1 + u_{\text{fp}})^h$ and use $\binom{h}{k} \leq h^k$ to bound it by $(1 - hu_{\text{fp}})^{-1}$. The lower deviation follows from $(1 - u_{\text{fp}})^h \geq 1 - hu_{\text{fp}}$. Each group term encounters two multiplications and at most ν additions. The bias encounters at most ν additions. Expanding the sequential computation expresses each exact term with a relative perturbation bounded by $\gamma_{\nu+2}$. The triangle inequality gives Equation 28. Finally, $y^{\text{out}}_{ij} = \widetilde{y}_{ij}(1 + \epsilon_{\text{out}})$ and $|\widetilde{y}_{ij}| \leq |y_{ij}| + |\widetilde{y}_{ij} - y_{ij}|$ give the stated bound on output-casting error relative to the exact quantized result. □

This bound states explicit sufficient arithmetic conditions, rather than asserting that every kernel uses that summation order. Non-exact integer conversion requires an additional conversion-error term. A fused or reassociated implementation requires its own operation count or the numerical checks below. Real-arithmetic equivalence concerns the quantized operands, not the original FP32 weights. Bitwise agreement requires identical arithmetic order and rounding. Appendix F.7 gives empirical tolerances for tested routes. The integer-product identity excludes protected non-integer operations.

A.4 VCF: LOGICAL INDEXING AND CONSUMER EQUIVALENCE

Assumption 3 (Segmented-consumer contract)**.** The $J \geq 1$ segments have positive integer lengths ℓ_j, compatible dtypes, and matching non-context dimensions. Offsets are $o_0 = 0$, $o_j = \sum_{r=1}^{j} \ell_r$. Segment lengths, offsets, and contents remain valid and immutable until the final read. The consumer reads tensor values and logical metadata, including positions, masks, and quantization scales. It uses neither contiguous-storage identity nor cross-segment pointer arithmetic and performs no mutation.

Proposition 4 (VCF preserves a compatible consumer)**.** *Under Assumption 3, virtual and materialized concatenation return the same values and metadata at every logical index. A deterministic consumer using that interface computes the same mathematical output.*

Proof. Positive lengths imply $0 = o_0 < o_1 < \cdots < o_J$. Every valid logical context index $p \in \{0, \ldots, o_J - 1\}$ belongs to exactly one interval $[o_{j-1}, o_j)$. For fixed non-context coordinates, concatenation satisfies

$$X[\ldots, p, \ldots] = X_j[\ldots, p - o_{j-1}, \ldots]. \tag{30}$$

VCF returns the right-hand side and the same logical metadata. Starting from an equal consumer state, induction over its sequence of interface reads, and deterministic state transitions give equal mathematical output. The interface excludes storage-address observations or mutations that would invalidate this induction. Floating-point reassociation needs a separate numerical check. □

Global normalization for attention. Let a query $q_{\text{att}} \in \mathbb{R}^{d_k}$ and logical keys $k_p \in \mathbb{R}^{d_k}$ have positive integer key dimension d_k, and let values $v_p \in \mathbb{R}^{d_v}$ have positive integer dimension d_v. Let $\mu_p \in \mathbb{R} \cup \{-\infty\}$ encode position and mask contributions and set $a_p = q_{\text{att}}^{\top} k_p / \sqrt{d_k} + \mu_p$. Assume all unmasked logits are finite and at least one index is unmasked. The mathematical output is

$$\frac{\sum_p e^{a_p - a_{\max}} v_p}{\sum_p e^{a_p - a_{\max}}}, \qquad a_{\max} = \max_p a_p. \tag{31}$$

Each sum covers the full logical context. Concatenating per-segment normalizations would change the operator. For each segment r that is not fully masked, define $m_r = \max_{p \in r} a_p$, $l_r = \sum_{p \in r} e^{a_p - m_r}$, and $u_r = \sum_{p \in r} e^{a_p - m_r} v_p$. Two nonempty aggregates merge as

$$m = \max(m_1, m_2), \tag{32}$$

$$l = e^{m_1 - m} l_1 + e^{m_2 - m} l_2, \tag{33}$$

$$u = e^{m_1 - m} u_1 + e^{m_2 - m} u_2. \tag{34}$$

Substitution expresses l, u as the full union's exponential denominator and numerator relative to m. Induction over segments yields the global expression u/l, with $l > 0$. Fully masked segments contribute zero and can be skipped. This is a sufficient compatible-consumer construction, not a claim that each evaluated VCF kernel uses attention or this merge.

Storage lifetimes. Keeping the original segments alive may extend their lifetimes beyond those needed for a conventional contiguous assembly. Such extensions, address-mapping metadata, and added workspace enter H_t in the live-set model. Aliases count once by underlying storage. Logical equivalence establishes correctness. Global memory reduction requires the separate peak criterion.

A.5 COMPLETE LIVE-SET PROOF AND EXTENSIONS

Assumption 4 (Complete aligned allocation accounting)**.** A nonempty finite event set $\mathcal{T}$ includes representatives of every constant-allocation interval relevant to either route. Logical events align the schedules. Equal wall-clock timestamps are not required. Baseline and transformed bytes M_t, M'_t count live device allocations under identical instrumentation, including persistent storage and workspace. All differences, including changed segment lifetimes and schedule-dependent allocations, enter removed storage $D_t \geq 0$ or added storage $H_t \geq 0$, so $M'_t = M_t - D_t + H_t \geq 0$. Each underlying allocation is counted once, including when several tensors alias the same storage.

Complete proof of Theorem 2. For $P = \max_t M_t$, $g_t = P - M_t$, and $d_t = D_t - H_t$,

$$P' = \max_t (M_t - D_t + H_t) \tag{35}$$

$$= \max_t \{P - (g_t + d_t)\} \tag{36}$$

$$= P - \min_t (g_t + d_t). \tag{37}$$

Both extrema are attained because $\mathcal{T}$ is finite and nonempty. Hence $P - P' = \min_t(g_t + d_t)$, which is positive exactly when $d_t > -g_t$ for every event. At a baseline maximum, $g_t = 0$ and d_t must be positive. At a non-peak event, a negative net saving is permissible only if its magnitude is smaller than the available headroom. A transformed-only peak omitted from $\mathcal{T}$ would invalidate the premise. □

Corollary 2 (Tied peaks and a target reduction)**.** *If $d_t \geq 0$ for all events, the global peak strictly decreases exactly when $d_t > 0$ at every baseline maximum. More generally, a desired reduction $r \geq 0$ is achieved exactly when $d_t \geq r - g_t$ for every t.*

