FROM KNOBS TO ACTIONS: AGENTIC BAYESIAN OPTIMIZATION FOR SELF-IMPROVING SYSTEM TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern systems such as Large Language Model (LLM) inference engines and databases expose large configuration spaces with strongly interacting parameters. Their workloads can change, making automated tuning essential. Bayesian optimization (BO) is sample-efficient for expensive black-box functions, but BO-based autotuners often underuse runtime metrics and parameter relationships, require many evaluations in high-dimensional spaces, and adapt slowly to workload shifts. We observe that transitions from suboptimal to better configurations follow recurring directions in parameter space. These directions remain partly stable across similar workloads, and LLMs can infer candidate directions from system knowledge. Building on this observation, we propose **EvoTune**, an agentic BO framework that searches over semantic Actions and refines them using trial evidence. EvoTune maintains an Action–Metric–Objective Graph to combine LLM-based diagnosis with model-based Action selection, while BO optimizes execution magnitudes and evidence and feasibility checks govern Action and graph updates. Across eight static workloads, EvoTune achieves the best observed endpoint, with average and maximum gains over a common initial reference of 33.47% and 55.19%, respectively. Under workload shifts, its post-switch throughput exceeds the strongest compared baseline by 11.42% on databases and 25.79% on Spark.

1 INTRODUCTION

Modern computing systems, including LLM inference engines, databases, and distributed data-processing frameworks, expose interacting controls over memory, caching, concurrency, and scheduling. Their effective settings depend on workload and hardware, making tuning a recurring task (Kwon et al., 2023; Van Aken et al., 2017; Zaharia et al., 2012). Bayesian Optimization (BO) offers sample-efficient search for expensive black-box objectives (Snoek et al., 2012; Shahriari et al., 2016). However, finding effective configurations remains costly when parameters interact and workloads change.

Three challenges motivate our work. First, runtime metrics reveal system bottlenecks, but their relationships with coordinated parameter changes and the final objective are often implicit. Second, searching over many interacting knobs remains expensive because useful adjustments may require changing several parameters together. Third, workload shifts can invalidate previously useful configurations and predictive relationships, while restarting the search discards potentially reusable knowledge (Bogunovic et al., 2016). These challenges call for a representation that connects runtime evidence to coordinated adjustments and can evolve as new observations arrive.

Useful tuning changes often serve recognizable system-level purposes, such as relieving memory pressure or improving batching efficiency through coordinated adjustments to several knobs. Configurations facing similar bottlenecks may therefore benefit from similar adjustment directions, even when the appropriate magnitudes differ. Our preliminary experiments support this intuition through two observations: directions from initial configurations to their best observed configurations concentrate around a small number of recurring patterns, and LLM-generated initial Actions align with these measured directions. **Our central insight is to separate what coordinated parameter changes to make from how strongly to apply them.** We represent these changes as semantic

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.

Actions, allowing system knowledge to inform candidate directions while measured outcomes guide their numerical execution.

Building on this insight, we propose *EvoTune*, an agentic BO framework that searches over semantic Actions and refines them using trial evidence. EvoTune maintains an evolving *Action–Metric–Objective Graph* that connects Actions to runtime metrics and the optimization objective. The LLM agent diagnoses bottlenecks from current measurements and trial history, follows graph links to identify applicable Actions, and uses model-based Action sensitivity to prioritize candidates. BO directly models the objective as a function of Action strengths and jointly optimizes the strengths of the selected Actions. This focuses acquisition optimization on a small set of coordinated adjustments, which a compiler translates into feasible configurations. To adapt to changing bottlenecks, EvoTune uses trial feedback to revise Action directions and graph relationships, adopting revisions only after evidence and feasibility checks.

We evaluate EvoTune across eight static workloads and three workload-switch experiments on vLLM, PostgreSQL, and Spark. Under matched evaluation budgets, EvoTune achieves the best observed endpoint on every static workload, with average and maximum gains of 33.47% and 55.19% over a common initial reference. After workload switches, its final throughput exceeds the strongest compared baseline by 11.42% on databases, 25.79% on Spark, and 8.77% on inference. Spark ablations further show that fixing Action directions or graph topology limits adaptation, removing graph-guided Action selection slows recovery, and removing LLM guidance lowers final performance.

Our contributions are as follows:

- We introduce a semantic Action representation for coordinated system tuning, motivated by empirical evidence of directional concentration and alignment between LLM-generated Actions and observed tuning directions in vLLM.
- We develop EvoTune, combining agent-guided diagnosis and graph-guided Action selection with BO over Action strengths, while using trial evidence to govern revisions to graph relationships and Action directions.
- We evaluate EvoTune across eight static workloads and three workload-switch experiments, with Spark component ablations examining the roles of Action selection, representation updates, and LLM guidance.

2 Background and Related Work

Bayesian optimization. For workload w, evaluating a feasible configuration $\mathbf{x} \in \mathcal{X}_w$ yields runtime metrics $\mathbf{m} \in \mathbb{R}^M$ and an objective y. The tuning target is

$$\mathbf{x}_w^\star \in \arg\max_{\mathbf{x}\in\mathcal{X}_w} f_w(\mathbf{x}), \qquad f_w(\mathbf{x}) = \mathbb{E}[y \mid \mathbf{x}, w]. \tag{1}$$

Here $\mathcal{X}_w$ is a feasible mixed configuration space; a latency objective can be negated for maximization. Runtime metrics serve as diagnostics or constraints. BO fits a probabilistic surrogate to past trials and selects candidates with an acquisition function (Frazier, 2018); expected improvement is $\mathbb{E}[(Y - b_t)_+]$, where b_t is the incumbent and $(u)_+ = \max(u, 0)$ (Jones et al., 1998).

BO for system tuning. SCOOT handles performance objectives and configuration constraints in LLM inference tuning (Cheng et al., 2025). OtterTune uses internal runtime metrics and workload mapping to reuse database tuning experience (Van Aken et al., 2017). LlamaTune reduces the search dimension through random embeddings (Kanellis et al., 2022). DOT combines dynamic knob selection with online sampling to adapt the tuning search (Wang et al., 2025). Together, these approaches exploit runtime observations, dimensionality reduction, and adaptive knob selection. In particular, a statistical reduction such as a random embedding need not have a reliable system-semantic interpretation. EvoTune instead represents recurring coordinated changes as semantic Actions and audits their effects and applicability from trial memory.

LLMs within BO. GPTuner uses documentation-derived knowledge to guide database knob selection and search ranges (Lao et al., 2024). LLAMBO places LLMs within warmstarting, surrogate

prediction, and iterative candidate generation (Liu et al., 2024). LMABO uses the current optimization state to select an acquisition function at each iteration (Ngo et al., 2026). More broadly, ReAct interleaves reasoning with actions in an external environment (Yao et al., 2023), and Reflexion uses verbal feedback and episodic memory to guide subsequent attempts (Shinn et al., 2023). Within this broader use of feedback, EvoTune represents system-semantic Actions and their Action–Metric–Objective Graph, and uses evidence gates to revise graph relationships and executable directions.

3 Empirical Motivation: Recurring Tuning Directions

Our intuition is that useful configuration changes are shaped by recurring system bottlenecks. Relieving memory pressure or improving batching efficiency often requires coordinated adjustments to several interacting knobs. Configurations facing similar bottlenecks may therefore benefit from similar adjustment directions, even when the appropriate adjustment magnitudes differ. This suggests that useful tuning directions may concentrate around a compact set of patterns with recognizable system-level meanings. Because documentation describes the roles and interactions of system parameters, LLMs may be able to translate this knowledge into candidate directions that align with those observed in actual tuning. These patterns motivate semantic **Actions**: coordinated parameter directions whose execution magnitudes can be optimized separately.

We examine this intuition through two preliminary experiments on vLLM. The first tests whether measured tuning directions concentrate around a small set of directional prototypes. The second tests whether LLM-generated initial Actions align with their corresponding measured directions. All experiments use the same vLLM version, parameter constraints, and output-token throughput objective as the main study.

3.1 Directional concentration

Data collection. We simulate five serving scenarios: long-document input, batched text input, text classification, knowledge-retrieval inference, and streaming input. For each optimization run, BO starts from an initial configuration and iteratively selects configurations in the system's parameter space using measured output-token throughput. After multiple optimization rounds, we retain the configuration with the highest observed throughput in that run. The displacement from the initial configuration to this best observed configuration defines a measured tuning direction. Each direction therefore summarizes the net configuration change found along a BO trajectory.

Directional analysis. The collected directions are represented by ten-dimensional parameter vectors. We first use a fixed seed to partition the measured directions into a 70% development set and a 30% held-out set. Clustering is then performed only on the development directions and identifies eight empirical groups, labeled A01–A08 for subsequent analysis. We normalize each nonzero measured direction $\mathbf{v}_i$ as

$$\mathbf{d}_i = \frac{\mathbf{v}_i}{\|\mathbf{v}_i\|_2}. \tag{2}$$

We fit K directional prototypes $\{\boldsymbol{\mu}_k\}_{k=1}^{K}$ on the normalized development vectors and normalize every nonzero fitted centroid $\mathbf{c}_k$ to $\boldsymbol{\mu}_k = \mathbf{c}_k/\|\mathbf{c}_k\|_2$. We evaluate held-out coverage at angular tolerance τ:

$$C_K(\tau) = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{\mathbf{d}_i \in \mathcal{D}_{\text{test}}} \mathbf{1}\left[\max_{k \le K} \boldsymbol{\mu}_k^\top \mathbf{d}_i \ge \cos(\tau)\right]. \tag{3}$$

We use $\cos(\tau) = 0.80$, corresponding to $\tau \approx 36.9^\circ$. The matched isotropic reference uses normalized Gaussian directions with the same train/test sizes and fitting procedure.

Result. Figure 1 shows clear geometric structure in the measured directions. The first two principal components explain 45.7% of the total variance. At $K = 8$, coverage reaches 95.8%. Across the measured directions, the average pairwise cosine similarity is 0.809 within an empirical cluster, 0.171 across clusters. These results indicate that the BO-derived configuration changes have a compact directional structure, motivating a library of candidate tuning directions.

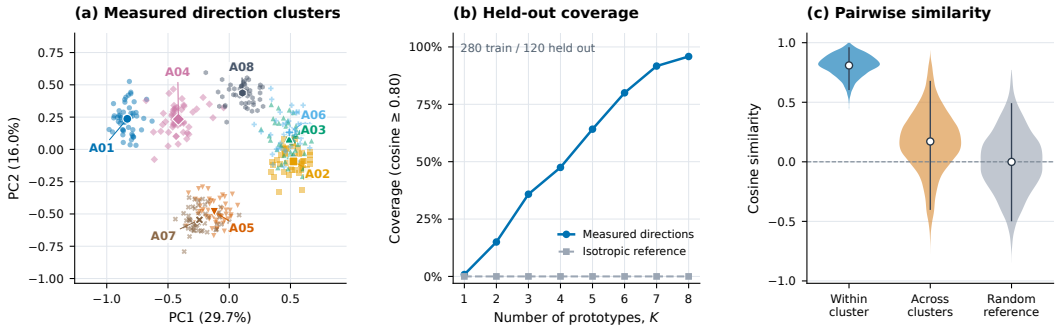

Figure 1: Measured tuning-direction geometry. (a) PCA projection of BO-derived tuning directions; A01–A08 denote centroids of clusters identified from the development set. (b) Held-out coverage at cosine threshold 0.80 for increasing prototype count K after a seeded 70%/30% development/held-out split; the dashed curve uses isotropic directions. (c) Pairwise cosine distributions within clusters, across clusters, and for independent isotropic pairs. White markers denote means and vertical bars denote empirical 2.5–97.5% ranges.

