Empowering Edge Agents: A Systematic Analysis of Edge-Cloud Collaboration in LLM Agent Execution

Pengcheng Huang[1], **Zhenghao Liu**[1*], **Chaojun Xiao**[2*],
Yukun Yan[2*], **Shi Yu**[2], **Sijia Yao**[3], **Yu Gu**[1], **Ge Yu**[1], **Maosong Sun**[2]
[1]School of Computer Science and Engineering, Northeastern University, Shenyang, China
[2]Department of Computer Science and Technology, Tsinghua University, Beijing, China
[3]China Mobile Online Services Co., Ltd. Beijing, China

Abstract

LLM-powered agents increasingly operate over local workspaces involving files, tools, commands, APIs, and application states. However, cloud-only and edge-only execution struggle to balance task utility, resource usage, and privacy. Edge-cloud collaboration offers a promising alternative by selectively using cloud-side capability, but its trade-offs remain poorly understood in realistic agent execution. Existing evaluations mostly focus on static, non-agentic tasks and overlook how cloud calls in multi-step agents expose evolving local context. We introduce ACE-BENCH (**A**gentic **C**loud-**E**dge Collaboration Benchmark), a multi-dimensional benchmark and evaluation framework for studying edge-cloud collaboration in workspace-grounded LLM agents. ACE-BENCH contains 128 executable real-world digital tasks, including 100 tasks with fine-grained privacy annotations for auditing sensitive-context exposure. Using ACE-BENCH, we evaluate representative single-side and edge-cloud collaboration strategies. Our results show that edge-cloud collaboration achieves a better utility–cost–privacy balance than purely edge-side or cloud-side execution, but the resulting trade-off depends strongly on how collaboration is organized, especially when cloud models are invoked and what execution context is transmitted. All codes and datasets are available at https://github.com/OpenBMB/AceBench.

1 Introduction

LLM-powered agents are becoming a major paradigm for applying foundation models to real-world digital tasks (Pan et al., 2024). Recent systems such as Claude Code (Anthropic, 2025) and OpenClaw (OpenClaw, 2025) move agent execution beyond isolated tool invocation toward workspace-based interaction, where agents pursue user goals by iteratively operating over tools, files, commands, APIs, and application states (Schick et al., 2023; Yao et al., 2022). This interaction enables complex workflows such as software development, data analysis, and information seeking (Shen et al., 2023; Peng et al., 2025; Wang et al., 2024), but also requires strong model capabilities in long-horizon planning, state tracking, and error recovery (Liu et al., 2024).

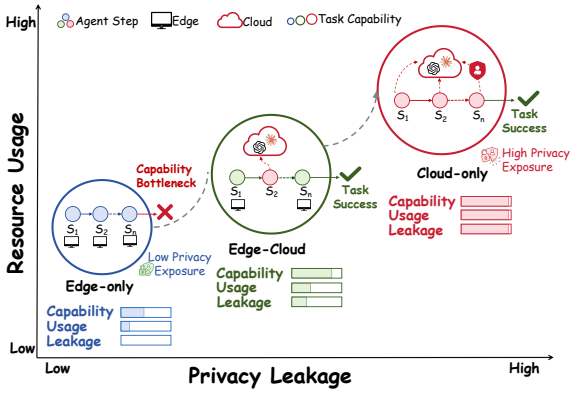

Figure 1: Comparison of edge-only, cloud-only, and edge-cloud execution. Edge-only reduces resource usage and privacy leakage, but is constrained by local model capability. Cloud-only offers stronger capability, yet incurs higher resource usage and privacy exposure. Edge-cloud execution selectively coordinates edge and cloud models for a more favorable trade-off among capability, resource usage, and privacy preservation.

Consequently, high-performing agent systems still rely heavily on cloud-hosted frontier models such as OpenAI GPT (Achiam et al., 2023) and Anthropic Claude (Anthropic, 2024). However, cloud-centric agent execution introduces a distinct privacy concern: as agents interact with local workspaces, evolving contextual information, including files, tool outputs, and application states, may be repeatedly transmitted to cloud models. Such context can contain sensitive personal or organizational information, making cloud invocation a potential channel for privacy leakage (Saad-Falcon et al.,

* indicates corresponding author.

2026). Executing agents on edge devices can alleviate this risk by retaining both inference and local context on the user side (Kan et al., 2023; Tian et al., 2025). Nevertheless, the limited computational and memory resources of edge devices typically restrict deployment to smaller-scale LLMs, causing edge-only agents to underperform on capability-intensive tasks (Xu et al., 2024). As illustrated in Figure 1, this results in an inherent trade-off between cloud-side capability and edge-side privacy preservation. Edge-cloud collaborative frameworks provide a promising compromise (Jin and Wu, 2025). However, their effectiveness critically depends on the collaboration strategy, particularly how and at what granularity edge and cloud models coordinate during execution.

Existing studies have explored different edge-cloud collaboration methods, including pre-execution coordination strategies such as pipeline-style collaboration and task-level routing (Zhang et al., 2024; Ong et al., 2024), as well as runtime coordination mechanisms such as step-level routing and on-demand cloud assistance (Chen et al., 2023; Kim et al., 2025). However, prior work has primarily evaluated edge-cloud collaboration on static, non-agentic tasks, such as mathematical reasoning, question answering, and dialogue (Li et al., 2026; Lu et al., 2025; Huang et al., 2025; Hu et al., 2024), with a primary focus on response quality and computational cost. This setting, however, misses the dynamic nature of agent execution, where cloud calls occur along multi-step trajectories and may expose accumulated local context.