Proof. For strict reduction, a non-peak event has $g_t > 0$, and nonnegative d_t makes its sum positive. At a peak, the sum is positive exactly when $d_t > 0$. The finite minimum is then positive. For the target condition, $P - P' \geq r$ is equivalent to $\min_t(g_t + d_t) \geq r$, or $g_t + d_t \geq r$ at every event. □

Corollary 3 (Single-stage saturation)**.** *Suppose one stage has baseline allocation $A \geq 0$ and saving $0 \leq d \leq A$. Let $C \geq 0$ be the maximum unchanged allocation, or $C = 0$ if none remain. Then*

$$P = \max(A, C), \quad P' = \max(A - d, C), \quad P - P' = \begin{cases} \min(d, A - C), & A \geq C, \\ 0, & A < C. \end{cases} \tag{38}$$

Proof. When $A < C$, C dominates both routes. When $A \geq C$ and $A - d \geq C$, the reduction is d. Otherwise, the new peak is C, and the reduction is $A - C$. The last two cases give the minimum. □

Wall-D has $A = 760.00$ MiB, $d = 28.00$ MiB, and $C = 830.91$ MiB, giving zero global reduction. PointMaze has $A = 898.80$ MiB, $d = 56.00$ MiB, and $C = 878.04$ MiB, giving about 20.76 MiB. These applications require the recorded aligned accounting, not only the nominal concatenation size.

Composition of execution changes. If two interventions have additive net savings $d_t^{(1)} + d_t^{(2)}$ at each aligned event, the combined peak reduction is

$$\min_t \big(g_t + d_t^{(1)} + d_t^{(2)}\big), \tag{39}$$

not generally the sum of the two individual peak reductions. For example, two tied 100-byte peaks can each be reduced by a different intervention by 20 bytes. Either intervention alone leaves a 100-byte maximum. Together they lower it to 80 bytes. This analytical example explains why local or module-level percentages do not add. If the transformations interact, the joint lifetime/workspace change must first be measured or accounted for rather than assumed additive.

Proposition 5 (Robustness to bounded accounting error)**.** *Let approximate event allocations $\widetilde{M}_t, \widetilde{M}'_t$ on the same complete event set obey $|\widetilde{M}_t - M_t| \leq e_0$ and $|\widetilde{M}'_t - M'_t| \leq e_1$ for all t, where $e_0, e_1 \geq 0$ are fixed byte bounds. Writing $\widetilde{P} = \max_t \widetilde{M}_t$ and $\widetilde{P}' = \max_t \widetilde{M}'_t$,*

$$|(\widetilde{P} - \widetilde{P}') - (P - P')| \leq e_0 + e_1. \tag{40}$$

Proof. For finite vectors x, y, $|\max_t x_t - \max_t y_t| \leq \max_t |x_t - y_t|$. The bounds for each route and the triangle inequality complete the proof. □

An approximate reduction greater than $e_0 + e_1$ therefore certifies a positive exact reduction under these error bounds. Rounded display values have bounded arithmetic resolution. Missing allocation events are a different problem and are not covered by such a bound.

Table 7: Primary mixed-precision workloads and evaluation units.

Identifier	Model family / task	N	Evaluation unit
Long	Fast-WAM / Long	500	Planning source
Wall-L	LeWM / Wall	50	Planning source
Push-L	LeWM / Push-T	150	Planning source
TwoRoom	LeWM / TwoRoom	100	Planning source
Cube	LeWM / Cube	50	Planning source
Push-D	DINO-WM / Push-T	50	Planning source
Ranking	V-JEPA2-AC / Franka	45	45 comparisons from 15 episodes
Windows	V-JEPA2-AC / RoboCasa	56	56 windows from 14 demonstrations

Table 8: Source-provenance audit for the controlled selector experiment. All listed overlaps are zero. Demos refers to the demonstration-level sources used for the Windows task.

Workload	Calibration	Development	Final	Calib.∩Dev	Calib.∩Final	Dev∩Final
Long	96	64	500	0	0	0
Wall-L	48	32	50	0	0	0
Push-L	72	48	150	0	0	0
TwoRoom	48	32	100	0	0	0
Cube	48	32	50	0	0	0
Push-D	48	32	50	0	0	0
Ranking	20 episodes	10 episodes	15 episodes	0	0	0
Windows	20 demos	10 demos	14 demos	0	0	0

A.6 System and module decomposition

For a frozen artifact, P disables only the tested module and Q enables it. The incremental memory reduction and relative timing change are

$$R_{\text{module}} = 1 - M_{\mathsf{Q}}/M_{\mathsf{P}}, \qquad \Delta T_{\text{module}} = (T_{\mathsf{Q}} - T_{\mathsf{P}})/T_{\mathsf{P}}. \tag{41}$$

For one aligned FP32/control/final triple, define

$$S_{\text{pre}} = T_{\mathsf{F}}/T_{\mathsf{P}}, \qquad S_{\text{module}} = T_{\mathsf{P}}/T_{\mathsf{Q}}, \qquad R_{\text{pre}} = 1 - M_{\mathsf{P}}/M_{\mathsf{F}}. \tag{42}$$

All times and peaks are positive. Cancellation of the common endpoint gives

$$S_{\text{total}} = S_{\text{pre}} S_{\text{module}}, \qquad 1 - R_{\text{total}} = (1 - R_{\text{pre}})(1 - R_{\text{module}}). \tag{43}$$

Also $S_{\text{module}} = 1/(1 + \Delta T_{\text{module}})$. Positive module timing cost yields a factor below one even if the total route is faster than FP32. The pre-module term contains all differences between FP32 and that packed control. It is not an isolated weight-only effect. Separately aggregated medians, devices, or replay workloads need not share an intermediate endpoint, so this identity cannot combine them.

B Evaluation units, source provenance, and repeated-run estimator

The six planning panels report success on fixed final-source panels. Auxiliary recorded-data prediction uses episodes for Ranking and demonstrations for Windows.

For selector s and final planning source j, average binary outcomes $y_{s,j,r}$ over $R = 3$ executions r:

$$\bar{y}_{s,j} = \frac{1}{3} \sum_{r=1}^{3} y_{s,j,r}. \tag{44}$$

The reported planning endpoint is

$$Q_s = \frac{100}{N} \sum_{j=1}^{N} \bar{y}_{s,j}. \tag{45}$$

Bootstrap resampling treats $\{y_{s,j,1}, y_{s,j,2}, y_{s,j,3}\}$ as one source cluster. Ranking resamples source episodes, and Windows resamples source demonstrations. The final deployment record instead stores a binary-success count and its stated trial denominator. Table 25 preserves those counts without interpreting them as pooled outcomes from the three-repeat selector experiment.