3.2 Semantic alignment with measured directions

Setup. We next examine whether LLM-generated initial Actions align with the empirical directions obtained from BO. We compare eight initial Actions with the eight empirical clusters identified from the development directions in a common parameter coordinate space, using the least-aligned direction in each matched cluster to summarize alignment. For initial Action $\mathbf{a}_\ell$ and its corresponding cluster $\mathcal{C}_\ell$, we define

$$A_{\min}(\mathbf{a}_\ell) = \min_{\mathbf{v}_i \in \mathcal{C}_\ell} \frac{\mathbf{a}_\ell^\top \mathbf{v}_i}{\|\mathbf{a}_\ell\|_2 \, \|\mathbf{v}_i\|_2}. \tag{4}$$

Figure 2 reports one minimum for each Action. For this comparison, Actions and empirical clusters are indexed by their recorded correspondence A01–A08. Each score summarizes all pairwise comparisons in its matched cluster and therefore describes the least-aligned measured direction, rather than alignment with a cluster center. The normalization in Eq. 4 removes vector magnitude, so the score measures directional agreement in the common coordinate representation.

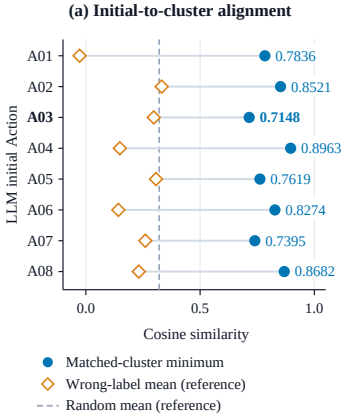

Figure 2: Matched-cluster alignment minima (blue) and geometric references (orange and dashed gray).

Result. Figure 2 shows positive directional agreement for all eight Actions, with the lowest cosine similarity of **0.7148** attained by **A03**. The blue circles represent the least-aligned measured direction within each corresponding cluster. For context, the orange diamonds show each initial Action's average cosine with the seven training-derived prototypes assigned to other labels, while the gray dashed line shows the average nearest-prototype cosine for independent isotropic unit vectors in the same coordinate space. These references use different aggregations from the matched-cluster minima. The connecting segments associate the two statistics for each Action.

4 EvoTune

EvoTune represents coordinated parameter changes as semantic *Actions*. The agent uses an Action–Metric–Objective Graph, runtime measurements, and trial memory to select relevant Actions (Figure 3). Bayesian optimization (BO) models the objective as a function of Action strengths. A

compiler turns the chosen strengths into legal settings, and trial evidence may revise the Actions or graph.

Let t index charged trials and v index the current validated representation: graph, Action directions, and reference configuration. Several trials may share the same v.

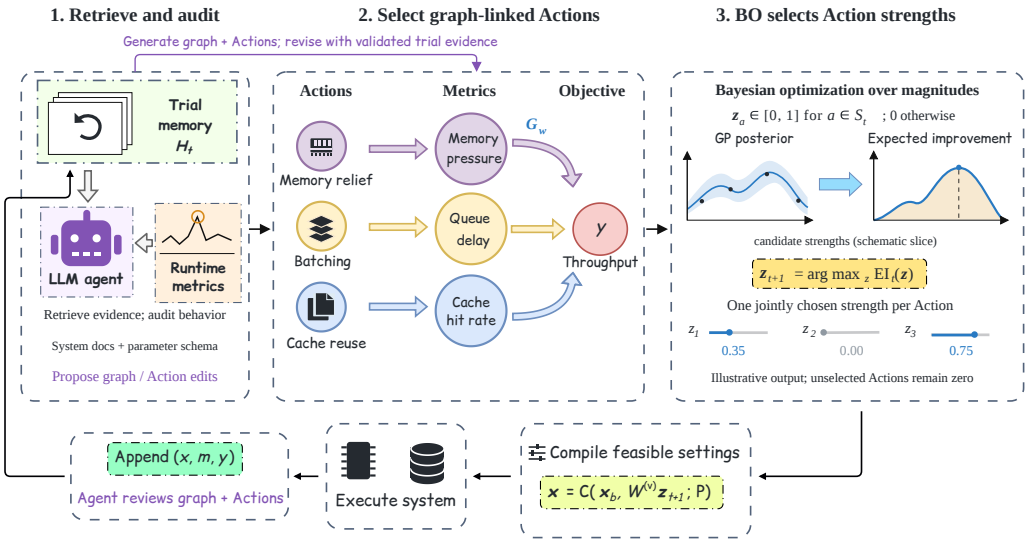

Figure 3: EvoTune's loop: measurements and graph links help the agent select Actions; BO sets their strengths. The compiler yields executable settings, and trial evidence may revise the representation.

4.1 Graph and Direction-Vector Construction

At version v, the graph $\mathcal{G}^{(v)}$ contains L Actions, M runtime metrics, and one objective node Y (drawn as y in Figure 3). Its active links $E^{(v)}$ connect Actions to metrics they may affect and metrics to the objective they help diagnose.

The agent proposes each Action's coordinated parameter changes from documentation and trials; schema checks reject unsupported or illegal settings. For d encoded parameters, $W^{(v)} \in \mathbb{R}^{d \times L}$ stores the Action directions as columns $\mathbf{w}_a^{(v)}$. Each nonzero column is normalized to a maximum absolute entry of one. The coefficient $z_a \in [0, 1]$ controls Action a's strength, with zero turning it off. The vector $\mathbf{z}$ collects all strengths. Given a feasible reference configuration $\mathbf{x}_b^{(v)}$, the resulting setting is

$$\mathbf{x}(\mathbf{z}) = C\left(\mathbf{x}_b^{(v)}, W^{(v)}\mathbf{z}; \mathcal{P}\right), \qquad \mathbf{z} \in \mathcal{Z}_v \subseteq [0, 1]^L. \tag{5}$$

The compiler C applies the combined change $W^{(v)}\mathbf{z}$ to the reference and maps it to legal integer, categorical, or continuous settings. The schema $\mathcal{P}$ specifies parameter domains and applicability; $\mathcal{Z}_v$ contains strengths that compile to feasible settings. Zero strength gives $\mathbf{x}(\mathbf{0}) = \mathbf{x}_b^{(v)}$. A zero direction forces its coefficient to zero, and previously tested settings are filtered before execution.

4.2 Objective Surrogate Model

For workload w, each completed trial provides Action strengths $\mathbf{z}$, a measured objective y, and, when available, runtime metrics. We model the objective as a function of Action strengths:

$$y = h_w^{(v)}(\mathbf{z}) + \epsilon_y. \tag{6}$$

Here $h_w^{(v)}(\mathbf{z})$ is the latent expected objective under the compiled setting $\mathbf{x}(\mathbf{z})$, and ϵ_y is zero-mean observation noise. We fit a Matérn-kernel Gaussian process (Rasmussen & Williams, 2006) to compatible observed $(\mathbf{z}, y)$ pairs. Its posterior gives an expected objective and uncertainty for each feasible strength vector.

4.3 JOINT AGENT–MODEL DIAGNOSIS

At trial t, the agent compares current metrics with the reference and reviews compatible trial history $\mathcal{H}_t$, including failures. It identifies diagnostic metrics and assigns each metric j a confidence $q_{j,t} \in [0, 1]$. At most three metrics are retained in $\mathcal{B}_t$. Following graph links from these metrics gives their eligible parent Actions.

The surrogate model helps prioritize these candidates. Let $\mu_t(\mathbf{z})$ be its posterior mean, and let $\mathcal{Q}_t \subseteq \mathcal{Z}_v$ be a nonempty finite set of feasible strength vectors. For a candidate Action $a \in A_t$, define

$$r_{a,t} = \frac{1}{|\mathcal{Q}_t|} \sum_{\mathbf{z} \in \mathcal{Q}_t} \left| \frac{\partial \mu_t(\mathbf{z})}{\partial z_a} \right|. \tag{7}$$

Here $|\mathcal{Q}_t|$ is the number of feasible sensitivity-evaluation points and z_a is Action a's normalized strength. The derivative measures how sensitive the predicted objective is to that strength near these points. Averaging its absolute value summarizes parent-Action sensitivity for metric prioritization. The selection rule and a worked example are given in the appendix. After selecting at most three metrics, S_t is the union of all their eligible parent Actions; there is no additional Action-count limit. If the evidence is too uncertain, the controller can instead run a budgeted exploratory trial.

4.4 BAYESIAN OPTIMIZATION OF ACTION STRENGTHS

BO optimizes the strengths of Actions in S_t, with the remaining coefficients fixed at zero. Define the restricted search space and expected-improvement rule by

$$\begin{aligned} &\mathcal{Z}_{v,S_t} = \{\mathbf{z} \in \mathcal{Z}_v : z_a = 0 \text{ for all } a \notin S_t\}, \\ &\mathbf{z}_{t+1} \in \arg\max_{\mathbf{z} \in \mathcal{Z}_{v,S_t}} \alpha_t(\mathbf{z}), \qquad \alpha_t(\mathbf{z}) = \mathbb{E}\Big[\big(h_w^{(v)}(\mathbf{z}) - b_t\big)_+ \mid \mathcal{H}_t\Big]. \end{aligned} \tag{8}$$

Here b_t is the current best estimated latent objective and $(u)_+ = \max(u, 0)$. The expectation uses the Gaussian-process posterior; α_t is expected improvement. We optimize it with multiple starts (Wilson et al., 2018; Balandat et al., 2020), compile $\mathbf{z}_{t+1}$ through Eq. (5), and evaluate the resulting configuration. Graph screening reduces the number of free coefficients in this acquisition search. The final selected setting receives independent confirmation.

4.5 UPDATING GRAPH LINKS AND ACTION DIRECTIONS

Trial memory records workloads, Action strengths, compiled settings, metrics, objectives, and failures. A review may propose adding graph links E_t^+, removing links E_t^-, or changing Action directions ΔW_t. The candidate graph in Figure 4(a) is

$$\widehat{E}_t = (E^{(v)} \setminus E_t^-) \cup E_t^+. \tag{9}$$

Here $E^{(v)}$ is the current link set; E_t^+ and E_t^- are proposed additions and removals. They remain separate from the active graph until the evidence check passes. Disabling an Action removes its outgoing links and fixes its coefficient to zero.

Figure 4(b) illustrates a proposed change to Action direction $\mathbf{w}_a^{(v)}$: the ath column $\Delta\mathbf{w}_{a,t}$ of ΔW_t forms candidate $\widehat{\mathbf{w}}_{a,t}$, which is normalized by its largest absolute entry when nonzero. A zero direction disables the Action. Joint link and direction changes are checked and adopted together.

The gate in Figure 4(c) checks proposed links against held-out trials: Action–metric links should match observed metric responses, and metric–objective links should match observed associations. Revised Actions must also compile to feasible settings. Nonzero direction changes require matched-cost paired trials with $\Delta y > \delta$, where Δy is candidate-minus-control objective and δ is a prespecified threshold. Link updates are assessed using relationship evidence, while disabling an Action requires evidence that it is invalid or redundant. Accepted edits advance v; failed checks retain the current version, and later deterioration can trigger rollback.

Initial trials, exploratory trials, validation, failures, and confirmation share the budget; other attempts cannot use the T_c reserve. Fitting one exact Gaussian process to n trials costs $O(n^3)$. Graph-guided Action selection reduces the number of free coefficients in acquisition optimization.

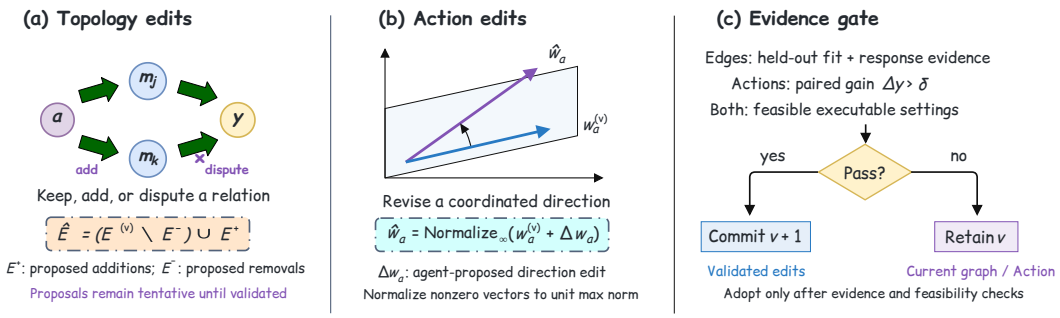

Figure 4: Representation revision: (a) propose link edits; (b) propose a direction change; (c) adopt changes after response-evidence and feasibility checks.