To address this gap, we introduce ACE-BENCH (**A**gentic **C**loud-**E**dge Collaboration Benchmark), a multi-dimensional benchmark for evaluating edge-cloud collaboration in harness-based, workspace-grounded agent execution. ACE-BENCH focuses on two key questions: (1) how edge-cloud collaboration performs when agents interact with tools, files, and evolving environment states; and (2) how different collaboration strategies trade off task utility, resource use, and privacy exposure. ACE-BENCH contains 128 executable workspace-grounded agent tasks with task-specific verifiers, including 100 with fine-grained privacy annotations for personally sensitive information and organizationally confidential data.

Using ACE-BENCH, we evaluate single-side and representative edge-cloud collaboration strategies. Our results show that edge-cloud collaboration can achieve a more favorable utility–cost–privacy balance than purely edge-side or cloud-side execution. However, this balance depends strongly on how collaboration is organized: different mechanisms incur different resource usage and privacy exposure depending on when the cloud is invoked and what execution context is transmitted. Further analysis shows that edge-model capability reshapes the collaboration trade-off. Stronger edge models reduce the need for cloud involvement, allowing collaboration strategies to lower cloud cost and privacy exposure while maintaining utility. The preferred cloud-use pattern also shifts from step-level fallback for weaker edge models toward task routing, planning, and recovery guidance for stronger edge models. Nevertheless, even selective cloud use can still expose sensitive local context. These findings suggest that future edge-cloud agent systems should jointly optimize model-use and context-sharing decisions, rather than treating cloud invocation as a simple quality–cost trade-off.

2 Related Work

A growing body of research investigates edge-cloud collaboration for LLM inference, aiming to leverage the heterogeneity of model capabilities, task complexities, and resource availability to reduce computational costs while maintaining output quality (Zhu and Yang, 2025; Dong et al., 2025). Existing approaches can be categorized based on when cloud involvement is determined (Chen et al., 2025). One line of work emphasizes pre-execution coordination, where the collaboration strategy is established prior to execution. Representative examples include pipeline-style collaboration, which assigns fixed roles to edge and cloud models during generation (Zhang et al., 2024; Lv et al., 2025), and task-level routing, which employs an additional routing module to determine whether an input should be processed locally or offloaded to the cloud (Ong et al., 2024; Ding et al., 2024).

Another line of research focuses on *runtime coordination*, where cloud involvement is dynamically adjusted during inference based on runtime signals. This category includes several approaches: cascading methods, which first perform inference on an edge-side model and defer to a cloud-side model when the edge result is uncertain or insufficient (Chen et al., 2023; Zeng et al., 2026b); step-level routing, which makes model-selection decisions at intermediate reasoning steps; token-level speculative decoding, where the edge-side

model drafts tokens and the cloud-side model verifies them (Zhang et al., 2026; Bhattacharjee et al., 2025); and assistive collaboration, where one model requests on-demand support from the other for planning, hints, verification, or recovery during execution (Kim et al., 2025; Anthropic, 2026).

Despite these diverse mechanisms, systematic benchmarks for LLM collaboration have mostly studied model selection over standalone inference tasks. RouterBench (Hu et al., 2024), RouterEval (Huang et al., 2025), RouterArena (Lu et al., 2025), and LLMRouterBench (Li et al., 2026) provide standardized protocols for comparing routers under quality-oriented or quality-cost objectives, covering broad model pools, query categories, and routing metrics. These benchmarks are useful for studying how to select among available models for a given input. In parallel, recent harness-based agent benchmarks focus on multi-step tasks in interactive workspaces, where agents interact with tools, files, services, and evolving environment states across a trajectory (Ding et al., 2026; Ye et al., 2026). Such trajectory-level execution also makes privacy exposure more salient, since intermediate cloud calls may operate over local context accumulated during workspace interaction. Our work brings edge-cloud collaboration into this agent setting, evaluating how representative collaboration strategies affect task completion, resource usage, and trajectory-level privacy exposure.

3 ACE-BENCH: A Multi-Dimensional Benchmark for Agentic Edge-Cloud Collaboration

In this section, we present ACE-BENCH, a multi-dimensional benchmark for evaluating edge-cloud collaboration in executable-workspace agent tasks. We first formalize edge-cloud collaboration in the context of LLM-agent execution (§3.1). We then describe how ACE-BENCH executes a candidate collaboration strategy and converts the resulting trace into multi-dimensional evaluation results (§3.2). Finally, we describe the construction of the executable task suite and the sensitivity-unit annotations that support privacy evaluation (§3.3).

3.1 Preliminary: Edge-Cloud Agent Collaboration

We first formalize the agent execution framework considered in this work and then describe how edge-cloud collaboration extends this framework through coordinated interactions between edge-side and cloud-side models. We consider an agent that completes a user-specified digital task via iterative interactions with an environment. The environment consists of the local workspace, available tools, application states, external information interfaces, and feedback generated by executed actions.