Table 9: Recorded native-task execution settings, not machine-level reproduction logs.

Workload family	Confirmed execution configuration
Fast-WAM LIBERO benchmark	50 evaluations per task identifier, one environment, 30-step initial wait, replanning every 10 steps, `action_horizon=null`, 10 inference/denoising steps, two 224×224 camera inputs. The runner caps Spatial/Object/Goal at 400 steps and LIBERO-10/90 suites at 700.
LeWM	`num_eval=50`, `world.num_envs=50`, 224×224 images, seed 42. Planning, receding, and action-block horizons are all 5. The cross-entropy method (CEM) uses 300 samples, variance scale 1, 30 optimization steps, and top-$k = 30$.
DINO-WM Wall/ PointMaze/ Push-T	Seed 99, $N = 50$, 224×224 inputs. Goal horizon 5 and CEM horizon 5 with 300 samples, variance scale 1, and top-$k = 30$. CEM optimization uses 10 steps for Wall/PointMaze and 30 for Push-T.
DINO-WM MetaWorld Reach	Seed 3, $N = 50$, input size 224. Encoder temporal context: 16 frames, prediction length: 4 frames, rollout: 1 step, and maximum episode length: 100 steps.
V-JEPA2-AC Ranking	Offline action-conditioned latent ranking over 15 valid episodes and three fractions per episode (45 samples). Horizon 3, raw-frame stride 8, and 19 candidates per sample. The recorded FP32 workload contains 45 encoder calls and 135 predictor calls.
V-JEPA2-AC Windows	56 windows from 14 demonstrations (four windows per demonstration). Stage decision budgets: 18/15/13/11 and action repeat: 5. The recorded FP32 control contains 3,882 environment steps, 836 encoder calls, and 2,340 predictor calls.

Table 10: Quantization conventions and serialized bytes per quantized parameter. Ranges give the minimum and maximum serialized bytes per quantized parameter across the 12 legal candidate maps.

Workload	Activation	Group	Pool range	Recon.	Hessian	Task
Long	A8	64	0.7203–0.7209	0.7207	0.7204	0.7206
Wall-L	A8	64	0.9073–0.9080	0.9078	0.9075	0.9079
Push-L	A8	64	0.7301–0.7308	0.7306	0.7302	0.7304
TwoRoom	A8	64	0.9072–0.9079	0.9076	0.9074	0.9077
Cube	A8	64	0.6423–0.6429	0.6428	0.6424	0.6427
Push-D	A8	32	0.7604–0.7609	0.7609	0.7605	0.7607
Ranking	A8	32	0.7173–0.7179	0.7178	0.7174	0.7177
Windows	A8/A16 protected	32	0.7664–0.7669	0.7669	0.7665	0.7668

B.1 Recorded workload execution configurations

Table 9 gives task settings from checked configuration files, runner code, or workload logs. Training-only fields do not determine deployment batch size or execution length. The Fast-WAM configuration uses the Lifelong Learning Benchmark on Robot Manipulation Tasks (LIBERO).

C Precision-selection details and robustness

C.1 Quantization conventions and packed-size audit

The controlled selector experiment uses W4/W8 mixed weights and A8 activations, with protected 16-bit activation (A16) operations for Windows. WxAy denotes x-bit weights and y-bit activations. Table 10 reports group size, the serialized size range across all 12 legal candidates, and the sizes of the three selected maps. These quantities define the comparison budget. The selected deployment artifacts use the quantizer conventions summarized in Table 11. Controlled-selector grouping remains the workload-specific setting shown in Table 10. Long per-unit precision vectors are retained as machine-readable experiment artifacts rather than typeset bit strings.

C.2 Frozen-map final-source uncertainty

Table 12 reports source-aware conditional intervals after the three maps are frozen. These intervals quantify final-source variation for realized selections. Selection-aware inference follows below.

Table 11: Selected-path quantizer audit.

Path	Scale / zero-point convention	Rounding, clipping, and calibration
Symmetric W4	Signed 4-bit, recorded W4A8 deployment path uses $G = 32$, group scale $s = \max \lvert w \rvert / 7$, no learned zero point.	`torch.round`, clamp to $[-8, 7]$, two 4-bit nibbles packed per byte.
Asymmetric W4	$G = 32$, range includes zero, $s = (w_{\max} - w_{\min})/15$ and $z = \mathrm{clip}(\mathrm{round}(-w_{\min}/s), 0, 15)$.	$q = \mathrm{clip}(\mathrm{round}(w/s + z), 0, 15)$, nibble packing.
W8 weights	Per-output-channel symmetric scale $s = \max \lvert w \rvert / 127$.	Round to nearest and clamp to signed int8 range $[-128, 127]$.
A8 activations	Dynamic per-row maximum-absolute-value scale $s = \max \lvert a \rvert / 127$, recomputed at each call.	Round to nearest and clamp to signed int8. Offline activation statistics guide allocation/protection but do not replace the per-call A8 scale.
A16 / training	A16 activations pass through without activation quantization.	Selected RTN paths use observed ranges and encoding-range clipping. No quantization-aware training (QAT) is used. The selected DINO-WM Wall GPTQ candidate instead uses a second-order Hessian, sequential quantization, and error feedback.

Table 12: Frozen-map selector differences (percentage points) with source-aware paired 95% intervals.

Workload	Task−Recon.	95% CI	Task−Hessian	95% CI
Long	+1.0	[+0.1,+1.9]	+0.6	[0.0,+1.3]
Wall-L	+4.0	[-1.4,+9.3]	+2.0	[-2.7,+6.7]
Push-L	+0.7	[-1.1,+2.5]	+0.7	[-1.0,+2.4]
TwoRoom	+2.0	[-1.3,+5.3]	+1.0	[-1.8,+3.8]
Cube	+4.0	[-2.0,+10.0]	+2.0	[-3.3,+7.3]
Push-D	+2.0	[-3.2,+7.2]	+2.0	[-3.1,+7.2]
Ranking	+2.2	[-4.4,+8.9]	+2.2	[-4.4,+8.9]
Windows	+1.8	[-5.4,+8.9]	+1.8	[-5.4,+8.9]
Six planning panels	+2.28	–	+1.38	–
All eight panels	+2.21	[+1.07,+3.35]	+1.54	[+0.65,+2.44]

The all-eight row is a descriptive unweighted mean over the eight prespecified endpoints. The six-panel row is the planning-specific summary used in the main text.