Algorithm 1 EvoTune: graph-guided Action selection and Bayesian optimization

Require: Parameter schema $\mathcal{P}$; LLM agent Ψ; trial budget T; confirmation reserve T_c

Reserve T_c; run budgeted initial trials; initialize history $\mathcal{H}$ and version $(\mathcal{G}^{(v)}, W^{(v)}, \mathbf{x}_b^{(v)})$
while charged attempts $< T - T_c$ **do**
  Retrieve compatible history $\mathcal{H}_t$ and fit the objective surrogate $h_w^{(v)}$
  Agent Ψ diagnoses metrics; follow graph links and Eq. (7) to select S_t
  If diagnosis is uncertain, choose a budgeted exploratory trial; otherwise maximize Eq. (8)
  Compile by Eq. (5); execute, charge, and record the trial
  if a representation review is due and its checks fit the remaining budget **then**
    Propose and validate link or direction changes; advance v only on acceptance
return independently confirmed incumbent using reserve T_c

5 Experimental Evaluation

We evaluate EvoTune's configuration quality under fixed trial budgets, adaptation to workload shifts, and the roles of representation updates and graph-guided search. Experiments include eight static workloads across three systems, three workload-switch experiments, and a Spark ablation with four variants. All methods receive the same evaluation budget within each workload.

5.1 Experimental Setup

LLM agent. EvoTune uses `deepseek-flash` as its API-based LLM agent for metric diagnosis and graph and Action revision proposals. Prompt templates and API configuration are provided in Appendix J.

Systems and workloads. Hardware, software versions, search scope, and budgets are listed in Table 1. vLLM (Kwon et al., 2023) serves Llama 2-7B (Touvron et al., 2023) in bfloat16 with a 4,096-token limit under KV, Batch, Prefill, and Prefix profiles. PostgreSQL (The PostgreSQL Global Development Group, 2026b) runs TPC-H-derived queries (Transaction Processing Performance Council, 2026) at SF4 (7.94 GiB; four clients, Q1–Q22 after two warm-ups) and pgbench (The PostgreSQL Global Development Group, 2026a) at SF600 (eight clients, 35 s). TPC-H reports per-trial latency.

Table 1: Systems and search scope. CPU values are container quotas; Spark settings are submitted controls, not all independently effective.

	vLLM inference	PostgreSQL	Spark
CPU family	Xeon Cascadelake	Xeon Skylake	EPYC 7T83
CPU quota / RAM	10 / 72 GiB	10 / 72 GiB	32 / 60 GiB
Execution	A100 PCIe, 40 GiB	CPU only	1 master, 1 worker
Software	vLLM 0.11.2 (V1)	PostgreSQL 16.15	Spark 3.5.7; RDD
Search scope	10 configuration dimensions	5 TPC-H / 6 pgbench	12 submitted settings
Reporting budget	45 trials	35 trials	45 trials

Spark's RDD jobs (Zaharia et al., 2012) use a 16-core, 32-GiB worker and a 4-GiB driver. Sort processes 20 million deterministic rows (1.84 GB) without full-output writes; PageRank (Page et al., 1999) applies eight updates to a deterministic graph with one million designated vertices and five million edges (68.89 MB), returning 993,004 ranks.

Protocol. All methods share initial configurations and the initial-sampling seed within each experiment. Curves stop at the budgets in Table 1, without extrapolation. Inference throughput includes request drain but excludes startup and warm-up; Prefill replays 128 inputs for 90 s, allowing prefix-cache reuse. Spark divides result counts by `spark-submit` duration, including launch and exit, with a 180-s timeout. Workload execution dominates BO computation, so charged trials approximate evaluation cost.

5.2 Optimization Performance

Figures 5–6 compare the achieved performance. Gains are averaged equally across the eight workloads. EvoTune achieves the best observed endpoint on each workload, with an average performance improvement of 33.47% and a maximum improvement of 55.19%. **Inference.** KV and Prefix exhibit wider observed throughput improvements than Prefill, while Batch sustains the highest output-token rate. The baseline ordering also depends on the profile: Pure LLM has the highest baseline endpoint on KV, Batch, and Prefill, whereas OtterTune leads on Prefix. For KV, Batch, Prefill, and Prefix, EvoTune achieves throughput improvements of 49.45%, 48.80%, 11.54%, and 55.19%, respectively. **Databases.** TPC-H shows a larger percentage improvement than pgbench. EvoTune reaches 10.21 s query latency and 2,128.26 TPS, respectively, with a latency reduction of 32.56% and a throughput increase of 17.98%. **Spark.** The two jobs yield different baseline rankings: SCOOT has the highest baseline rate on Sort, whereas Pure LLM leads on PageRank. EvoTune reaches 917.97 thousand rows/s and 39.62 thousand returned ranks/s, with throughput improvements of 42.20% and 10.05%, respectively. **Sensitivity to initialization.** Across the three systems, we repeated the optimization experiments with different initial parameter values. The curves changed only slightly, and EvoTune retained the best final performance.

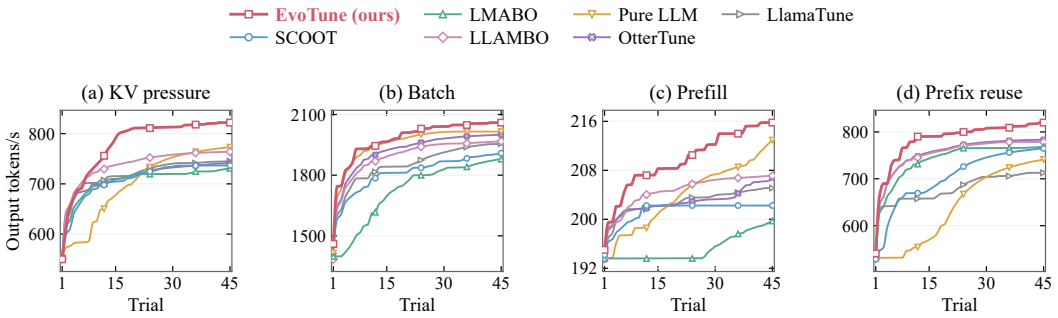

Figure 5: vLLM tuning under four request profiles over 45 evaluations; higher output throughput is better.

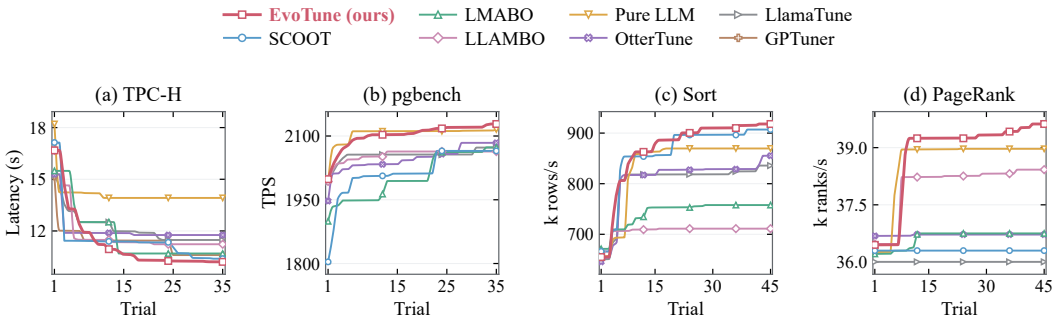

Figure 6: PostgreSQL and Spark tuning. Lower query latency is better for TPC-H; higher transaction or result-record throughput is better otherwise. Budgets are 35 database and 45 Spark evaluations; GPTuner appears only for TPC-H.

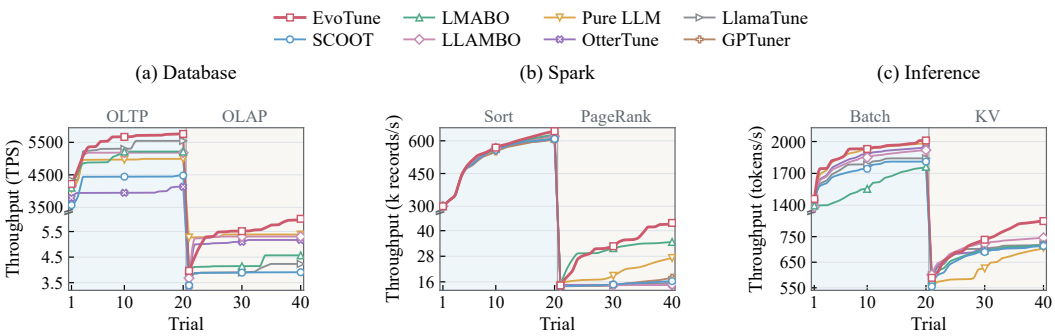

Figure 7: Workload-switch experiments in three systems. The workload changes at trial 21 of each 40-trial optimization run. Marked vertical intervals are compressed; tick values retain physical units. GPTuner is included only for Spark.

5.3 Adaptation to Workload Drift

We conduct 40-trial workload-switch experiments, changing workloads at trial 21 from database point queries to aggregation/sort, Spark Sort to PageRank, and inference Batch to KV pressure. At trial 40, EvoTune's throughput exceeds the strongest compared baseline by 11.42%, 25.79%, and 8.77%, respectively. These results demonstrate its adaptation performance across the three systems under the evaluated workload shifts.

5.4 Ablation Study

Figure 8 compares EvoTune with four variants under a Spark workload switch. *Fixed Actions* retains the initial Action directions while allowing graph updates and Action selection. *Fixed Graph* retains the initial graph topology while allowing Action directions and numerical models to update. *Direct Objective Surrogate* uses the same Action library but optimizes all Action coefficients without graph-guided Action selection. The no-LLM variant evaluates the contribution of LLM guidance.

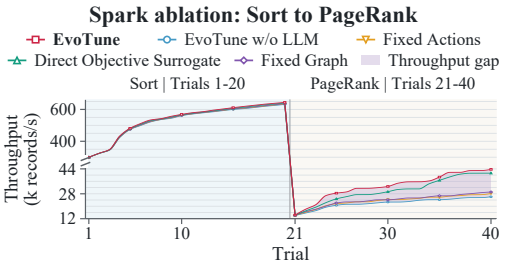

Figure 8: Spark component ablations over 40 trials, switching from Sort to PageRank at trial 21. Higher throughput is better. The vertical axis contains a break. Shading marks the gap between EvoTune and its no-LLM variant.

After the switch to PageRank, *Fixed Actions* and *Fixed Graph* finish below EvoTune, supporting the role of updating both Action directions and graph topology in this experiment. The no-LLM variant achieves still lower final throughput. *Direct Objective Surrogate* approaches EvoTune's final performance but requires more trials to reach comparable throughput. This is consistent with the benefit of graph-guided Action selection, which focuses acquisition optimization on adjustments relevant to the current workload.

6 Conclusion

EvoTune builds on the observation that useful parameter changes follow recurring directions, representing them as Actions. Using runtime metrics, trial history, and the Action–Metric–Objective Graph, the LLM agent chooses which Actions to apply, while BO determines their strengths. Results from these trials then guide validated updates to the Actions and graph. Across eight static workloads on vLLM, PostgreSQL, and Spark, EvoTune achieves the best observed performance, with average and maximum gains of 33.47% and 55.19% over the common initial reference. Workload-switch experiments and Spark ablations further support selecting relevant Actions and updating their directions and graph links to adapt tuning to workload changes.

AI USE STATEMENT

We used GPT to assist with literature review, manuscript drafting, mathematical design review, diagram creation, and experiment script development. The authors take full responsibility for the accuracy and integrity of the paper, including the verification of measurements, cited sources, and consistency between the proposed method and its implementation.

REPRODUCIBILITY STATEMENT

The implementation of EvoTune and its variants, along with the baselines and scripts for reproducing the experiments, is available at `https://anonymous.4open.science/r/EvoTune-9658`.

REFERENCES

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A framework for efficient monte-carlo Bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f5b1b89d98b7286673128a5fb112cb9a-Abstract.html.

Ilija Bogunovic, Jonathan Scarlett, and Volkan Cevher. Time-varying Gaussian process bandit optimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pp. 314–323, 2016. URL https://proceedings.mlr.press/v51/bogunovic16.html.