Agent Execution Framework. Formally, task execution proceeds as a sequence of discrete interaction steps. At step t, the agent conditions on the task instruction I_{Task} together with the previous interaction history:

$$h_t = (y_1, a_1, o_1, \ldots, y_{t-1}, a_{t-1}, o_{t-1}), \tag{1}$$

where y_i, a_i, and o_i denote the agent output, executed action, and environment observation at step i, respectively. Based on h_t, the agent produces a new output y_t and an action a_t. The environment then executes a_t and returns the subsequent observation o_t. This interaction loop continues until task completion, yielding the execution trajectory:

$$\tau = \{(y_t, a_t, o_t)\}_{t=1}^{T}. \tag{2}$$

Edge-Cloud Collaboration. Building upon the standard execution loop above, edge-cloud collaboration replaces the single-model step output with a collaboration-aware output generated through coordinated edge-cloud execution. We use z_t to denote the step-level output produced under a collaboration strategy $\mathcal{C}$ and exposed to subsequent agent steps. The corresponding collaboration-aware interaction history is defined as:

$$h_t^{\mathcal{C}} = (z_1, a_1, o_1, \ldots, z_{t-1}, a_{t-1}, o_{t-1}). \tag{3}$$

At step t, the collaboration strategy $\mathcal{C}$ coordinates the edge-side model M_e and the cloud-side model M_c to generate the current step output and the environment action:

$$(z_t, \mathcal{V}_t^c, \text{Tok}_t^c, a_t) = \mathcal{C}(h_t^{\mathcal{C}}; M_e, M_c), \tag{4}$$

where $\mathcal{V}_t^c$ denotes the context transmitted to the cloud model at step t, Tok_t^c denotes the associated cloud-side token usage, and a_t denotes the environment action when applicable. If the cloud model is not invoked at step t, then $\mathcal{V}_t^c = \emptyset$ and $\text{Tok}_t^c = \emptyset$. Repeated execution under $\mathcal{C}$ produces an instrumented execution trace:

$$\xi_{\mathcal{C}} = \big((z_t, \mathcal{V}_t^c, \text{Tok}_t^c, a_t, o_t)\big)_{t=1}^{T}. \tag{5}$$

This formulation captures how edge-cloud collaboration unfolds throughout execution. Different

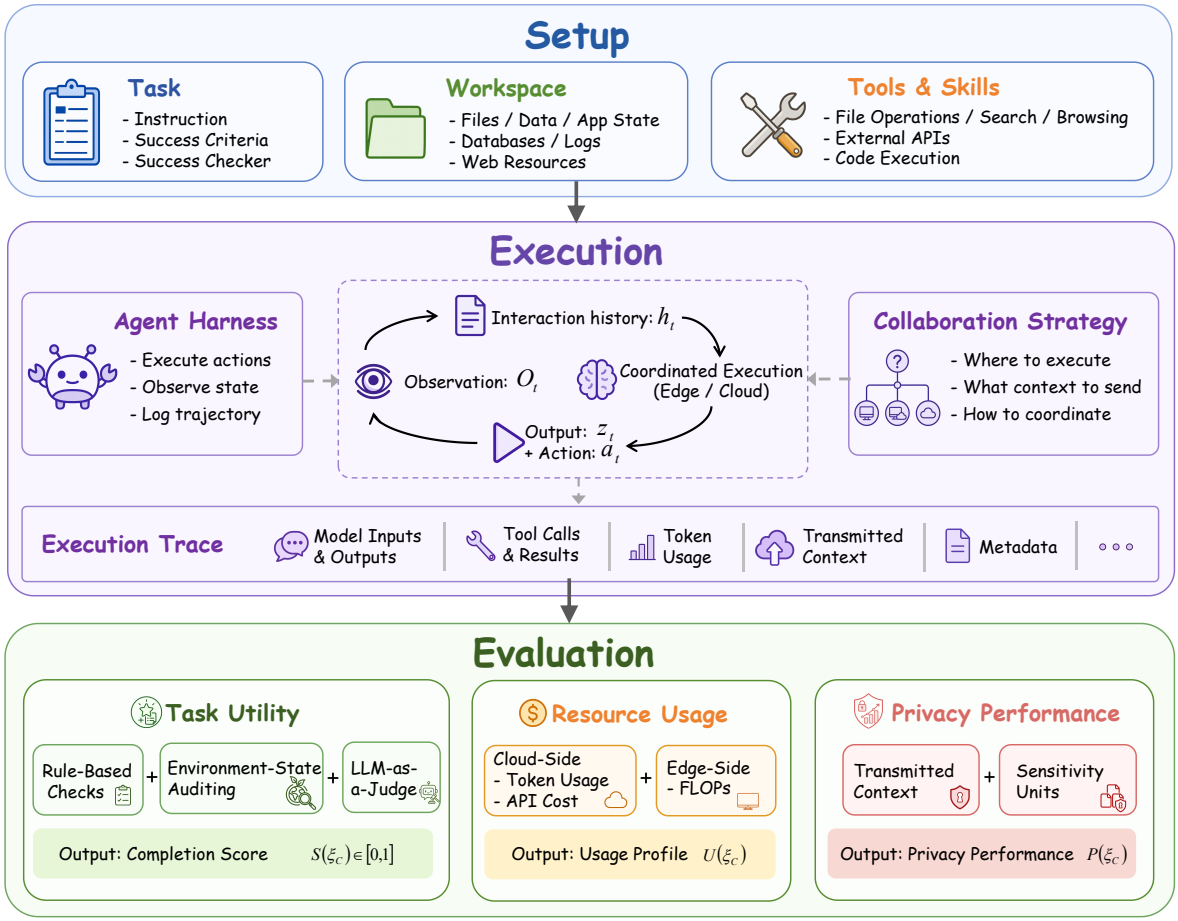

Figure 2: Overview of ACE-BENCH. Each task instance includes a user instruction, workspace resources, and available tools and skills. Given an edge-cloud collaboration strategy, the harness records the agent execution trace and evaluates task completion, resource usage, and privacy exposure.