C.3 Selection-aware robustness

Each workload uses 10 independent source-level calibration/development resamples. The three selectors are rerun from the same fixed legal candidate pool for each resample, and final data remain excluded from selection. Procedure-level intervals use 20,000 nested bootstrap replicates over calibration, development, and final source units.

Table 13 gives workload effects and map stability. The main text gives aggregate planning inference.

Aggregate entries follow the procedure-level statistical summaries rather than an average of the rounded per-workload means. The underlying resample selections and nested-bootstrap draws are required to reproduce those aggregates. Final deployment counts alone cannot reconstruct them.

C.4 Random-map sanity check

The random-map pool measures legal-map variability separately from the controlled $K = 12$ pool.

C.5 Matched quantization baselines

Table 15 compares matched-budget RTN, GPTQ, and the task-selected route using deployment-count denominators. These are separate from the three-repeat selector estimates. On the six planning rows,

Table 13: Task-selector stability and selection-aware Task−Hessian effects. Modal frequency is the fraction of 10 source-level selection resamples that choose the most common task-selected map.

Workload	Task modal frequency	Selection-aware mean Δ	95% nested CI
Long	80%	+0.5	[0.0,+1.0]
Wall-L	70%	+1.6	[-1.0,+4.2]
Push-L	90%	+0.6	[-0.4,+1.6]
TwoRoom	80%	+0.8	[-1.0,+2.6]
Cube	70%	+1.6	[-1.2,+4.5]
Push-D	80%	+1.6	[-1.1,+4.3]
Ranking†	70%	+1.9	[-2.4,+6.0]
Windows†	70%	+1.4	[-3.1,+5.8]
Six planning panels	–	**+1.11**	**[+0.28,+1.93]**
All eight panels	–	**+1.26**	**[+0.38,+2.08]**

Table 14: Random-map sanity check (quality, %). SD denotes standard deviation. The retrospective final oracle is the highest final score among 20 random maps. It characterizes that separate pool, not a valid development-only selector. It is unrelated to Random-12 contraction.

Workload	Random mean±SD	Final oracle among 20	Recon.	Hessian	Task
Long	91.9 ± 0.5	92.8	92.4	92.8	93.4
Wall-L	89.4 ± 2.1	92.0	90.0	92.0	94.0
Push-L	88.5 ± 1.0	89.3	89.3	89.3	90.0
TwoRoom	83.2 ± 1.4	85.0	84.0	85.0	86.0
Cube	79.2 ± 2.0	82.0	80.0	82.0	84.0
Push-D	89.0 ± 1.6	90.0	90.0	90.0	92.0
Ranking	88.1 ± 1.7	88.9	88.9	88.9	91.1
Windows	34.4 ± 2.0	35.7	35.7	35.7	37.5

Table 15: Matched-budget final counts for RTN, GPTQ, and the task-selected map.

Workload	N	RTN	GPTQ	Task-selected
Long	500	458	459	467
Wall-L	50	44	45	47
Push-L	150	132	131	135
TwoRoom	100	82	84	86
Cube	50	38	39	42
Push-D	50	44	45	46
Ranking	45	40	40	41
Windows	56	19	20	21

differences computed from these exact counts average $+3.04$ points over GPTQ and $+4.30$ over RTN. Neither aggregate is substituted into the matched-pool selector comparison.

C.6 Broader native-setting quality and FP32 audit

Table 16 reports the broader native-setting quality comparison. FP32 and WAMPTQ follow the final deployment record. RTN, GPTQ, AWQ, and SmoothQuant-style entries retain their recorded results and precision conventions. This comparison provides deployment context. Table 2 isolates the selector criterion under a matched pool and budget.

C.6.1 Matched LeWM Wall audit

The LeWM Wall FP32/task-selected difference was rechecked under a paired protocol with the same checkpoint, planner, final initial states, stopping rule, preprocessing, and scoring script. Five paired executions give means of 79.2% for FP32 and 93.6% for the task-selected W4/W8 route, a paired mean difference of +14.4 percentage points. Table 17 reports the paired outcomes.

Table 16: Native-setting quality (%). SQ denotes the SmoothQuant-style W8A8 route.

Task	FP32	RTN	GPTQ	AWQ W4A16	SQ W8A8	WAMPTQ
Fast-WAM Spatial	98.0	94.8	95.6	96.8	97.4	96.4
Fast-WAM Object	99.6	98.8	99.0	99.4	99.6	99.4
Fast-WAM Goal	97.2	92.0	92.8	94.6	96.0	96.4
Fast-WAM Long	95.4	90.8	90.6	92.8	92.0	93.4
LeWM Wall	78.0	82.0	88.0	92.0	92.0	94.0
LeWM Push-T	91.3	86.7	85.3	89.3	88.7	90.0
LeWM TwoRoom	87.0	80.0	83.0	85.0	86.0	86.0
LeWM Cube	78.0	72.0	78.0	82.0	84.0	84.0
LeWM Reacher	94.0	86.0	90.0	92.0	94.0	92.0
DINO-WM Wall	94.0	98.0	96.0	98.0	96.0	98.0
DINO-WM PointMaze	100.0	98.0	98.0	99.3	99.3	100.0
DINO-WM Push-T	92.0	86.0	88.0	92.0	92.0	92.0
DINO-WM MetaWorld Reach	90.0	84.0	86.0	88.0	90.0	88.0
Franka Ranking	93.3	86.7	88.9	91.1	93.3	91.1
RoboCasa Windows	35.7	30.4	32.1	35.7	39.3	37.5

Table 17: LeWM Wall protocol audit on the same 50 final initial states. Labels identify paired runs.

Paired run	FP32 success	Task W4/W8 success	Difference
1	40/50	46/50	+12 pp
2	39/50	47/50	+16 pp
3	41/50	47/50	+12 pp
4	40/50	48/50	+16 pp
5	38/50	46/50	+16 pp
Mean	79.2%	93.6%	+14.4 pp

Table 18: Executed Agent configuration for the recorded contraction experiment.

Field	Recorded value
Provider/runtime	Self-hosted Hugging Face Transformers 4.51.0. External application programming interface (API) cost is $0.00.
Model snapshot	`Qwen/Qwen3-8B`, revision `47719a242beab8f9aecc40ce3928b034dd5dd559`.
Inference dtype	BF16. Thinking mode disabled.
Sampling	temperature 0.7, top-p 0.8, top-k 20, maximum 1024 new tokens.
Randomness	Run-specific proposal seed. The reference trace uses seed 4402.
Output contract	One strict JavaScript Object Notation (JSON) object specifying proposal version, W4-count target/window, required/preferred W8/W4 units, diversity constraints, candidate count, and rationale tags.
Source-access contract	Calibration-only evidence is allowed. Development/final sources and scores are excluded.