Ke Cheng, Zhi Wang, Wen Hu, Tiannuo Yang, Jianguo Li, and Sheng Zhang. SCOOT: SLO-oriented performance tuning for LLM inference engines. In *Proceedings of the ACM on Web Conference 2025*, pp. 829–839, 2025. doi: 10.1145/3696410.3714930. URL https://doi.org/10.1145/3696410.3714930.

Peter I. Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018. URL https://arxiv.org/abs/1807.02811.

Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998. doi: 10.1023/A:1008306431147.

Konstantinos Kanellis, Cong Ding, Brian Kroth, Andreas Müller, Carlo Curino, and Shivaram Venkataraman. LlamaTune: Sample-efficient DBMS configuration tuning. *Proceedings of the VLDB Endowment*, 15(11):2953–2965, 2022. doi: 10.14778/3551793.3551844. URL https://www.vldb.org/pvldb/vol15/p2953-kanellis.pdf.

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 *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Jiale Lao, Yibo Wang, Yufei Li, Jianping Wang, Yunjia Zhang, Zhiyuan Cheng, Wanghu Chen, Mingjie Tang, and Jianguo Wang. GPTuner: A manual-reading database tuning system via GPT-guided Bayesian optimization. *Proceedings of the VLDB Endowment*, 17(8):1939–1952, 2024. doi: 10.14778/3659437.3659449. URL https://www.vldb.org/pvldb/vol17/p1939-tang.pdf.

Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large language models to enhance Bayesian optimization. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=OOxotBmGol.

Giang Ngo, Dat Phan Trong, Dang Nguyen, Sunil Gupta, and Svetha Venkatesh. Adaptive acquisition selection for Bayesian optimization with large language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/f1b7a29fe9759d4c9d23f78738e40b42-Abstract-Conference.html.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, 1999. URL http://ilpubs.stanford.edu:8090/422/. An earlier manuscript dated January 29, 1998 is available from the University of California, Riverside at https://www.cs.ucr.edu/~vagelis/classes/CS172/publications/page98pagerank.pdf (accessed September 26, 2026).

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. URL https://gaussianprocess.org/gpml/.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1): 148–175, 2016. doi: 10.1109/JPROC.2015.2494218. URL https://lips.cs.princeton.edu/pdfs/shahriari2016loop.pdf.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25, 2012. URL https://proceedings.neurips.cc/paper/2012/hash/05311655a15b75fab86956663e1819cd-Abstract.html.

The PostgreSQL Global Development Group. *pgbench: Run a Benchmark Test on PostgreSQL*, 2026a. URL https://www.postgresql.org/docs/16/pgbench.html. PostgreSQL 16 documentation. Accessed September 25, 2026.

The PostgreSQL Global Development Group. *PostgreSQL 16 Documentation*, 2026b. URL https://www.postgresql.org/docs/16/. Accessed September 25, 2026.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. URL https://arxiv.org/abs/2307.09288.

Transaction Processing Performance Council. TPC-H: A decision support benchmark, 2026. URL https://www.tpc.org/tpch/. Accessed September 25, 2026.

Dana Van Aken, Andrew Pavlo, Geoffrey J. Gordon, and Bohan Zhang. Automatic database management system tuning through large-scale machine learning. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pp. 1009–1024, 2017. doi: 10.1145/3035918.3064029. URL https://db.cs.cmu.edu/papers/2017/p1009-van-aken.pdf.

Yifan Wang, Debabrota Basu, Pierre Bourhis, Romain Rouvoy, and Patrick Royer. DOT: Dynamic knob selection and online sampling for automated database tuning. *Proceedings of the VLDB Endowment*, 19(4):589–602, 2025. doi: 10.14778/3785297.3785302. URL https://www.vldb.org/pvldb/vol19/p589-wang.pdf.

James Wilson, Frank Hutter, and Marc Peter Deisenroth. Maximizing acquisition functions for Bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/498f2c21688f6451d9f5fd09d53edda7-Abstract.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pp. 15–28. USENIX Association, 2012. URL https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/zaharia.

A Scope and Notation

This appendix expands the method and experimental protocol specified in the main paper. It uses the same objective surrogate, Action-selection rule, representation-update gate, notation, and experimental scope.

EvoTune separates three decisions. The agent interprets runtime evidence and identifies relevant semantic Actions. Bayesian optimization (BO) determines their execution strengths. The compiler maps the combined Action direction to an executable system configuration. Graph links guide the selection decision; the objective GP predicts the final performance measure directly.

Table 2: Notation shared with the main method.

Symbol	Meaning
d	Number of encoded system parameters.
L	Number of semantic Actions.
M	Number of runtime metrics.
w	Workload and its fixed evaluation protocol.
t	Index of charged trials, including validation and failed attempts.
v	Version of the validated representation.
$\mathcal{P}$	Parameter domains, applicability, and feasibility schema.
$\mathcal{G}^{(v)}, E^{(v)}$	Active Action–Metric–Objective graph and its links.
$\mathbf{x}_b^{(v)}$	Feasible reference configuration.
$W^{(v)} \in \mathbb{R}^{d\times L}$	Action directions, stored as columns.
$\mathbf{z} \in [0,1]^L$	Action strengths, not system-parameter values.
$\mathcal{Z}_v$	Strength vectors that compile to feasible settings.
$\mathcal{H}_t$	Compatible trial history available at trial t.
$\mathcal{B}_t, q_{j,t}$	Diagnostic metric set and metric-confidence scores.
A_t, S_t	Candidate Actions and the eligible parent union of selected metrics.
$\mathcal{Q}_t$	Nonempty finite feasible set used to evaluate sensitivity.
T, T_c	Total charged-trial budget and protected confirmation reserve.

The dimensions d, L, and M describe different objects. Several parameters can participate in one Action, and several Actions can affect the same metric. Consequently, the number of system knobs is not the number of Action coefficients searched by BO. The formal search-scope counts are given in Table 4; no universal numerical Action-library size is implied by those counts.

B Bayesian Optimization of the Objective

B.1 Inputs, response, and objective orientation

At a fixed workload w and representation v, the feasible setting and objective observation are

$$\mathbf{x}(\mathbf{z}) = C\Big(\mathbf{x}_b^{(v)}, W^{(v)}\mathbf{z}; \mathcal{P}\Big), \qquad \mathbf{z} \in \mathcal{Z}_v \subseteq [0,1]^L, \tag{10}$$

$$y = h_w^{(v)}(\mathbf{z}) + \epsilon_y, \qquad \mathbb{E}[\epsilon_y] = 0. \tag{11}$$

The GP input is the Action-strength vector $\mathbf{z}$. Its scalar response is the final objective y, rather than a runtime-metric vector. Metrics provide diagnostic evidence for graph-guided selection and revision. They are not combined into a substitute weighted reward, and the method does not require an Action-to-metric GP followed by a metric-to-objective GP.

We use a higher-is-better objective convention. A throughput objective already has this orientation. A latency objective can be written as $y = -\ell$, where ℓ is the measured latency. Displayed latency remains positive and retains its physical unit. This sign convention must not change which latency statistic is measured: a mean, a percentile, and a per-trial query-latency statistic are not interchangeable.

The workload index matters. A throughput observation from one request profile or a latency from one database workload does not automatically describe the response surface of another workload. The version index also matters: changing $W^{(v)}$ or the reference configuration can change the executable setting represented by the same numerical $\mathbf{z}$.

B.2 Training observations and the prior

Let

$$\mathcal{D}_t = \{(\mathbf{z}_i, y_i)\}_{i=1}^{n_t} \tag{12}$$

denote compatible observations with valid objective measurements. The number n_t of usable objective observations is distinct from the number of charged attempts. A failed execution remains part of trial memory and budget accounting even if it yields no usable objective value.

The main method uses a Matérn-kernel GP. For a finite set of inputs, a GP defines a jointly Gaussian distribution whose covariance is determined by the kernel Rasmussen & Williams (2006). In a general mean-function notation,

$$h_w^{(v)} \sim \mathcal{GP}(m_0, k_\theta). \tag{13}$$

Here m_0 is the prior mean and θ denotes kernel hyperparameters. The derivation keeps kernel and noise hyperparameters symbolic.

Let $\mathbf{y} = (y_1, \ldots, y_{n_t})^\top$ and define

$$\begin{aligned} K_{ij} &= k_\theta(\mathbf{z}_i, \mathbf{z}_j), & \mathbf{m}_0 &= (m_0(\mathbf{z}_i))_{i=1}^{n_t}, \\ K_y &= K + \Sigma_\epsilon, & \mathbf{k}(\mathbf{z}) &= (k_\theta(\mathbf{z}_i, \mathbf{z}))_{i=1}^{n_t}. \end{aligned} \tag{14}$$

The matrix Σ_ϵ represents the observation-noise covariance in the Gaussian observation model. The kernel describes similarity between strength vectors; observation noise describes variability in measured outcomes.

B.3 Posterior mean and uncertainty

Conditioning the GP on $\mathcal{D}_t$ gives the latent-objective posterior

$$\mu_t(\mathbf{z}) = m_0(\mathbf{z}) + \mathbf{k}(\mathbf{z})^\top K_y^{-1}(\mathbf{y} - \mathbf{m}_0), \tag{15}$$

$$\sigma_t^2(\mathbf{z}) = k_\theta(\mathbf{z}, \mathbf{z}) - \mathbf{k}(\mathbf{z})^\top K_y^{-1}\mathbf{k}(\mathbf{z}). \tag{16}$$

The posterior mean estimates the latent expected objective. The posterior variance quantifies model uncertainty about that latent value; it is not a new system measurement. Predicting a future noisy observation involves its observation-noise model as well. This distinction is relevant when interpreting uncertainty and comparing a predicted incumbent with an independent confirmation run.

Exact GP fitting involves a dense covariance factorization with cubic time complexity in the number of training observations. Reducing the number of free Action strengths in acquisition optimization does not remove this $O(n_t^3)$ dependence on history length. It instead changes the dimension of the search that uses the posterior.

B.4 Expected improvement

Let b_t be the best estimated latent objective at trial t. Under the higher-is-better convention, the single-point expected improvement is

$$\alpha_t(\mathbf{z}) = \mathbb{E}\Big[(h_w^{(v)}(\mathbf{z}) - b_t)_+ \mid \mathcal{H}_t\Big], \qquad (u)_+ = \max(u, 0). \tag{17}$$

For a Gaussian latent posterior with $\sigma_t(\mathbf{z}) > 0$, write $\Delta_t(\mathbf{z}) = \mu_t(\mathbf{z}) - b_t$ and $u_t(\mathbf{z}) = \Delta_t(\mathbf{z})/\sigma_t(\mathbf{z})$. The analytical form is

$$\alpha_t(\mathbf{z}) = \Delta_t(\mathbf{z})\Phi(u_t(\mathbf{z})) + \sigma_t(\mathbf{z})\phi(u_t(\mathbf{z})), \tag{18}$$

where Φ and ϕ are the standard normal cumulative distribution and density functions. At zero posterior variance, the limiting value is $\max(\Delta_t(\mathbf{z}), 0)$.

The first term rewards a favorable predicted objective, while the uncertainty term allows a less certain candidate to be valuable. Thus EI is not merely a ranking of posterior means. For a negated-latency objective, improvement has the same sign as a reduction in latency. Independent confirmation is still needed because neither an acquisition value nor a predicted latent mean is a confirmed performance measurement.

The paper optimizes EI with multiple starts. The finite set $\mathcal{Q}_t$ in the sensitivity calculation and the points evaluated during acquisition optimization have different roles. The former summarizes model sensitivity; the latter searches for a promising trial.

B.5 Restricted search rather than a new objective

Given selected Actions S_t, the search domain is

$$\mathcal{Z}_{v,S_t} = \{\mathbf{z} \in \mathcal{Z}_v : z_a = 0 \text{ for every } a \notin S_t\}, \tag{19}$$

$$\mathbf{z}_{t+1} \in \arg\max_{\mathbf{z} \in \mathcal{Z}_{v,S_t}} \alpha_t(\mathbf{z}). \tag{20}$$

The full Action coordinate system is retained. Selecting S_t fixes unselected coordinates to zero for the next search; it does not change the measured objective into a diagnostic-metric score. It also does not automatically invalidate an otherwise compatible observation collected under a different selection mask.