strategies instantiate $\mathcal{C}$ differently: task-level strategies may assign the whole trajectory to a single executor, while runtime strategies may keep some steps local and invoke the cloud only for selected steps. For example, in the cascading method, easy steps are handled by M_e with $\mathcal{V}_t^c = \emptyset$, whereas uncertain steps transmit context to M_c for cloud-assisted generation. Thus, $\xi_{\mathcal{C}}$ may contain a mixture of edge-generated outputs, cloud-generated outputs, and jointly generated edge-cloud outputs, while explicitly recording the information exposed to the cloud at each step.

3.2 Benchmarking Edge-Cloud Agent Collaboration

Figure 2 illustrates the overall benchmarking workflow of ACE-BENCH. Given a standardized task instance and a candidate collaboration strategy $\mathcal{C}$, ACE-BENCH executes the agent under $\mathcal{C}$ within the harness and records an execution trace $\xi_{\mathcal{C}}$. The trace captures both agent-environment interactions and cloud-side observations, including cloud-visible contexts and token-usage statistics. Based on $\xi_{\mathcal{C}}$, ACE-BENCH produces a multi-dimensional evaluation profile consisting of task utility S, resource usage U, and privacy performance P.

Task Utility. Task utility evaluates how effectively the agent completes the target task. Each task is associated with a success specification that defines the expected outcome, such as required file modifications, tool outputs, application-state transitions, or semantic constraints. ACE-BENCH applies the corresponding success checker to the completed execution trace and computes a completion score $S(\xi_{\mathcal{C}}) \in [0, 1]$. Depending on the task, the checker may combine rule-based validation, environment-state auditing, and rubric-based LLM-as-a-judge evaluation. In addition to the completion score, we report Pass^n, defined as the proportion of tasks that satisfy a predefined success threshold across all n repeated trials.

Resource Usage. Resource usage characterizes

the computational cost incurred during execution. Unlike task utility, resource usage is reported as a structured profile rather than a single scalar metric. Specifically, $U(\xi_{\mathcal{C}})$ includes aggregated cloud token consumption, estimated cloud monetary cost, and edge-side FLOPs.

For each cloud invocation, the execution trace records a token-usage tuple $\mathrm{Tok}_t^c = (T_{\mathrm{raw}}^t, T_{\mathrm{cache}}^t, T_{\mathrm{out}}^t)$, corresponding to raw input tokens, cache-read input tokens, and output tokens, respectively. These statistics are aggregated over all cloud calls and reported separately. For ease of comparison, we additionally estimate the corresponding monetary cost according to provider pricing. We do not aggregate cloud and edge computation into a unified resource score, since they exhibit fundamentally different cost semantics. Instead, we focus comparison on cloud token usage and estimated provider cost, as cloud invocation constitutes the explicit marginal cost that edge-cloud collaboration seeks to reduce. Edge-side FLOPs are reported separately to reflect the local computational burden, whose realized cost depends on deployment hardware and runtime conditions.

Privacy Performance. Privacy performance measures the extent to which a collaboration strategy prevents sensitive local information from being exposed to the cloud model during execution.

From the execution trace $\xi_{\mathcal{C}}$, we collect the cloud-visible contexts along the trajectory as $\mathcal{V}_T^c = \{\mathcal{V}_t^c\}_{t=1}^T$. For tasks equipped with sensitivity-unit annotations, let $\mathcal{U}$ denote the annotated unit set and $\tilde{w}(u)$ denote the normalized risk weight associated with unit u; the construction of these annotations and weights is described in §3.3. We define privacy performance as the following risk-weighted non-leakage score:

$$P(\xi_{\mathcal{C}}) = 1 - \sum_{u \in \mathcal{U}} \tilde{w}(u) \cdot \mathbb{I}_{\mathrm{leak}}\left(u, \mathcal{V}_T^c\right), \tag{6}$$

where $\mathbb{I}_{\mathrm{leak}}$ indicates whether sensitivity unit u appears in any cloud-visible context during execution. Higher values of $P(\xi_{\mathcal{C}})$ indicate stronger privacy preservation. If no cloud model is invoked, then $\mathcal{V}_t^c = \emptyset$ for all t, yielding a privacy score of 1.

3.3 Task Construction and Privacy Annotation

This subsection describes how we construct the benchmark assets: standardized executable task instances and sensitivity annotations for privacy evaluation. The goal is to obtain realistic agentic tasks that support reliable utility, resource, and privacy measurements.

Task Collection and Standardization. We construct the task suite from two complementary sources. First, we curate tasks from Claw-style executable-workspace agent benchmarks. To ensure broad coverage of agentic operations, we perform coverage-guided task selection based on the real-world agentic operation taxonomy proposed by Z.AI (2025). The resulting benchmark covers a diverse range of scenarios, including Information Retrieval, Content Generation, Data Analysis, Office Productivity, Development, and Workflow Automation. Second, we manually design a small set of additional tasks to better capture privacy-sensitive edge-cloud scenarios. Following the protocol of Ding et al. (2026), we further transform all candidate tasks into standardized executable instances by normalizing task specifications, workspace structures, tool interfaces, and verifier formats. Detailed task selection procedures, manual task design details, source statistics, and category distributions are provided in Appendix A.2, while task adaptation protocols are described in Appendix A.3.