D Auxiliary Agent contraction: executed protocol, cost, and stability

D.1 Executed model, prompt, and data-access boundary

The Agent contracts a proposal region before any native development score is available. Equation 2 still selects the final map. Inputs contain architecture metadata, calibration-only statistics, the packed-size target and tolerance, quantization constraints, and hardware-format constraints. Development/final sources and their outcomes are excluded from the Agent input. The legal generator then enforces uniqueness, the 10^{-3} packed-size tolerance, the fixed activation/group/protected-operation rules for that experiment, and an index-based tie rule before task scoring.

Executed prompt. The system prompt and user template reproduce the experimental prompt with normalized punctuation. Workload-specific content replaces `WORKLOAD_JSON`. IDs identify units.

```
You are ProposalContractor, a pre-development
candidate-region contractor for W4/W8 post-training
quantization.
Your job is NOT to choose the final precision map.  The
downstream native-task selector will do that after legal
candidates are generated.
Allowed information:
- architecture metadata for eligible units.
- calibration-only statistics.
- packed-size target and tolerance.
- quantization, protected-operation, and hardware
constraints.
Forbidden information:
- development sources or native development scores.
- final/evaluation sources or final scores.
- any result derived from either development or final
outcomes.
Return one strict JSON object and no prose.  The object must
contain:
proposal_version, target_w4_count, w4_count_window, must_w8,
prefer_w8,
prefer_w4, min_pairwise_hamming, max_same_stage_flips,
candidate_count,
and rationale_tags.
Requirements:
1.  candidate_count must equal 12.
2.  target_w4_count must match the supplied budget target.
3.  must_w8/prefer_w8/prefer_w4 may only contain eligible
unit IDs.
4.  Never alter activation precision, group size, protected
operations, or hardware format.
5.  Use must_w8 only for a small set of consistently
high-sensitivity units.
6.  Keep enough free units to permit 12 distinct legal maps.
7.  min_pairwise_hamming is a diversity constraint, not a
task-quality score.
8.  If the requested region appears infeasible, set
rationale_tags to include "repair_required".  Do not relax
the packed-size tolerance yourself.
USER TEMPLATE:
Contract the legal proposal region for the following
workload.  Use calibration-only evidence and the fixed
deployment constraints.  Do not infer or request
development/final outcomes.
<I_w>
{{WORKLOAD_JSON}}
</I_w>
Return strict JSON only.
```

D.2 Representative contraction trace and candidate-pool audit

A recorded Wall-L trace illustrates the paper-level interface. Its input contains 108 eligible units, with per-unit role, stage, calibration sensitivity, reconstruction error, parameter count, and eligible bits. The target is 0.9079 serialized bytes per quantized parameter with relative tolerance 10^{-3}, W4/W8 weights, A8 activations, and group size 64. The Agent returns target W4 count 72 with window $[72, 72]$, `must_w8={u028}`, 18 `prefer_w8` units, 7 `prefer_w4` units, minimum pairwise Hamming distance 12, maximum same-stage flips 10, and 12 requested candidates. The legal generator returns twelve 72-W4/36-W8 maps corresponding to Broad-48 aliases B01, B02, B04, B05, B07, B08, B12, B15, B18, B22, B31, and B36. The task selector chooses B07 for Wall-L.

Table 19: Pool overlap and selected Broad-48 aliases. Here, "sel." denotes the selected map and "Sens." denotes sensitivity-based 12-map contraction.

Workload	Current∩Agent	Agent∩Sens.	Broad sel.	Current sel.	Agent sel.	Sens. sel.
Long	6	5	B17	B07	B12	B04
Wall-L	7	5	B07	B07	B07	B04
Push-L	6	5	B25	B07	B12	B10
TwoRoom	6	5	B34	B07	B12	B04
Cube	7	5	B07	B07	B07	B04
Push-D	6	4	B07	B07	B07	B04
Ranking	5	3	B07	B07	B07	B04
Windows	5	3	B07	B07	B07	B04

Table 20: Reference-run outcome counts for the contraction comparison. Planning denominators are final sources times three executions. Statistical clusters remain the original sources.

Workload	Outcome denominator	Broad-48	Random-12	Agent-12
Long	$500 \times 3 = 1500$	1403	1387	1401
Wall-L	$50 \times 3 = 150$	141	137	141
Push-L	$150 \times 3 = 450$	406	400	405
TwoRoom	$100 \times 3 = 300$	259	254	258
Cube	$50 \times 3 = 150$	126	122	126
Push-D	$50 \times 3 = 150$	138	134	138
Ranking†	45 comparisons	41	40	41
Windows†	56 windows	21	20	21

For auditability without typesetting hundreds of unit-level bits, Table 19 reports pool membership and selected-map aliases. All Current-12 and Agent-12 maps are members of the corresponding Broad-48 pool in this experiment. Full 4/8 vectors remain machine-readable experiment artifacts.

D.3 Reference-run count audit

Current-12 is the original fixed-pool selector route. Broad-48 scores 48 legal maps with the same native development selector, Random-12 scores a 12-map random subset of Broad-48, and Agent-12 scores the 12 maps admitted by the contracted region. Broad-48 is development-selected rather than a final-data oracle. The reference Agent-12 endpoint values match Current-12 numerically on all eight workloads, while selected-map identity can differ (Table 19).

The unrounded six-planning means in the reference run are 90.0148% for Broad-48, 88.0037% for Random-12, and 89.9000% for Agent-12. The maximum planning gap between Agent-12 and Broad-48 is 0.33 points on TwoRoom. Equal numbers of within-source repeats do not change the source-cluster unit used for statistical resampling.

D.4 Repeated Agent versus Random contraction

The repeated experiment uses 12 independent Agent proposal runs and 20 Random-12 subset runs. Table 21 reports the mean and sample SD across runs. The 95% intervals resample Agent and Random runs independently and take percentile intervals for the difference in run means.

Repetitions vary by proposal/subset construction, not final-source resampling. Discrete Ranking and Windows denominators give coarser run-to-run quality resolution than the six planning panels.

D.5 Sensitivity-based 12-map baseline

For an equal-size non-LLM comparison, Sensitivity-12 contracts the broad candidate set by calibration sensitivity, then applies the same native development selector. Agent-12 is higher on every recorded endpoint in Table 22. The six-planning mean difference is $+0.98$ points.