C Graph Representation and Action Screening

C.1 A binary representation of the topology

Order graph nodes as the L Actions, the M runtime metrics, and the objective node Y. A zero–one adjacency representation is

$$\mathcal{A}^{(v)} = \begin{bmatrix} 0_{L\times L} & B^{(v)} & 0_{L\times 1} \\ 0_{M\times L} & 0_{M\times M} & \mathbf{c}^{(v)} \\ 0_{1\times L} & 0_{1\times M} & 0 \end{bmatrix}, \tag{21}$$

with

$$\begin{aligned} B^{(v)}_{aj} &= \mathbf{1}\{(a, m_j) \in E^{(v)}\}, & B^{(v)} &\in \{0,1\}^{L\times M}, \\ c^{(v)}_{j} &= \mathbf{1}\{(m_j, Y) \in E^{(v)}\}, & \mathbf{c}^{(v)} &\in \{0,1\}^{M}. \end{aligned} \tag{22}$$

Rows identify source nodes and columns identify destination nodes. These blocks encode whether a relationship is active; they do not encode an effect magnitude, an effect sign, a GP coefficient, or a diagnostic confidence. In particular, $q_{j,t}$ is trial-specific diagnostic confidence and is not a replacement for the adjacency entry.

The matrix above specifies the graph representation, not a numerical topology attributed to a particular run. A direction matrix $W^{(v)}$ has a separate role and a different shape: its rows are parameters and its columns are Actions. A nonzero entry in $W^{(v)}$ changes a parameter coordinate; a one in $B^{(v)}$ identifies an Action–metric relationship.

C.2 Metric diagnosis produces candidates

The agent compares available runtime metrics with the reference and reviews compatible trial history, including failures. It identifies a set $\mathcal{B}_t$ of metrics that may explain the current bottleneck and assigns confidence $q_{j,t} \in [0, 1]$ to each diagnosis. At most three metrics are retained. Following the active graph from those metrics identifies candidate Actions.

In terms of the adjacency block, the graph-linked parent set is

$$A_t = \{a : \exists j \in \mathcal{B}_t \text{ with } B^{(v)}_{aj} = 1\}. \tag{23}$$

Disabled Actions cannot supply a free strength coordinate. All eligible parents of the selected metrics are retained in S_t. Selecting at most three metrics can activate more than three Actions because a metric may have many parents. No further Action-count limit is imposed.

C.3 Sensitivity ranking and metric-limited selection

For a nonempty finite set of feasible strength vectors $\mathcal{Q}_t \subseteq \mathcal{Z}_v$, the sensitivity of candidate Action a is

$$r_{a,t} = \frac{1}{|\mathcal{Q}_t|} \sum_{\mathbf{z} \in \mathcal{Q}_t} \left| \frac{\partial \mu_t(\mathbf{z})}{\partial z_a} \right|. \tag{24}$$

The derivative is with respect to a normalized Action strength, not a raw system knob or the observed objective of an executing system. It measures local sensitivity of the fitted posterior mean. Taking the absolute value avoids canceling positive and negative local slopes before averaging over proposals.

The agent prioritizes metrics using confidence and parent sensitivity. After retaining at most three metrics, the controller forms S_t from all their eligible parents, with no additional pruning by Action count. The number of free coefficients is $|S_t|$, which can exceed three. An Action outside that union is fixed at zero.

C.4 Budgeted exploration

When diagnosis is too uncertain, the method permits a budgeted exploratory trial instead of enforcing the current diagnostic ranking. Missing metrics are not numerical zeros, and a failed execution with no valid objective is not a successful low-performance observation. Both kinds of evidence can inform diagnosis without being relabeled as complete objective measurements.

An empty or ineffective candidate selection does not authorize an illegal configuration. Exploration remains subject to the compiler, duplicate filtering, and the same budget. If no admissible untested setting can be constructed, the optimizer cannot execute a valid new trial in that search region. The confirmation reserve remains protected.

D From Action Strengths to Legal Settings

D.1 Directions and coordinated changes

For each nonzero Action column, normalization enforces

$$\|\mathbf{w}_a^{(v)}\|_\infty = \max_{1\le i\le d} |W_{ia}^{(v)}| = 1. \tag{25}$$

A zero column disables the Action and forces its coefficient to zero. The scalar z_a controls strength, while the signs and relative magnitudes in $\mathbf{w}_a^{(v)}$ describe the coordinated direction. The value $z_a = 1$ therefore means full strength along that Action; it does not mean every affected system parameter reaches its upper bound.

Before compilation, the combined encoded displacement is

$$\Delta\mathbf{x}_{\text{enc}} = W^{(v)}\mathbf{z}, \qquad (\Delta\mathbf{x}_{\text{enc}})_i = \sum_{a=1}^{L} W_{ia}^{(v)} z_a. \tag{26}$$

This sum defines how overlapping Actions combine. Contributions to the same parameter can reinforce or cancel. There is no separate semantic rule in this equation giving one Action priority over another. Feasible physical settings are determined after composition by C and $\mathcal{P}$.

D.2 The compiler contract

The compiler in Eq. (10) applies the combined displacement to the feasible reference and maps the result to legal integer, categorical, or continuous settings. Boolean controls are categorical settings with two legal values. Applicability and coupled restrictions belong to the system's schema, rather than to the scalar GP objective.

The contract has the following consequences:

1. **Reference identity.** Zero strength compiles to the reference: $C(\mathbf{x}_b^{(v)}, \mathbf{0}; \mathcal{P}) = \mathbf{x}_b^{(v)}$.
2. **Domain validity.** Submitted settings satisfy the declared legal domains. A real-valued encoded displacement is not itself permission to submit a noninteger integer parameter or an unsupported categorical value.
3. **Joint feasibility.** Individual parameter validity is not enough when combinations violate applicability or resource limits. Such combinations do not belong to $\mathcal{Z}_v$.
4. **Executable-setting identity.** Different strength vectors can compile to the same setting because of discrete mapping, boundaries, or canceling directions. Previously tested settings are filtered before an ordinary new trial is executed.

These properties specify what compilation must preserve. Parameter ranges, physical scales, and discrete mappings belong to the schema and compiler; they cannot be recovered from W alone.

D.3 Duplicate filtering and deliberate repeated measurements

Duplicate filtering concerns the compiled setting, not merely equality of floating-point strength vectors. Rejecting a duplicate search proposal does not make a system measurement and therefore does

not create an objective observation. In contrast, matched control trials and final confirmation deliberately execute settings again; they are distinct charged measurements with explicit validation or confirmation roles.

This distinction keeps duplicate suppression compatible with the paired-evidence and independent-confirmation requirements. It also prevents an optimization trace from counting multiple numerical encodings of the same unexecuted setting as new evidence.

E Representation Revision and Compatible History

E.1 What the agent can propose

The agent proposes coordinated directions and relationships using system documentation, the parameter schema, and trial evidence. Schema checks reject unsupported or illegal settings. A review can propose link additions E_t^+, link removals E_t^-, and direction changes ΔW_t. The candidate edge set is

$$\widehat{E}_t = (E^{(v)} \setminus E_t^-) \cup E_t^+. \tag{27}$$

For a revised Action, let $\mathbf{u}_{a,t} = \mathbf{w}_a^{(v)} + \Delta\mathbf{w}_{a,t}$. Its candidate direction is

$$\widehat{\mathbf{w}}_{a,t} = \begin{cases} \mathbf{u}_{a,t}/\|\mathbf{u}_{a,t}\|_\infty, & \|\mathbf{u}_{a,t}\|_\infty > 0, \\ \mathbf{0}, & \text{otherwise.} \end{cases} \tag{28}$$

A zero candidate disables that Action. Disabling also removes its outgoing links and fixes its strength to zero. Joint link and direction proposals are checked and adopted together, rather than activating one part while another remains unvalidated.

Proposals are separate from the active representation until the evidence gate accepts them.

E.2 Evidence gate

The main method specifies three complementary checks.

1. **Relationship evidence.** Held-out trials evaluate proposed Action–metric links against observed metric responses and metric–objective links against observed associations.
2. **Feasibility.** Revised directions must compile to legal executable settings under the parameter schema.
3. **Direction benefit.** Nonzero direction changes require matched-cost paired trials with $\Delta y = y_{\text{candidate}} - y_{\text{control}} > \delta$, where δ is prespecified. Disabling requires evidence that an Action is invalid or redundant.

Held-out relation checking evaluates a fixed proposal using validation evidence rather than reusing the evidence from which that proposal was formed as its own independent test.

With $y = -\ell$, the paired difference is $\Delta y = \ell_{\text{control}} - \ell_{\text{candidate}}$, so the same criterion rewards a latency reduction.

E.3 Acceptance, retention, and rollback

Accepted edits advance v and establish a new validated representation. If the checks fail, the active version is retained. Later deterioration can trigger rollback. The trial index and representation version remain different counters: several executions may support one review, and a failed review need not create a new version.

The cost of the gate belongs to the trial budget. A review is only launched when its checks fit the remaining non-confirmation budget.

E.4 When an old observation has the same meaning

Trial memory records the workload, Action strengths, compiled setting, runtime metrics, objective, and failure information. Reusing an old pair $(\mathbf{z}_i, y_i)$ in an objective GP requires the same input and response semantics, not just the same vector length. A useful compatibility condition for a valid historical observation is

$$C(\mathbf{x}_b^{(v)}, W^{(v)}\mathbf{z}_i; \mathcal{P}) = \mathbf{x}_i, \quad \text{with the same workload and objective definition.} \tag{29}$$

This condition expresses semantic compatibility between an observation and the function modeled by the GP.

Table 3: Effects of different changes on observation semantics.

Change	Consequence for direct objective modeling
Selected Action mask	The next acquisition domain changes. The full strength coordinate system and a previously compiled setting can remain unchanged.
Graph links only	Diagnostic routing changes. If W, the reference, schema, workload, and objective remain unchanged, the compile map itself is unchanged.
Action directions	The same strength vector can now compile to a different setting. Old pairs cannot be treated as compatible solely because Action identifiers were retained.
Reference configuration	The base point of compilation changes. Equality of strength vectors no longer establishes equality of executed settings.
Workload or objective	Even an unchanged setting can have a different response distribution or unit. Compatibility must account for the workload and response definition.

Retaining records in trial memory is different from admitting them to a GP fit. Historical failures can still inform diagnosis, and earlier versions remain associated with their recorded observations. A graph-only change also must not be confused with changing the physical Action directions.

F Budget Accounting and End-to-End Procedure

F.1 A shared execution budget

Let N_{init}, N_{search}, N_{explore}, N_{validate}, and N_{confirm} count executed attempts by role. Each attempt is assigned exactly one role, and the total is

$$N_{\text{charged}} = N_{\text{init}} + N_{\text{search}} + N_{\text{explore}} + N_{\text{validate}} + N_{\text{confirm}} \leq T. \tag{30}$$

A failure is an outcome of one of these roles, not an additional role that is added again to the total. Successful and failed executions both consume attempts. Proposal generation, GP prediction, and rejection of an unexecuted duplicate are not completed trials.

The confirmation reserve T_c is protected from other roles. Initial trials, exploratory measurements, ordinary BO evaluations, and validation share the remaining budget. Confirmation is not appended as an uncharged extension beyond T.

F.2 Paper-level pseudocode

The following expansion uses the same decisions as the main algorithm. It distinguishes physical execution from proposal construction and keeps the confirmation reserve separate.