Privacy Annotation. Existing executable-workspace agent benchmarks primarily focus on task completion and lack explicit annotations regarding which local information should be audited once exposed to cloud-based models from a privacy perspective. To address this gap, we augment the task suite with sensitive local context and annotate it as *sensitivity units*. Following prior work (Shao et al., 2024), we define a sensitivity unit as an atomic piece of private or confidential information whose exposure to a cloud-side model constitutes a privacy leakage event.

We construct sensitivity-unit annotations in two ways: preserving and annotating existing local or private information, or injecting synthetic sensitive content into natural task locations when the original task lacks explicit sensitive information, while preserving the task objective, execution workflow, and success criteria used for utility evaluation. All injected content is synthetically generated and does not reuse real private data. Each sensitivity unit records four fields: *value*, *source location*, *privacy category*, and *risk weight*. We classify units into two broad domains, *Personal Sensitive Information* and *Organizational Confidential Information*, with 12 fine-grained subcategories in total. The risk

weight distinguishes leakage events with different potential harms, so that exposing a highly sensitive unit incurs a larger privacy penalty than exposing a low-risk unit. To ensure annotation quality, we apply LLM-based consistency checks and use manual human review to resolve ambiguous or inconsistent annotations. Detailed annotation procedures, category definitions, statistics, and weighting details are provided in Appendix A.4.

4 Experimental Setup

This section describes the experimental setup for evaluating different strategies on ACE-BENCH. We first introduce the evaluated single-side and edge-cloud collaboration strategies (§4.1), and then present the experimental settings (§4.2).

4.1 Edge-Cloud Collaboration Strategies

We evaluate six representative execution strategies, including two single-side baselines and four edge-cloud coordination patterns: pipeline-style collaboration, task-level routing, step-level routing, and assistive collaboration. For each coordination pattern, we implement one concrete strategy and summarize it below, with full implementation details provided in Appendix A.5.

Edge-only. All agent steps are handled by the edge-side model M_e. Thus, no cloud model is invoked and $\mathcal{V}_t^c = \emptyset$ for all steps, avoiding cloud-side resource usage and privacy exposure, but performance may be limited by the capability of the local model.

Cloud-only. All agent steps are handled by the cloud-side model M_c. This strategy provides an upper reference point for model capability, but incurs the highest cloud-side resource usage and may expose sensitive execution context to the cloud.

Edge-Cloud Collaboration. We evaluate four representative edge-cloud collaboration strategies, each corresponding to a different way of coordinating M_e and M_c during agent execution.

Sketch-Guided Edge Execution. We instantiate pipeline-style collaboration as a sketch-guided strategy following (Zhang et al., 2024). The cloud-side model M_c first generates high-level execution guidance from the initial task context, and the edge-side model M_e then performs all subsequent steps conditioned on this guidance. This strategy uses the cloud only for upfront task structuring while keeping the main execution on the edge.

Task-Level Routing. This strategy makes a one-time routing decision from the user instruction and assigns the entire trajectory to either M_e or M_c. We instantiate the router with the matrix-factorization-based routing model from RouteLLM (Ong et al., 2024), which estimates the relative preference between the edge-side and cloud-side models for each task and selects the model with the higher predicted score for execution.

Step-level Routing. Step-level routing follows an edge-first escalation pattern: M_e first attempts each step, and the step is escalated to M_c when the edge model is judged uncertain. We instantiate the escalation rule with the entropy of the edge model's next reasoning-token distribution, following GlimpRouter (Zeng et al., 2026b). Thus, $\mathcal{V}_t^c$ is exposed only for escalated steps.

Adaptive Cloud Assistance. We instantiate assistive collaboration as an edge-primary strategy inspired by prior assistance-based settings (Kim et al., 2025; Anthropic, 2026). At each step, M_e serves as the primary executor and decides whether to invoke M_c for high-level assistance. When assistance is requested, M_e sends an assistance request based on the current execution state; M_c returns a plan or hint; and M_e continues the step conditioned on this assistance. The resulting collaboration-aware output z_t records this assisted interaction, including the cloud-assistance request, the cloud-provided plan or hint, and the subsequent edge-side continuation. In this strategy, $\mathcal{V}_t^c$ is exposed only at steps where cloud assistance is requested.

4.2 Experimental Settings

We implement all strategies in the OpenClaw harness[1] under a fixed configuration across methods. For the single-side baselines, Edge-only is evaluated with Qwen3.5-9B and Qwen3.5-27B (Qwen Team, 2026), while Cloud-only is evaluated with GPT-5.4 (OpenAI, 2026), DeepSeek-v4-Flash, DeepSeek-v4-Pro (DeepSeek-AI, 2026), and GLM-5.1 (Zeng et al., 2026a). For edge-cloud collaboration, we use Qwen3.5-9B and Qwen3.5-27B as edge-side models and GPT-5.4 as the cloud-side model. For utility and privacy evaluation, we use GPT-5.4-mini as the LLM-as-a-judge model, with reasoning mode enabled following (Zharmagambetov et al., 2026). Privacy performance is computed only on the 100 privacy-annotated tasks. Each strategy is evaluated over three runs, with results averaged across runs unless otherwise specified.