Table 21: Repeated Agent-12 versus Random-12: quality (%) and Agent-minus-Random Δ (points).

Workload	Agent n	Agent mean±SD	Random n	Random mean±SD	Δ [95% CI]
Long	12	93.36 ± 0.05	20	92.48 ± 0.32	$+0.87$ [$+0.73$, $+1.01$]
Wall-L	12	93.72 ± 0.34	20	91.03 ± 0.85	$+2.69$ [$+2.28$, $+3.10$]
Push-L	12	89.93 ± 0.11	20	88.88 ± 0.53	$+1.05$ [$+0.81$, $+1.28$]
TwoRoom	12	85.89 ± 0.16	20	84.75 ± 0.76	$+1.14$ [$+0.79$, $+1.47$]
Cube	12	83.72 ± 0.34	20	81.57 ± 1.02	$+2.16$ [$+1.70$, $+2.63$]
Push-D	12	91.72 ± 0.34	20	89.37 ± 1.00	$+2.36$ [$+1.89$, $+2.82$]
Ranking†	12	90.19 ± 1.14	20	88.67 ± 1.42	$+1.52$ [$+0.67$, $+2.37$]
Windows†	12	36.76 ± 0.92	20	35.80 ± 1.08	$+0.95$ [$+0.27$, $+1.61$]
Six planning mean	12	89.72 ± 0.17	20	88.01 ± 0.28	$+1.71$ [$+1.56$, $+1.86$]

Table 22: Agent-12 versus calibration-sensitivity 12-map contraction. Frozen final quality (%).

Workload	Agent-12	Sensitivity-12	Agent−Sensitivity (pp)
Long	93.40	92.87	+0.53
Wall-L	94.00	92.67	+1.33
Push-L	90.00	89.33	+0.67
TwoRoom	86.00	85.33	+0.67
Cube	84.00	82.67	+1.33
Push-D	92.00	90.67	+1.33
Ranking†	91.11	88.89	+2.22
Windows†	37.50	35.71	+1.79
Six planning mean	89.90	88.92	+0.98

Table 23: Native development candidate evaluations for one reference run of each route. A candidate evaluation contains its workload's prescribed development sources and executions.

Route	Maps/workload	Eight-workload total	Relative to 12	Reduction versus Broad
Broad-48	48	384	4.00×	–
Random-12	12	96	1.00×	75%
Agent-12	12	96	1.00×	75%
Current-12	12	96	1.00×	75%

Table 24: Reference-run Agent usage and offline time (s). "Tok." denotes input/output tokens.

Workload	LLM	Tok. in/out	Generation	Pack	Dev score	Agent total	Broad total	Reduction
Long	18	13240/612	3.0	26.4	660	887.4	2937.6	69.79%
Wall-L	6	4820/438	2.0	4.8	96	133.8	432.2	69.04%
Push-L	8	4955/452	2.5	7.2	144	211.7	659.8	67.91%
TwoRoom	6	4810/445	2.0	4.8	120	167.8	538.2	68.82%
Cube	6	4870/447	2.0	4.8	96	133.8	432.2	69.04%
Push-D	7	3760/401	2.0	12.0	108	159.0	514.0	69.07%
Ranking	9	7320/528	2.5	18.0	132	196.5	640.0	69.30%
Windows	10	7240/536	2.5	18.0	144	209.5	688.0	69.55%
Total	70	51015/3859	18.5	96.0	1500	2099.5	6842.0	69.31%

D.6 Candidate-evaluation budget and measured offline cost

The 75% evaluation reduction is exact for the $48 \to 12$ design. Table 24 additionally reports measured offline wall time. Agent total includes shared setup, the LLM call, candidate generation, packing, and native development scoring. Broad total includes the corresponding shared setup, broad candidate generation, packing, and development scoring. Self-hosting gives zero external API cost in every row.

Measured wall-clock reductions range from 67.9% to 69.8% by workload. The aggregate reduction is $1 - 2099.5/6842.0 = 69.3\%$. These offline quantities are separate from online speedups (Equation 8).

Table 25: Final deployment counts, quality changes, and module counts. Quality differences use unrounded fractions before rounding to two decimal places.

Model / task	FP32	Final	Δ quality (pp)	W4 / eligible
Fast-WAM Spatial	490/500	482/500	-1.60	614/614
Fast-WAM Object	498/500	497/500	-0.20	614/614
Fast-WAM Goal	486/500	482/500	-0.80	610/614
Fast-WAM Long	477/500	467/500	-2.00	369/614
LeWM Wall	39/50	47/50	+16.00	72/108
LeWM Push-T	137/150	135/150	-1.33	92/108
LeWM TwoRoom	87/100	86/100	-1.00	72/108
LeWM Cube	39/50	42/50	+6.00	100/108
LeWM Reacher	47/50	46/50	-2.00	108/108
DINO-WM Wall	47/50	49/50	+4.00	66/66
DINO-WM PointMaze	50/50	50/50	+0.00	66/66
DINO-WM Push-T	46/50	46/50	+0.00	42/66
DINO-WM MetaWorld Reach	45/50	44/50	-2.00	72/72
V-JEPA2-AC Ranking†	42/45	41/45	-2.22	160/261
V-JEPA2-AC Windows†	20/56	21/56	+1.79	144/259

Table 26: Parameter-weighted W4 composition and serialized deployment artifact ledger.

Task	Module W4	Parameter W4	W4 payload	Artifact (MB)	Compression
Fast-WAM Spatial	100.00%	100.00%	100.00%	3775.13	3.19×
Fast-WAM Object	100.00%	100.00%	100.00%	3775.13	3.19×
Fast-WAM Goal	99.35%	99.79%	99.58%	3779.93	3.19×
Fast-WAM Long	60.10%	66.63%	49.96%	4531.04	2.66×
LeWM Push-T	85.19%	64.66%	47.77%	14.09	5.13×
LeWM Cube	92.59%	82.33%	69.96%	14.07	5.14×
LeWM Reacher	100.00%	100.00%	100.00%	12.36	5.85×
DINO-WM Wall	100.00%	100.00%	100.00%	39.48	4.26×
DINO-WM PointMaze	100.00%	100.00%	100.00%	37.63	4.48×
DINO-WM Push-T	63.64%	68.30%	51.86%	43.55	3.87×
DINO-WM MetaWorld Reach	100.00%	100.00%	100.00%	57.30	4.90×
V-JEPA2-AC Ranking	61.30%	76.80%	62.34%	950.26	5.55×
V-JEPA2-AC Windows	55.60%	66.88%	50.25%	999.30	5.27×

E FINAL DEPLOYMENT COUNTS AND DESCRIPTIVE AGGREGATES

Table 25 provides the exact numerator/denominator audit for Table 1. One GiB is 2^{30} bytes and one MiB is 2^{20} bytes. W4 share is W4 modules divided by eligible modules in the final deployment record. These counts are neither parameter-weighted nor serialized-byte fractions.