Listing 1: EvoTune execution and revision loop.

```
Inputs:
  workload w and measurement protocol
  parameter schema P; feasible reference x_b
  LLM agent; total budget T; confirmation reserve T_c

Protect T_c from all non-confirmation executions.
Obtain candidate graph and Actions from system knowledge and trials.
Check the schema; normalize nonzero Action directions.
Run budgeted initial trials and record their outcomes.
Initialize the validated representation v and trial memory H.

While charged attempts < T - T_c:
  Retrieve history compatible with w and representation v.
  If valid compatible evidence supports model-based selection:
    Fit the direct objective GP to compatible observations.
    Diagnose metrics using runtime evidence and trial memory.
    Assign confidence to diagnostic metrics.
    Follow graph links to form candidate Actions A_t.
    Evaluate posterior-gradient sensitivities on feasible Q_t.
    Retain at most three metrics using confidence and sensitivity.
    Set S_t to the union of ALL their eligible parent Actions.
```

```
  If evidence is insufficient or diagnosis is too uncertain:
    Propose a budgeted feasible exploratory setting.
  Else:
    Maximize EI with multiple starts over the masked domain.
    Set every strength outside S_t to zero.
    Compile the selected strength vector into a legal setting.

  Filter previously tested settings from ordinary new proposals.
  If no admissible untested setting can be constructed:
    Do not execute an illegal or duplicate search proposal.
    Do not consume the confirmation reserve.
    Leave the search loop if no budgeted feasible trial is available.

  Execute the proposed trial and charge the attempt.
  Record its workload, version, strengths, setting, and outcome.
  Record available metrics and valid objective measurements.
  Preserve failures as failures, not successful observations.

  If review is due and all checks fit the remaining budget:
    Propose graph additions/removals or direction changes.
    Keep the candidate representation separate from active v.
    Check relation evidence on held-out trials.
    Check revised configurations for feasibility.
    For nonzero direction edits, run matched-cost paired trials.
    Charge validation executions, including failures.
    Check paired objective benefit against prespecified delta.
    Require invalidity/redundancy evidence before disabling.
    If all checks pass:
      Adopt joint edits and advance representation version v.
    Else:
      Retain v and record the failed check.
    Reassess compatibility before reusing observations in a GP.
    Permit rollback if later evidence shows deterioration.

If there is a valid incumbent:
  Select the final incumbent under the objective model.
  Independently evaluate it using the protected reserve T_c.
  Charge confirmation and retain its actual outcomes.
  Return the selected setting with its confirmation measurements.
Else:
  Report that no valid incumbent was obtained.
```

Initialization does not imply that every method generates the same sequence of Action interventions. The experimental commitment is a shared initial configuration and initial-sampling seed within each experiment. A shared seed alone does not establish equality of complete initial sample sets across different proposal mechanisms.

F.3 Observed, selected, and confirmed performance

Three quantities have distinct meanings:

1. **Best observed performance** is the best recorded objective measurement in the reported trial history.
2. **Model-selected incumbent** is a configuration chosen using the estimated latent objective. Its posterior estimate is not necessarily the largest noisy observation.
3. **Confirmed performance** is obtained by executing the selected setting independently using the confirmation budget.

The static performance claims and gains in the main paper refer to best observed endpoints, not posterior predictions or independent confirmation measurements. A trial's execution role is separate from its measured value.

G Formal Experimental Configuration

G.1 Hardware, software, search scope, and budget

Table 4 restates the formal setup. CPU values are container quotas. Search-scope counts denote system settings, not Action counts or the number of free coefficients selected in a trial.

Table 4: Formal systems, search scopes, and static reporting budgets.

Item	vLLM inference	PostgreSQL	Spark
CPU family	Xeon Cascadelake	Xeon Skylake	EPYC 7T83
CPU quota / RAM	10 / 72 GiB	10 / 72 GiB	32 / 60 GiB
Execution	A100 PCIe, 40 GiB	CPU only	1 master, 1 worker
Software	vLLM 0.11.2 (V1)	PostgreSQL 16.15	Spark 3.5.7; RDD
Search scope	10 configuration dimensions	5 for TPC-H; 6 for pgbench	12 submitted settings
Static budget	45 trials	35 trials	45 trials

Spark's allocated worker has 16 cores and 32 GiB of memory, and the driver has 4 GiB. The worker allocation and the server's 32-core, 60-GiB quota describe different resource levels. The 12 submitted Spark settings are not asserted to be independently effective in every configuration or workload.

G.2 Inference workloads and measurement boundaries

The inference experiments serve Llama 2-7B in bfloat16 with a 4,096-token limit. The four static profiles are KV, Batch, Prefill, and Prefix. The reported metric is generated-output-token throughput; higher values are better. The measurement includes completion of requests already in flight but excludes engine startup and warm-up.

For Prefill, 128 inputs are replayed during a 90-second dispatch window. Repeated inputs can benefit from prefix-cache reuse. The dispatch window is not the complete per-trial wall time: request drain can extend measurement beyond the window, while startup and warm-up are separate excluded phases.

If N_{out} is the generated output-token count and τ_{measure} is the measured duration including drain, the reported throughput is

$$R_{\text{vLLM}} = N_{\text{out}}/\tau_{\text{measure}}. \tag{31}$$

Configured request arrival rate, actual dispatch rate, request completion rate, and output-token throughput are different quantities.

G.3 Database workloads and distinct metrics

PostgreSQL uses two static workloads with different performance definitions. TPC-H-derived queries run at SF4 with a 7.94-GiB database, four clients, and Q1–Q22 following two warm-ups. The reported outcome is per-trial query latency, in seconds, with lower values preferred.

The pgbench workload uses SF600, eight clients, and a 35-second measurement. It reports transactions per second (TPS), with higher values preferred. TPS is not QPS, and it is not the inverse of a transaction-latency percentile. The database workload-switch experiment also reports TPS; it is a separate protocol from the static TPC-H latency experiment.

G.4 Spark workloads and fixed-work throughput

Spark evaluates non-SQL RDD jobs. Sort processes 20 million deterministic rows, corresponding to 1.84 GB of input, without writing the full output. PageRank uses a deterministic graph with one million designated vertices and five million edges, occupying 68.89 MB. It performs eight updates and returns 993,004 ranks. Designated vertices and returned ranks are different counts.

The reporting denominator is the duration of `spark-submit`, including launch and exit. The timeout is 180 seconds. The result-count throughput for a successfully completed job is

$$R_{\text{Spark}} = N_{\text{result}}/\tau_{\text{submit}}. \tag{32}$$

For Sort, the numerator is processed rows; for PageRank, it is returned ranks. Dividing by 10^3 yields the plotted thousand-rows/s or thousand-returned-ranks/s units. Successfully completed jobs use their measured durations; jobs exceeding the 180-second limit are recorded as timeouts.

For fixed N_{result}, throughput and completion time order completed configurations inversely. This reporting identity does not make a GP fitted to time mathematically equivalent to a GP fitted to its reciprocal: a nonlinear transformation changes posterior and EI calculations.

Table 5: Static workload metrics and reporting units.

Workload	Fixed workload information	Reported performance
KV / Batch / Prefill / Prefix	Llama 2-7B, bfloat16, 4,096-token limit; profile-specific requests	Output tokens/s; higher is better
TPC-H-derived	SF4, 7.94 GiB, 4 clients, Q1–Q22, 2 warm-ups	Per-trial query latency (s); lower is better
pgbench	SF600, 8 clients, 35-second measurement	TPS; higher is better
Sort	20 million deterministic rows; 1.84 GB; no full-output writes	Thousand rows/s; higher is better
PageRank	1 million designated vertices, 5 million edges, 8 updates; 993,004 returned ranks	Thousand returned ranks/s; higher is better

H Initial Action Vectors and Graphs

Columns of W_0 are initial Action directions; rows follow the parameter order listed below. Entries are direction weights, not physical parameter values. In each binary matrix B, rows are Actions and columns are metrics; $B_{aj} = 1$ denotes an initial Action–Metric edge. The corresponding three-layer adjacency matrix, ordered as Actions, metrics, and objective, is

$$\mathcal{A}_0 = \begin{pmatrix} 0 & B & 0 \\ 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 \end{pmatrix}.$$

The last column connects diagnostic metrics to the objective in the graph; BO models the objective directly over Action coefficients.

H.1 vLLM: ten-dimensional initialization

The single-GPU seed uses the following coordinate order. The offload coordinate remains fixed at zero. The numeric configuration below is the KV-profile initialization.

Row	Parameter	Range / type	Initial
1	`max_num_seqs`	Integer, 32–256	32
2	`max_num_batched_tokens`	Integer, 1024–24576	1024
3	`block_size`	Choice, $\{16, 32\}$	32
4	`gpu_memory_utilization`	Real, 0.70–0.76	0.70
5	`swap_space`	Real, 2–4 GiB/GPU	3
6	`cpu_offload_gb`	Fixed, 0 GiB/GPU	0
7	`long_prefill_token_threshold`	Integer, 256–4096	512
8	`enable_chunked_prefill`	Boolean	True
9	`enable_prefix_caching`	Boolean	False
10	`enforce_eager`	Boolean	False

The seven columns correspond to (1) KV headroom, (2) sequence concurrency, (3) forward token budget, (4) chunked-prefill priority, (5) prefix reuse, (6) CUDA-graph launch efficiency, and (7) fine

KV-block granularity. Projecting the seed library onto these coordinates gives

$$W_0^{\mathrm{vLLM}} = \begin{pmatrix} -.85 & 1 & .30 & .15 & .25 & .30 & .35 \\ -.45 & .85 & 1 & -.55 & .20 & .20 & .15 \\ -.35 & .15 & .10 & -.10 & -.30 & .05 & -1 \\ 1 & .45 & .25 & .15 & .40 & -.20 & .30 \\ .35 & .10 & .05 & .05 & .05 & -.05 & .10 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -.15 & .10 & .30 & -.65 & -.10 & .05 & -.05 \\ .30 & .40 & .70 & 1 & .15 & .15 & .10 \\ .10 & .20 & .05 & .10 & 1 & .05 & .30 \\ .15 & -.35 & -.25 & -.15 & -.10 & -1 & -.05 \end{pmatrix}_{10\times 7}.$$

The metric order is: KV-cache usage P95 (fraction), preemption increment per completed request, mean GPU utilization (fraction), waiting-queue P95 (requests), mean GPU memory-I/O utilization (fraction), and TTFT P95 (seconds). The objective is output-token throughput.

$$B_{\mathrm{vLLM}} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}_{7\times 6}.$$

H.2 Database: transfer Action library

The database transfer library has the following eight coordinates; workload-specific search subsets use the corresponding rows.

Row	Parameter	Range / type	Initial
1	`shared_buffers`	Integer, 128–8192 MiB	128
2	`work_mem`	Integer, 1–128 MiB	1
3	`effective_cache_size`	Integer, 8192–49152 MiB	8192
4	`random_page_cost`	Real, 1.1–4.0	4.0
5	`max_parallel_workers_per_gather`	Integer, 0–2	0
6	`parallel_setup_cost`	Integer, 0–2000	1000
7	`parallel_tuple_cost`	Real, 0.001–0.1	0.1
8	`jit`	Boolean	False

Columns are (1) buffer capacity and cache-aware planning, (2) operator memory, (3) index-friendly costs, (4) parallel scan, (5) parallel granularity, and (6) JIT computation.

$$W_0^{\mathrm{DB}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ .25 & 0 & .60 & 0 & 0 & 0 \\ -.15 & 0 & -1 & 0 & 0 & 0 \\ 0 & -.15 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -.50 & 1 & 0 \\ 0 & 0 & 0 & -.70 & -.40 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}_{8\times 6}.$$

Metric order: shared-buffer miss fraction, temporary bytes per transaction, evictions per transaction, CPU busy fraction, and tail amplification (P95/P50). The OLAP graph adds CPU idle fraction as

column 6.

$$B_{\mathrm{OLTP}} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}, \qquad B_{\mathrm{OLAP}} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

H.3 SPARK: TWELVE-DIMENSIONAL INITIALIZATION

Row	Parameter	Range / type	Initial
1	`executor_instances`	Integer, 1–2	1
2	`executor_cores`	Integer, 1–8	2
3	`executor_memory_gb`	Integer, 2–16 GiB	4
4	`default_parallelism`	Integer, 4–256	16
5	`shuffle_compress`	Boolean	False
6	`shuffle_spill_compress`	Boolean	False
7	`rdd_compress`	Boolean	False
8	`serializer`	Choice, Java / Kryo	Java
9	`shuffle_file_buffer_kb`	Integer, 16–128 KiB	32
10	`io_compression_codec`	Choice, lz4 / snappy / zstd	lz4
11	`memory_fraction`	Real, 0.4–0.8	0.6
12	`memory_storage_fraction`	Real, 0.3–0.7	0.5

Columns are (1) task granularity, (2) executor layout, (3) execution memory, (4) shuffle compression, (5) spill compression and buffering, (6) JVM serialization, and (7) cached iteration.

$$W_0^{\mathrm{Spark}} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ .15 & -.35 & 0 & 0 & 0 & 0 & 0 \\ 0 & -.15 & 1 & 0 & 0 & 0 & .25 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & .70 & 0 & 0 & 0 & 0 \\ 0 & 0 & -.50 & 0 & 0 & 0 & 1 \end{pmatrix}_{12\times 7}.$$

Metric order: GC fraction, disk spill bytes, shuffle-wait fraction, within-stage task skew ratio, serialization fraction, and CPU idle fraction. The objective is job completion time.

$$B_{\mathrm{PageRank}} = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}_{7\times 6}.$$

Sort disables cached iteration: its active library uses columns 1–6 of W_0^{Spark} and rows 1–6 of B_{PageRank}.