[1] https://openclaw.ai/

Method	Utility		Resource Usage			Privacy
	Completion	Pass^3	Cloud Tok. (R/C/O)	Cloud Cost	Edge FLOPs	
Edge-only						
Qwen3.5-9B (2026)	40.09	9.40	–	–	121.41	100.00
Qwen3.5-27B (2026)	58.52	35.20	–	–	446.84	100.00
Cloud-only						
GPT-5.4 (2026)	71.81	49.20	5.46 / 22.92 / 1.02	34.69	–	22.09
DeepSeek-v4-Flash (2026)	69.47	51.56	7.73 / 61.21 / 1.53	1.68	–	12.68
DeepSeek-v4-Pro (2026)	72.90	53.12	5.81 / 50.20 / 1.60	16.41	–	13.34
GLM-5.1 (2026a)	72.46	55.50	12.66 / 75.72 / 1.26	30.08	–	14.74
Edge-Cloud Collaboration						
Qwen3.5-9B						
Sketch-Guided (2024)	46.32	14.10	0.07 / 0.00 / 0.03	0.58	148.45	100.00
Task-Routing (2024)	46.62	25.78	2.25 / 9.43 / 0.40	13.94	85.91	88.81
Step-Routing (2026b)	55.58	19.50	2.63 / 7.67 / 0.31	13.06	23.48	58.06
Adaptive Assist. (2026)	53.75	18.80	3.30 / 2.90 / 0.20	12.03	280.23	58.59
Qwen3.5-27B						
Sketch-Guided (2024)	62.03	35.20	0.05 / 0.00 / 0.02	0.42	546.56	100.00
Task-Routing (2024)	66.15	40.62	2.13 / 8.44 / 0.36	12.75	181.75	84.32
Step-Routing (2026b)	64.98	41.40	2.23 / 7.10 / 0.26	11.19	111.21	59.48
Adaptive Assist. (2026)	65.01	42.97	0.78 / 0.40 / 0.07	3.07	495.73	67.03

Table 1: Overall performance of edge-cloud collaboration strategies on ACE-BENCH. **Utility**: completion rate and Pass^3. **Resource Usage**: Cloud Tok. (R/C/O) reports raw input / cache-read input / output tokens in millions; Cloud Cost is estimated in USD; Edge FLOPs are reported in PetaFLOPs. **Privacy**: privacy performance.

For Pass^n, we binarize each trial by considering a task completed if its completion score exceeds 0.7. Cloud-side monetary cost is computed using OpenRouter[2] pricing over raw input, cache-read input, and output tokens. Edge-side computation is estimated as FLOPs following Kaplan et al. (2020). Detailed pricing assumptions and FLOPs calculation are provided in Appendix A.6.

5 Results and Analysis

This section evaluates how different execution strategies trade off task utility, cloud cost, and privacy performance on ACE-BENCH. We first compare single-side baselines with representative edge-cloud collaboration strategies, and then analyze how edge-model capability affects the preferred collaboration strategy.

5.1 Overall Performance of Edge-Cloud Collaboration Strategies

Table 1 shows the trade-offs among task utility, resource consumption, and privacy performance achieved by different strategies on ACE-BENCH.

The results reveal the distinct characteristics of single-side agents and edge-cloud collaboration methods. Among the single-side strategies, Cloud-only execution achieves the highest utility score, but its privacy performance remains below 25%. In contrast, Edge-only execution eliminates cloud exposure, yet reduces the utility score by more than 10%. These findings highlight the inherent limitations of single-side systems, as neither can simultaneously achieve a satisfactory balance among utility, resource efficiency, and privacy preservation in practical agent deployment scenarios. Compared with these single-side strategies, edge-cloud collaboration methods achieve a substantially better trade-off between utility and privacy. By selectively leveraging cloud capabilities only when necessary, these methods significantly improve utility, while still maintaining stronger privacy protection.

We next compare different edge-cloud collaboration strategies to evaluate their effectiveness. Among all strategies, *Sketch-Guided* achieves the highest privacy score but yields the lowest utility score. This is mainly because the cloud agent only provides a high-level sketch for task completion, preventing further cloud assistance for recovering from reasoning errors, incorporating tool feedback, or resolving stalled intermediate states. In contrast, *Task-Routing* achieves a more balanced trade-off between utility and privacy by routing tasks to different agents in advance. With Qwen3.5-27B, it achieves the highest utility among all evaluated collaboration strategies while still preserving 84.32%

[2] https://openrouter.ai/

privacy performance. However, its advantage becomes much smaller with Qwen3.5-9B, suggesting that task-level routing still heavily depends on the capability of edge-side models to perform effective routing decisions.

Unlike strategies whose cloud-use pattern is fixed before the main execution, *Step-level routing* and *Adaptive Cloud Assistance* decide whether to involve the cloud at runtime, allowing cloud assistance to be triggered based on the evolving execution state. As shown in the results, *Step-level routing* is particularly effective for weaker edge models, where cloud escalation helps compensate for uncertain edge-side reasoning steps. However, this benefit comes at a noticeable privacy cost because intermediate execution contexts are repeatedly exposed at escalated steps. Meanwhile, *Adaptive Cloud Assistance* achieves the highest Pass[3] among all collaboration strategies with Qwen3.5-27B, while incurring less than 10% of the cost of GPT-5.4 Cloud-only execution. This result suggests that stronger edge models are better able to leverage cloud-provided plans and hints.