Across the 15 recorded task ratios, median speedup is $2.36\times$, arithmetic-mean speedup is $2.84\times$, and geometric-mean speedup is $2.73\times$. Median peak-memory reduction is 71.46% and arithmetic-mean reduction is 73.84%. These are descriptive, equally weighted summaries of heterogeneous native workloads. The main text emphasizes the full range and individual rows. A ratio of summed times would give a different workload-weighted quantity.

Parameter-weighted precision and artifact size. Module-count W4 share can differ materially from the fraction of parameters assigned to W4. Table 26 joins the selected deployment precision ledger to recorded serialized artifacts. The W4 payload fraction is

$$\frac{4P_4}{4P_4 + 8P_8 + 16P_{16} + 32P_{32}}, \tag{46}$$

where P_b is the number of weight parameters stored at b bits. The payload fraction excludes scales, zero points, padding, bias, and container metadata. Artifact megabytes (MB) equal the recorded byte count divided by 10^6. Compression uses the recorded baseline rather than a theoretical bit-width ratio.

Table 27: Primary workload software environments. CUDA is the version against which PyTorch was compiled, not the driver's maximum supported version. cuDNN denotes the CUDA Deep Neural Network library. The recorded labels use V-JEPA for video joint-embedding predictive architecture, the model underlying the action-conditioned workload.

Environment	Python	PyTorch	PyTorch CUDA	cuDNN	Triton
Fast-WAM	3.10.20	2.7.1+cu128	12.8	9.7.1	3.3.1
LeWM	3.10.20	2.7.1+cu128	12.8	9.7.1	3.3.1
DINO-WM	3.9.19	2.3.0+cu121	12.1	8.9.2	2.3.0
V-JEPA MetaReach	3.10.20	2.7.1+cu128	12.8	9.7.1	3.3.1
V-JEPA RoboCasa	3.10.20	2.7.1+cu128	12.8	9.7.1	3.3.1

Endpoint alignment for module comparisons. For the final endpoint $T_{\mathsf{Q}}, M_{\mathsf{Q}}$ and recorded module ratios, the aligned control values satisfy

$$T_{\mathsf{P}} = \frac{T_{\mathsf{Q}}}{1 + \Delta T_{\text{module}}}, \qquad M_{\mathsf{P}} = \frac{M_{\mathsf{Q}}}{1 - R_{\text{module}}}. \tag{47}$$

The absolute values in Table 30 follow the aligned summary and these relations, up to rounding. This is an endpoint alignment, not an additional independently acquired control run. The paired timing confidence intervals and the independent replay below retain their own statistical and workload definitions. The allocation accounting in Table 5 follows the corresponding liveness summary. Displayed stage sums can differ by 0.01 MiB due to rounding.

F Execution protocol and secondary systems results

F.1 Primary hardware and software stack

The primary host is identified by the driver as an NVIDIA GeForce RTX 4090 (Peripheral Component Interconnect device identifier `10de:2684`) with 49,140 MiB visible memory. PyTorch reports 50,866,487,296 visible CUDA bytes. The device reports compute capability 8.9 with 128 streaming multiprocessors (SMs, architecture SM89), Peripheral Component Interconnect Express (PCIe) Gen4 $\times 16$ capability, a current/default/maximum power limit of 450 watts (W), maximum graphics/SM clock 3,105 megahertz (MHz), and maximum memory clock 10,501 MHz. These software-observed limits/capabilities do not establish sustained benchmark clocks. Software identification cannot establish physical memory modifications or memory-chip type. We therefore report only the model and memory capacity observed through the software interface.

The host uses driver 595.71.05 (reported CUDA compatibility 13.2) and GNU Compiler Collection (GCC) and its C++ compiler (G++) at version 11.4.0. Table 27 lists the environments. Kernels use Triton just-in-time (JIT) compilation, with no `torch.compile` in measured paths.

The separate replication uses a stock 24 GB GPU. Complete-panel timing excludes model loading, packing, compilation, and initialization and measures the warmed evaluation panel. Each timing comparison uses 10 paired runs with alternating AB/BA order and synchronized measurement boundaries. Result-dependent early stopping is disabled for timing and fixed-work replay.

For paired run i, define relative time change

$$r_i = 100 \frac{T_{\text{intervention},i} - T_{\text{control},i}}{T_{\text{control},i}}. \tag{48}$$

Table 28 reports $\text{median}_i\, r_i$ and a percentile bootstrap interval over the 10 paired r_i values.

F.2 Operator and context-boundary applicability across all 15 tasks

Table 29 audits compatible sites for both materialization mechanisms across the deployment suite. QEF compatibility is defined over eligible packed producers. VCF counts only observed context-assembly boundaries whose consumers meet the segmented-read contract.

Per-task QEF coverage ranges from 93.94% to 99.35%. VCF is therefore a narrower structural optimization than QEF: seven tasks expose at least one compatible context boundary, whereas QEF applies to the large majority of packed producers in every family.

Table 28: Paired panel timing: median relative change and paired-bootstrap percentile 95% intervals.

Workload	Design/reference	Median Δ time	95% CI
Push-L	QEF / optimized packed	+2.10%	[+1.69,+2.50]%
Cube	QEF / optimized packed	+2.50%	[+1.98,+3.01]%
Long	QEF / optimized packed	+1.95%	[+1.50,+2.40]%
Wall-D	VCF / packed/common	0.00%	[-0.31,+0.40]%
PointMaze	VCF / packed/common	-5.60%	[-6.20,-5.05]%

Table 29: QEF/VCF applicability by family. "VCF tasks" counts workloads with compatible boundaries.

Family	Tasks	QEF compatible / eligible	QEF coverage	VCF compatible / observed	VCF tasks
Fast-WAM	4	2432/2456	99.02%	1/2	1/4
LeWM	5	520/540	96.30%	0/0	0/5
DINO-WM	4	254/270	94.07%	4/4	4/4
V-JEPA2-AC	2	494/520	95.00%	5/5	2/2
All	15	3700/3786	97.73%	10/11	7/15

Table 30: **Endpoint-aligned module accounting.** Times are in seconds. Memory is peak allocated MiB. The module-on endpoints equal the final deployment endpoints. Module-off absolute values are aligned using the recorded module ratios. ΔT retains the paired timing point estimate. This comparison isolates one module, not FP32-to-final deployment.