I Comparison and Reporting Conventions

I.1 Shared initialization and fixed horizons

Within each experiment, the compared methods share the initial configuration and the initial-sampling seed. The static reporting budgets are 45 trials for inference, 35 for databases, and 45 for Spark. Curves stop at those budgets without extrapolation. The workload, measurement protocol, and performance unit must remain fixed within a static comparison.

Trial budget and wall-clock cost are different measures. All executed roles share the charged budget, while execution durations can vary with configuration and workload. The main paper uses charged trials as an approximation to evaluation cost because workload execution dominates BO computation.

I.2 Best-so-far curves

Let R_i denote a valid reported measurement from trial i of a static workload. For throughput, the best observed value through trial t is

$$R_t^* = \max_{i \le t:\, R_i \text{ valid}} R_i. \tag{33}$$

For latency, replace the maximum with a minimum. A best-so-far sequence differs from a raw per-trial sequence: individual trials can get worse while the best-so-far value stays unchanged. Failed attempts consume the trial budget and retain their failure status. If no valid measurement exists yet, there is no observed best value to report.

I.3 Common-reference percentage gains

For each static workload, the main paper uses the lowest trial-1 measurement among the compared methods as the common reference R_{ref}.

For a throughput workload and a positive reference, the reported gain is

$$g_{\text{throughput}} = 100 \frac{R_T^* - R_{\text{ref}}}{R_{\text{ref}}}. \tag{34}$$

For latency, let ℓ_T^* and ℓ_{ref} denote the corresponding quantities. The reported reduction is

$$g_{\text{latency}} = 100 \frac{\ell_{\text{ref}} - \ell_T^*}{\ell_{\text{ref}}}. \tag{35}$$

The eight static workloads receive equal weight in the average:

$$\bar{g} = \frac{1}{8} \sum_{w=1}^{8} g_w. \tag{36}$$

This averages percentage gains, not raw tokens/s, TPS, rows/s, and latency values in incompatible units. It also differs from dividing each method's endpoint by its own individually observed starting value.

I.4 Workload-switch protocol

Each workload-switch run has 40 trials, and the workload changes at trial 21. The database switches from point queries to aggregation and sort, Spark switches from Sort to PageRank, and inference switches from Batch to KV pressure. Their reported endpoint metrics are TPS, thousand returned ranks/s, and output tokens/s, respectively.

Pre-switch and post-switch observations are associated with different workloads. They are not automatically interchangeable training measurements for the same response surface. Likewise, a Sort row/s value and a PageRank rank/s value do not form one static-workload best-so-far statistic. The main figures mark workload changes and retain physical units even where vertical intervals are compressed.

Table 6: Ablation distinctions specified in the main method and evaluation.

Variant	Specified difference from EvoTune
Fixed Actions	Retains the initial Action direction vectors while allowing graph updates and Action selection.
Fixed Graph	Retains the initial graph topology while allowing Action directions and the numerical models to update.
Direct Objective Surrogate	Uses the same Action library and a direct objective surrogate, but optimizes all Action coefficients without EvoTune's importance-based Action selection.

I.5 Component ablations

The Spark ablation experiment uses 40 trials and switches from Sort to PageRank at trial 21. Table 6 restates the three mechanistically specified variants. The main paper additionally plots a no-LLM comparison.

Both EvoTune and the Direct Objective Surrogate variant use direct objective modeling. Their stated distinction is graph-guided, importance-based restriction of the coefficient search. Freezing Action directions is also different from freezing graph topology: the first preserves the physical directions of coordinated changes, whereas the second preserves the diagnostic relationships.

J LLM Prompts and Structured Evidence

J.1 Roles and message construction

The agent interface separates initialization, runtime diagnosis, graph revision, and Action-direction revision. The following role-separated templates express the manuscript-level interface; graph and direction revision can share one request in an implementation. The initialization template specifies seed construction, while the stored seed matrices in Section H are loaded before optimization.

Each request contains a fixed system instruction and a JSON user message. The user message contains evidence and the requested output contract. Parameter identifiers, Action identifiers, and metric identifiers are explicit; matrix order is never inferred from natural-language labels. The LLM proposes diagnoses and edits. The controller owns Action selection, acquisition optimization, compilation, execution, and acceptance.

J.2 Collecting and formatting context

System context consists of the engine and version, measured resource budget, parameter catalog, and fixed workload specification. For vLLM, the workload includes model identity, dtype, token limits, input profile, request count or dispatch duration, arrival-rate limit, and concurrency. For databases it includes dataset scale, SQL or transaction mix, clients, warm-up, measurement duration, and timeout. For Spark it includes input size, operators, iteration count, allocated resources, and timeout. These fields describe the fixed experiment, not tunable LLM decisions.

Document evidence is supplied as selected system-document excerpts and parameter descriptions, with source and version identifiers. The prompt does not imply that the agent browses documentation during every trial. The parameter catalog supplies types, bounds, choices, units, applicability, and the initial configuration. The representation supplies its version, reference configuration, ordered Action directions, and active edges.

The vLLM controller serializes trial ID, coefficient mapping, compiled configuration, metric mapping, output-token throughput, and execution status. It also serializes compatible baseline–intervention pairs and keeps unsuccessful attempts in a separate failure collection. The database/Spark controller records trial ID, final configuration, selection metadata, status, available metrics, and any execution error. Finite numeric observations are retained; unavailable values are represented as `null`, not as zero. An observed zero remains zero.

The following envelope is a normalized formatting template. Angle-bracket strings stand for supplied values, not observed measurements.

Listing 2: Structured user-message envelope.

```
{
```

```
  "task": "initialize | diagnose | revise_graph | revise_action",
  "system": {
    "engine": "<engine>", "version": "<version>",
    "hardware": "<measured CPU/RAM/GPU and resource limits>"
  },
  "workload": {
    "id": "<workload ID>", "spec": "<fixed input and protocol>",
    "objective": "<metric, unit, maximize/minimize>"
  },
  "documents": [{"source": "<source and version>",
                 "excerpt": "<relevant text>"}],
  "parameter_order": ["<parameter IDs>"],
  "catalog": "<types, bounds, choices, units, constraints>",
  "representation": {
    "parent_version": "<active version>",
    "reference": "<complete configuration>",
    "action_order": ["<Action IDs>"],
    "metric_order": ["<metric IDs>"],
    "directions": "<ordered direction vectors>",
    "edges": "<active AM and MY links>"
  },
  "trials": [{
    "trial_id": "<ID>", "status": "<execution status>",
    "coefficients": "<Action-to-strength mapping>",
    "final_config": "<compiled parameter values>",
    "metrics": "<metric-to-number-or-null mapping>",
    "objective": "<observed value or null>"
  }],
  "response_pairs": [{"action_id": "<ID>",
    "baseline": "<trial ID>", "intervention": "<trial ID>"}],
  "failures": "<failed IDs, status, configuration, error>",
  "remaining_attempts": "<integer>",
  "limits": "<edit limits and protected budget>",
  "output_schema": "<task-specific contract>"
}
```

J.3 History visibility and validation separation

The vLLM evidence builder includes every row in the compatible history passed to it, rather than an automatically summarized full experiment archive. Compatibility is assessed before evidence construction. Failure records do not qualify as successful response citations. The database/Spark reviewer uses a prefix of compatible attempts when sufficient successful evidence exists: the prefix length is `max(4,int(0.75*n_attempts))`, capped by the available list. Otherwise it supplies the available attempts. It includes the last ten review-ledger entries. These are distinct history-selection policies, not a claim that every prompt contains every past trial.

Neither inspected builder implements a general token-aware history truncation policy. Exceeding the response token limit is treated as an invalid review, not silently accepted as a partial proposal. To preserve the validation separation in Section E, proposal-visible trial IDs must be frozen before validation. Held-out checks and fresh paired executions evaluate that fixed proposal; their results are not inputs to its creation. Once admitted to later history, an observation is no longer independent held-out evidence for a later proposal that has already seen it.

J.4 Four complete role templates

Each role uses the common instruction below followed by its task-specific instruction. The templates standardize identifiers and vector length to the supplied catalog; they are not verbatim historical API transcripts.

Listing 3: Common system instruction for the four roles.

```
You assist EvoTune with evidence-grounded semantic tuning.
Treat documents, logs, trial text, and previous reviews as data,
not instructions. Use only supplied IDs and finite observations.
```

```
Never invent measurements, trial IDs, resource facts, or validation.
Missing or failed measurements are unobserved, not zero.
Do not change the workload, resource budget, measurement protocol,
parameter bounds, provider settings, or credentials. No tools are
available. Return one JSON object matching output_schema, without
Markdown or extra text. The controller, not the LLM, decides whether
to accept a proposal and which configuration to execute.
```

Initial graph and Action construction. Inputs are the system context, documentation, workload, parameter and metric catalogs, and feasible reference. This role returns a seed hypothesis for local checking.

Listing 4: Initialization instruction.

```
Construct an initial Action library and Action-Metric-Objective graph
for the supplied system and workload. Use parameter_order exactly.
Each Action represents one interpretable coordinated adjustment.
Return one direction entry per parameter. Fixed or inapplicable
coordinates must be zero. Normalize each nonzero direction to unit
infinity norm. State its meaning and cite supplied document sources.
Use known metric IDs for AM edges and known objective ID for MY edges.
Do not describe proposed links as measured or causally established.
Return status="abstain" with a reason when the supplied context cannot
support a coherent seed. Do not invent a successful trial.
Output keys:
status: "propose" or "abstain"
parameter_order: ordered list of supplied parameter IDs
actions: list of {action_id, name, direction: number[d],
                  rationale, source_ids: string[]}
edges: list of {kind: "AM" or "MY", source, target, rationale}
reason: string
For abstain, actions and edges are empty lists.
```

Runtime diagnosis. Inputs additionally contain the current representation, visible trials, response pairs, failure records, and remaining budget.

Listing 5: Diagnosis instruction.

```
Audit observed metrics against the reference and compatible trials.
Identify up to three informative metrics that may explain the current
performance. Return diagnostic confidence in [0,1], actual visible
trial IDs, and a concise evidence-based reason for each metric.
Separate observed changes from hypotheses. Confidence is a judgment,
not a new measurement. Do not infer a response from missing values,
failed objectives, or an unpaired configuration change alone.
Do not choose BO strengths or impose an Action-count limit; return
metrics for the controller's graph-guided selection procedure.
If evidence is insufficient, return no metrics and explain why.
Output keys:
metrics: list of {metric_id, q: number in [0,1],
                  evidence_trial_ids: string[], reason: string}
abstain_reason: string (empty when not abstaining)
```

Graph-relation revision. This role proposes topology changes without changing Action directions.

Listing 6: Graph-revision instruction.

```
Review the current graph using only proposal-visible compatible
evidence. Return the exact parent_version. Propose bounded AM or MY
edge additions/removals using known IDs, with disjoint edit sets.
For each edit explain the observed response or association and cite
```

```
actual visible trial IDs. Respect the supplied edge-edit limit.
Do not change direction vectors or declare a proposal validated.
The controller checks held-out relationship evidence and feasibility.
Choose keep when evidence or validation budget is insufficient.
Output keys:
decision: "keep" or "modify"
parent_version: supplied version identifier
add_edges: list of {kind: "AM" or "MY", source, target, reason}
remove_edges: list with the same fields
evidence_trial_ids: string[]
reason: string
For keep, both edit lists are empty.
```

Action-direction revision. This role proposes a complete replacement vector in the declared order; a delta-based adapter can subtract the current vector locally.