5.2 Pareto Analysis of Utility, Cost, and Privacy across Edge Backbones

We further analyze how edge-model capability influences the Pareto trade-off among task utility, cloud cost, and privacy in edge-cloud collaboration. Figure 3 plots each collaboration strategy as a point in this three-dimensional trade-off space across three edge backbones: Qwen3.5-9B, Qwen3.5-27B, and Qwen3.6-27B (Qwen Team, 2026).

First, stronger edge models reduce the need for cloud involvement. With a constrained edge backbone such as Qwen3.5-9B, improving task utility often requires more frequent cloud calls, which increases both cloud cost and privacy exposure. As the edge backbone becomes stronger, the local model can handle more execution steps reliably, reducing the marginal benefit of transmitting intermediate context to the cloud. Consequently, collaboration strategies can achieve competitive completion scores with fewer cloud invocations, leading to lower cloud cost and better privacy preservation.

Second, edge-model capability fundamentally reshapes the interaction pattern with cloud agents. When the edge model is relatively weak, step-level routing is particularly beneficial, as uncertain intermediate reasoning steps can be offloaded to the cloud for assistance, requiring more frequent intervention from the cloud model. However, as

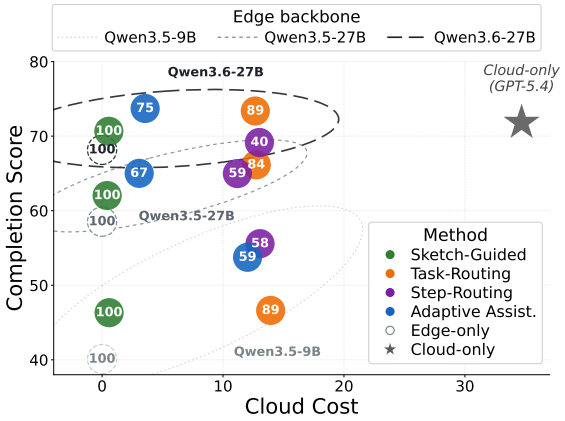

Figure 3: Utility–cost–privacy trade-off across edge backbones. Each point denotes an evaluated strategy, with cloud cost on the x-axis, completion score on the y-axis, and the number inside the point indicating privacy performance. Dashed contours group strategies using the same edge backbone, while hollow circles and the star denote Edge-only and Cloud-only references.

the edge model becomes stronger, this fallback mechanism becomes less appealing: local execution is already more reliable, while each cloud invocation may still expose accumulated execution context and incur additional privacy risks. Consequently, stronger edge models shift the preferred collaboration paradigm from cloud-assisted execution toward cloud-coordinated guidance under cost and privacy constraints. Nevertheless, improved edge capability does not eliminate privacy concerns, since even selective cloud interactions may still leak sensitive local context.

6 Conclusion

We introduced ACE-BENCH, a multi-dimensional benchmark for studying edge-cloud collaboration in executable-workspace LLM-agent tasks. Experiments show that edge-cloud collaboration can improve the utility–cost–privacy trade-off over purely edge-side or cloud-side execution, but its effectiveness depends on cloud-invocation timing and transmitted context. Further analysis shows that edge-model capability reshapes this trade-off: stronger edge models reduce cloud involvement and make lower-cost, more privacy-preserving strategies more viable, while the preferred cloud-use pattern shifts from step-level fallback toward task routing, planning, and recovery guidance. Nevertheless, even selective cloud use can still expose sensitive local context. Overall, these results call for jointly optimizing model-use and context-sharing decisions in edge-cloud agent systems.

Limitations

This work has two main limitations. First, ACE-BENCH focuses on executable-workspace LLM-agent tasks under a harness-based execution setting. Although this setting captures many realistic digital-agent workflows, it does not cover all possible deployment environments, such as GUI-only applications, mobile agents, or embodied agents. Second, our resource measurements focus on cloud-side token usage, monetary cost, and estimated edge-side FLOPs. Real deployment efficiency may also depend on latency, energy consumption, hardware utilization, and system-level optimization, which are deployment-dependent and difficult to compare fairly in a benchmark setting.

Ethics Statement

This work studies edge-cloud collaboration for LLM-powered agents, with a focus on evaluating and reducing unnecessary cloud exposure of local workspace context. The benchmark is designed for research evaluation and responsible system development. We do not use real private user data when constructing privacy-sensitive benchmark content. For tasks that require additional sensitive context, the inserted information is synthetically generated and does not correspond to real individuals, organizations, credentials, or financial records. The privacy annotations in ACE-BENCH are intended to support auditing of cloud-visible context and to encourage privacy-aware agent design.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. 2025. Claude code. Accessed: 2026-05-07.

Anthropic. 2026. The advisor strategy: Give agents an intelligence boost. https://claude.com/blog/the-advisor-strategy. Claude Blog. Accessed: 2026-05-17.

AI Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*, 1(1):4.

Payel Bhattacharjee, Fengwei Tian, Meiyu Zhong, Guangyi Zhang, Osvaldo Simeone, and Ravi Tandon. 2025. Conformal sparsification for bandwidth-efficient edge-cloud speculative decoding. *arXiv preprint arXiv:2510.09942*.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*.