Workload	Module	Time off→on	Peak MiB off→on	Memory ↓	ΔT
Push-L	QEF	$3.2485 \rightarrow 3.3167$	$182.17 \rightarrow 158.49$	13.0%	+2.10%
Cube	QEF	$4.0673 \rightarrow 4.1690$	$172.32 \rightarrow 156.64$	9.1%	+2.50%
Long	QEF	$6.4435 \rightarrow 6.5691$	$7074.59 \rightarrow 6766.84$	4.35%	+1.95%
Wall-D	VCF	$5.0301 \rightarrow 5.0301$	$830.91 \rightarrow 830.91$	0.0%	0.00%
PointMaze	VCF	$6.3476 \rightarrow 5.9921$	$898.80 \rightarrow 878.04$	2.31%	−5.60%

F.3 Absolute module endpoints and stage accounting

Tables 30 and 5 preserve the absolute endpoint-aligned summary and its allocation-stage accounting. Module-off time and memory values are reconstructed using Equation 47. Their shared module-on endpoints connect the system and module ratios algebraically. They are not independently measured in additional control trials. The paired timing intervals above use separate AB/BA records.

F.4 Memory metric definitions and secondary peaks

The primary memory metric is peak allocated device memory over a complete warmed workload. Allocated memory counts live tensor and workspace allocations. Reserved memory is the allocator-managed pool, which can include inactive cached blocks. Process-level memory can additionally include runtime and library state. Only allocated memory enters the main live-set equations.

Table 31 preserves the earlier-run diagnostic log in its original MB labeling. All three columns belong to those same earlier runs. The current final deployment bytes and endpoint-aligned module comparisons are in Tables 1 and 4. Headline reductions use allocated memory only.

F.5 Second-hardware replication

Long is omitted from the second-device replication because the available execution setup does not support that workload. The remaining four workloads reproduce the primary device's memory pattern: QEF reduces peaks, while VCF gives both positive and null effects.

F.6 Independent fixed-work replay

The following 320-event replay is retained as an independent experiment. Its workload length and absolute times differ from the complete native panels in Table 1. Its speed factors are computed

Table 31: Earlier-run memory diagnostics (MB as recorded). Allocated, reserved, and process peaks are retained together from the same original log. These are not the current final-deployment endpoints.

Workload	Route	Allocated	Reserved	Process peak
Push-L	Optimized packed	184	248	326
	+QEF	160	224	298
Cube	Optimized packed	176	232	310
	+QEF	160	208	287
Long	Optimized packed	6900	7320	7700
	+QEF	6600	7000	7380
Wall-D	Optimized packed	864	1016	1188
	+VCF	864	1016	1188
PointMaze	Optimized packed	952	1112	1295
	+VCF	930	1080	1258

Table 32: Separate replication on a stock 24 GB GPU. Memory is the peak allocated MB as recorded on that device. These endpoints are not the primary-device final-deployment values.

Workload	Route	Peak allocated	Δ memory	Median Δ time
Push-L	Optimized	186	–	–
	+QEF	162	-12.9%	+2.3%
Cube	Optimized	178	–	–
	+QEF	162	-9.0%	+2.6%
Wall-D	Optimized	868	–	–
	+VCF	868	0.0%	+0.1%
PointMaze	Optimized	958	–	–
	+VCF	936	-2.3%	-5.1%

Table 33: Fixed-work replay over 320 recorded events. Times are in seconds. Specialized denotes QEF for Push-L/Cube/Long and VCF for Wall-D/PointMaze.

Workload	Dense+C	Packed+C	Specialized	S_{quant}	S_{material}	Total
Push-L	14.0	6.00	6.12	2.33×	0.98×	2.29×
Cube	12.0	4.80	4.92	2.50×	0.98×	2.44×
Wall-D	10.8	7.20	7.20	1.50×	1.00×	1.50×
PointMaze	16.2	10.80	10.55	1.50×	1.02×	1.54×
Long	40.0	21.00	21.40	1.90×	0.98×	1.87×

Table 34: Numerical agreement with the corresponding unfused quantized route. Error entries are maximum absolute error / relative-L_2 error. Failures count not-a-number (NaN) outputs, infinity (Inf) outputs, contract violations, and fallback errors.

Workload/design	Cases	FP32 output	FP16 output	Failures
Push-L / QEF	96	7.1e−6/7.6e−7	1.95e−3/2.9e−4	0/0/0
Cube / QEF	96	8.4e−6/9.1e−7	1.95e−3/3.2e−4	0/0/0
Long / QEF	64	9.6e−6/9.4e−7	2.44e−3/4.1e−4	0/0/0
Wall-D / VCF	96	3.6e−6/2.0e−7	1.46e−3/2.5e−4	0/0/0
PointMaze / VCF	96	4.1e−6/2.4e−7	1.95e−3/2.9e−4	0/0/0

within each replay row only. For dense execution D, packed execution P, common scheduling C, and specialized materialization route E, replay reports

$$S_{\text{quant}} = \frac{T(D,C)}{T(P,C)}, \qquad S_{\text{material}} = \frac{T(P,C)}{T(P,E)}. \tag{49}$$

F.7 Numerical agreement

The unfused reference uses identical codes, scales, zero points, precision map, and inputs. Absolute and relative tolerances are atol and rtol. FP32-output checks use $\mathrm{atol} = \mathrm{rtol} = 10^{-5}$ and relative-$L_2 < 10^{-5}$. FP16-output checks use $\mathrm{atol} = \mathrm{rtol} = 3 \times 10^{-3}$ and relative-$L_2 < 10^{-3}$.

QEF and VCF match each paired quantized reference within the recorded numerical contract.

G SCOPE

Selection and contraction scope. The controlled selector experiment fixes one 12-map pool per workload. Its resampling evidence concerns selection within that pool. The auxiliary experiment compares Broad-48, Random-12, Agent-12, and a sensitivity-based 12-map contraction, with 12 Agent and 20 Random construction runs. The reported Agent results are tied to the self-hosted Qwen3-8B snapshot, prompt, and sampling configuration in Appendix D.

Endpoints and quality. Planning, episode-clustered ranking, and demonstration-clustered windows measure different endpoints. The six-planning summary is the primary quality aggregate. The all-eight mean is descriptive.

Execution scope. Deployment gains use FP32 as the baseline and a fully warmed workload. They exclude loading, packing, compilation, initialization, and offline search. QEF/VCF require their stated operator, storage, and numerical contracts. Byte savings alone do not determine runtime.