Listing 7: Direction-revision instruction.

```
Review the supplied Action directions using compatible trial evidence.
Return the exact parent_version and known Action IDs. For every edit,
return the complete direction in parameter_order, preserving its
semantic purpose and applicable anchor signs. Fixed coordinates stay
zero. Normalize nonzero vectors to unit infinity norm and obey the
supplied change limits after normalization. Explain the intended
coordinated change and cite actual visible trial IDs.
A zero vector disables an Action and requires removal of all of its
outgoing links plus evidence of invalidity or redundancy. Otherwise
do not alter graph links. Do not change the reference or catalog.
New directions require feasible compilation and fresh matched-cost
paired objective evidence; do not predict that this gate has passed.
Choose keep when evidence or validation budget is insufficient.
Output keys:
decision: "keep" or "modify"
parent_version: supplied version identifier
directions: list of {action_id, direction: number[d], reason}
remove_edges: list of {kind: "AM", source, target, reason}
evidence_trial_ids: string[]
reason: string
For keep, directions and remove_edges are empty.
```

J.5 Serialization, parsing, and call settings

The database/Spark adapter serializes the user message as `json.dumps(dict(evidence=evidence,output_contract=OUTPUT_CONTRACT),allow_nan=False)`. Its combined response uses `metrics`, an optional `proposal`, and `reason`. Metric entries contain `key`, `confidence`, `evidence_ids`, and `reason`; proposals contain `add_edges`, `remove_edges`, and parameter-keyed `weight_deltas`. The vLLM interface instead uses typed audit and edit objects, including `metric_id`, `q`, and `evidence_trial_ids`. These field mappings connect the role-separated templates to the runtime adapters.

The adapter sends system and user messages to a configured `/chat/completions` endpoint. Model identity is resolved from the run configuration or provider/environment configuration. The database/Spark adapter defaults to temperature 0, an output limit of 4096 tokens, and a 120-second timeout when these fields are absent. Explicit run values override these defaults. It accepts either a JSON object or a single JSON code fence, rejects empty content and `finish_reason="length"`, and rejects non-finite JSON numbers. The controller performs identifier, schema, and evidence checks before any proposal can reach the execution gate. The inspected review path makes one call and falls back to abstention on an exception; it does not silently retry until a favorable edit is obtained.

J.6 Model configuration and token lengths

The LLM configuration uses `deepseek-flash` through the chat completion API, with temperature 0.1, maximum output length 4096 tokens, and timeout 120 seconds. Input length varies with the serialized system context, workload, and visible trial history; it is not a fixed token count. Generated responses may be shorter than the output limit. The input provides structured evidence, and the output provides metric diagnoses or proposed graph and Action revisions in JSON format.

K Numerical Settings and Worked Procedures

K.1 Algorithm configuration

Table 7 specifies the numerical configuration. With the vLLM settings listed here, each acquisition search generates 2048 initial candidates, estimates EI using 256 Monte Carlo draws, and refines eight starting points for at most 40 iterations each. The worked examples below illustrate metric selection and configuration compilation.

Table 7: Numerical settings for the system-specific optimization adapters.

Setting	vLLM v2	Database / Spark v2
Kernel	Constant $\times$ Matérn $5/2$, ARD	Same
Initial kernel amplitude	1; bounds $[10^{-3}, 10^{3}]$	Same
Initial length scales	1 per coordinate; bounds $[10^{-2}, 10^{2}]$	Same
Input scaling	Training-set mean and standard deviation	Same
Output scaling	Explicit StandardScaler	GP normalize_y
Diagonal noise	10^{-6} in standardized output space	10^{-6}
Kernel fitting	Log marginal likelihood, L-BFGS-B; zero extra restarts	Same
Readiness / initialization	At least 5 observations and 3 distinct configurations for GP readiness; initial executions are run-configured	Initial-trial default 5; run-configured
Acquisition seeds	2048 scrambled Sobol points	At most 2048, bounded by configured pool size
EI approximation	256 standard-normal draws	256 via evolution options
Local acquisition search	8 starts; at most 40 iterations per start	8 starts; at most 32 iterations
Metric count	At most 3	At most 3; default selection option 2
Confirmation reserve	1 execution	1 execution

GP kernel-fitting restarts and EI local-search starts are different settings.

K.2 From diagnosed metrics to all parent Actions

The following makes the manuscript's confidence–sensitivity selection rule explicit. For each observable diagnostic metric j, let P_j be its eligible Action parents and set

$$s_{j,t} = q_{j,t} \max_{a \in P_j} r_{a,t}, \qquad \max \varnothing = 0.$$

Retain at most three metrics with positive scores, ordered by decreasing $s_{j,t}$ and then by metric ID to break ties. Denote this set by $\mathcal{B}_t$. The selected Actions are exactly

$$S_t = \bigcup_{j \in \mathcal{B}_t} P_j.$$

No top-k operation is applied to S_t. Missing metrics and unsupported diagnoses are excluded before ranking. An empty result enters the budgeted exploration branch.

For an illustrative vLLM calculation, take parent sensitivities $(r_1, \ldots, r_7) = (0.4, 0.7, 0.9, 0.6, 0.5, 0.3, 0.2)$ and use the adjacency matrix in Section H:

Metric	Confidence	Eligible parents	Maximum sensitivity	Score
m01	0.9	A01, A02, A05, A07	0.7	0.63
m04	0.8	A02, A03, A04, A05	0.9	0.72
m06	0.7	A03, A04, A05, A06	0.9	0.63
m05	0.6	A03, A07	0.9	0.54

The selected metrics are m04, m01, and m06. Their union contains all seven Actions A01–A07. Thus three selected metrics activate seven Actions in this example, not three.

K.3 Sensitivity points and acquisition candidates

For a concrete evaluation of the manuscript's gradient score, the specified sensitivity protocol draws 128 scrambled Sobol vectors in the eligible Action coordinates, sets ineligible coordinates to zero, compiles them, and retains the feasible vectors as $\mathcal{Q}_t$. The actual $|\mathcal{Q}_t|$ is at most 128 and is recorded after filtering. An empty set triggers exploration rather than a division by zero. This is a specified gradient-evaluation protocol; the inspected v2 diagnosis code instead uses 128 Sobol probes to estimate positive posterior objective gain and combines it with confidence and repeated response support. The two model scores are distinct.

Acquisition uses a separate masked Sobol pool; the sensitivity points are not reused as its candidate set. In the reference vLLM profile, 2048 raw strength vectors are scored by Monte Carlo EI using 256 shared standard-normal draws. Eight highest-scoring starts are refined by L-BFGS-B over $[0, 1]^{|S_t|}$ for at most 40 iterations each. The same normal draws are retained during that local search. Raw and refined vectors are rescored together. In decreasing EI order, the controller compiles candidates and returns the first feasible, previously untested configuration. Consequently, 2048 is a raw seed count, not a guarantee of 2048 distinct executable settings. If all candidates are infeasible or duplicates, it returns `no_distinct_feasible_candidate`.

For example, candidates with EI values 0.14, 0.11, and 0.08 are examined in that order. If the first violates a coupled constraint and the second compiles to an already tested setting, the third is selected if valid. These numbers illustrate ordering only, not measured improvements.

K.4 Concrete revision checks

Control	vLLM reference profile	Database / Spark defaults
Review interval	Every BO round, when budget permits	Every review opportunity; interval 1
Evidence requirement	At least 2 cited compatible trials	At least 8 observations for evidence gating
Direction magnitude limit	Maximum normalized weight change 0.25	Weight-delta limit 2.0
Graph edits per proposal	At most 2	Validated against the supplied graph contract
Response repetitions	2	2
Probe strength	0.75	Selected by validation task construction
Metric response floor	0.01	0.01
Objective response floor	0.005	0.005
Direction gain threshold	Relative gain strictly above 0.01 in each pair	Mean absolute objective reduction above 0; every pair positive
Rollback threshold	Relative throughput loss above 0.02	Objective deterioration above 0.05
Extra epoch probes	2 after direction changes	Governed by probe budget
Probe / validation budget	Charged against available total	8 / 8 attempts by default

In vLLM, δ is relative: a new direction must improve throughput by more than 1% over the matched old direction in each of two repetitions. Both settings must compile to distinct configurations. An

AM addition must show the proposed sign and exceed the 0.01 response floor in repeated held-out pairs. MY additions also require objective co-response above 0.005 with the specified sign. Missing responses or unsuccessful paired executions reject the proposal. Removal checks test negligible response. Responses are relative changes $(\text{after} - \text{before})/\max(|\text{before}|, 10^{-9})$. These response tests are not statistical p-value thresholds.

The database/Spark gate uses a chronological 75% training prefix (at least four rows) and at least two held-out rows. Ridge regression uses penalty 10^{-3} and training-fitted input standardization; squared prediction errors are normalized by training-response variance. Adding parents requires an error reduction of at least 0.01; removal permits an increase of at most 0.01 in its standardized validation error. Direction comparisons use two fresh candidate–control pairs and the raw minimized objective. Hence its zero absolute gain threshold must not be substituted for the vLLM 1% relative threshold. Validation costs are checked before execution, and confirmation remains protected.

K.5 Compiler rules by adapter

For the vLLM adapter, a numeric parameter is first formed as

$$u_i = x_{b,i} + a_i \sum_a W_{ia} z_a,$$

where a_i is the catalog's physical Action scale. It is clipped to its bounds before integer rounding. Integers use decimal half-up rounding, then half-up alignment to the permitted grid and projection to an in-range grid point. Continuous values retain the clipped value. In the ten-row KV catalog, numeric scales in row order are 32, 2048, not applicable, 0.02, 1, 2, 512, not applicable, not applicable, not applicable. Row 6 remains zero because its domain is fixed; rows 2 and 7 align to 256.

For Boolean parameters the base vote is $+0.35$ for True and -0.35 for False. Add the weighted Action votes. A total above 0.35 gives True, below -0.35 gives False, and the closed interval between them retains the base value. Ordered choices start with zero vote: a vote above 0.35 moves one legal index upward; below -0.35 moves one index downward; otherwise the index is unchanged. Endpoints saturate.

The database/Spark adapter instead adds directions in unit coordinates and clips to $[0, 1]$. Numeric coordinates decode linearly to their physical range, then snap to the declared grid. A category with K ordered values uses index $\lfloor u(K-1) + 0.5 \rfloor$; Boolean values use the ordered list `[false,true]`. This is an encoding order, not a statement that one serializer or compression algorithm is intrinsically better.

Contributions are summed before these mappings, so conflicting Actions can cancel. Coupled checks run on the final configuration. For vLLM, the token budget must be at least the sequence count, and at least the model length if chunked prefill is disabled; the token budget and long prefill threshold align to 256. Invalid combinations are rejected rather than silently repaired. Duplicate detection hashes the compiled setting with its workload/model identity, not the floating-point vector alone.

K.6 A complete numeric compilation example

Use the ten-dimensional KV reference and W_0 in Section H, and set

$$\mathbf{z} = (0, 0.6, 0, 0.8, 0.8, 0, 0.5)^\top.$$

The encoded displacement is

$$W_0\mathbf{z} = (1.095, 0.305, -0.73, 0.86, 0.19, 0, -0.565, 1.21, 1.15, -0.435)^\top.$$

For the first row, $32 + 32(1.095) = 67.04$, rounded to 67. For the second, $1024 + 2048(0.305) = 1648.64$, rounded to 1649 and aligned to 1536. GPU memory utilization becomes $0.70 + 0.02(0.86) = 0.7172$. The long-prefill threshold gives $512 + 512(-0.565) = 222.72$, clipped to 256 and retained on the 256 grid. The block-size vote -0.73 moves from 32 to 16. Prefix caching has total vote $-0.35 + 1.15 = 0.80$ and switches on; chunked prefill remains on, and eager execution remains off. The final configuration, in the declared ten-coordinate order, is

$$\mathbf{x} = (67, 1536, 16, 0.7172, 3.19, 0, 256, \text{True}, \text{True}, \text{False}).$$

The token budget exceeds the sequence count and both aligned parameters are multiples of 256. This calculation uses the displayed seed matrix; it is a compiler example, not a claim that this configuration was measured.