Yi Chen, JiaHao Zhao, and HaoHao Han. 2025. A survey on collaborative mechanisms between large and small language models. *arXiv preprint arXiv:2505.07460*.

DeepSeek-AI. 2026. Deepseek-v4: Towards highly efficient million-token context intelligence.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks Lakshmanan, and Ahmed H Awadallah. 2024. Hybrid llm: Cost-efficient and quality-aware query routing. In *International Conference on Learning Representations*, volume 2024, pages 41348–41366.

Shuangrui Ding, Xuanlang Dai, Long Xing, Shengyuan Ding, Ziyu Liu, Yang JingYi, Penghui Yang, Zhixiong Zhang, Xilin Wei, Xinyu Fang, and 1 others. 2026. Wildclawbench: A benchmark for real-world, long-horizon agent evaluation. *arXiv preprint arXiv:2605.10912*.

Jiangwen Dong, Jiayu Li, Tianhang Zheng, and Wanyu Lin. 2025. Hybridflow: Resource-adaptive subtask routing for efficient edge-cloud llm inference. *arXiv preprint arXiv:2512.22137*.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. 2024. Routerbench: A benchmark for multi-llm routing system. *arXiv preprint arXiv:2403.12031*.

Zhongzhan Huang, Guoming Ling, Yupei Lin, Yandong Chen, Shanshan Zhong, Hefeng Wu, and Liang Lin. 2025. Routereval: A comprehensive benchmark for routing llms to explore model-level scaling up in llms. *arXiv preprint arXiv:2503.10657*.

Hongpeng Jin and Yanzhao Wu. 2025. Ce-collm: Efficient and adaptive large language models through cloud-edge collaboration. In *2025 IEEE International Conference on Web Services (ICWS)*, pages 316–323. IEEE.

Zhigang Kan, Linbo Qiao, Hao Yu, Liwen Peng, Yifu Gao, and Dongsheng Li. 2023. Protecting user privacy in remote conversational systems: A privacy-preserving framework based on text sanitization. *arXiv preprint arXiv:2306.08223*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Yujin Kim, Euiin Yi, Minu Kim, Se-Young Yun, and Taehyeon Kim. 2025. Guiding reasoning in small language models with llm assistance. *arXiv preprint arXiv:2504.09923*.

Hao Li, Yiqun Zhang, Zhaoyan Guo, Chenxu Wang, Shengji Tang, Qiaosheng Zhang, Yang Chen, Biqing Qi, Peng Ye, Lei Bai, and 1 others. 2026. Llmrouterbench: A massive benchmark and unified framework for llm routing. *arXiv preprint arXiv:2601.07206*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, and 1 others. 2024. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pages 52989–53046.

Yifan Lu, Rixin Liu, Jiayi Yuan, Xingqi Cui, Shenrun Zhang, Hongyi Liu, and Jiarong Xing. 2025. Routerarena: An open platform for comprehensive comparison of llm routers. *arXiv preprint arXiv:2510.00202*.

Zheqi Lv, Tianyu Zhan, Wenjie Wang, Xinyu Lin, Shengyu Zhang, Wenqiao Zhang, Jiwei Li, Kun Kuang, and Fei Wu. 2025. Collaboration of large language models and small recommendation models for device-cloud recommendation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 962–973.

Ivoline C Ngong, Keerthiram Murugesan, Swanand Kadhe, Justin D Weisz, Amit Dhurandhar, and Karthikeyan Natesan Ramamurthy. 2026. Agentscope: Evaluating contextual privacy across agentic workflows. *arXiv preprint arXiv:2603.04902*.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. 2024. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*.

OpenAI. 2026. Introducing GPT-5.4.

OpenClaw. 2025. Openclaw: Your own personal AI assistant. GitHub repository. Accessed: 2026-05-07.

Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, and 1 others. 2024. Webcanvas: Benchmarking web agents in online environments. *arXiv preprint arXiv:2406.12373*.

Pranoy Panda, Raghav Magazine, Chaitanya Devaguptapu, Sho Takemori, and Vishal Sharma. 2025. Adaptive llm routing under budget constraints. *arXiv preprint arXiv:2508.21141*.

Chunyi Peng, Zhipeng Xu, Zhenghao Liu, Yishan Li, Yukun Yan, Shuo Wang, Zhiyuan Liu, Yu Gu, Minghe Yu, Ge Yu, and 1 others. 2025. Learning to route queries across knowledge bases for stepwise retrieval-augmented reasoning. *arXiv preprint arXiv:2505.22095*.

Qwen Team. 2026. Qwen3.5: Towards native multimodal agents.

Jon Saad-Falcon, Avanika Narayan, Hakki Orhun Akengin, J Wes Griffin, Herumb Shandilya, Adrian Gamarra Lafuente, Rebecca Joseph, Etash Kumar Guha, Shlok Natarajan, Shang Zhu, and 1 others. 2026. Support your local lms: Redistributing lm traffic from cloud to edge with trafficbench.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551.

Yijia Shao, Tianshi Li, Weiyan Shi, Yanchen Liu, and Diyi Yang. 2024. Privacylens: Evaluating privacy norm awareness of language models in action. *Advances in Neural Information Processing Systems*, 37:89373–89407.

Hao Shen, Zhouhong Gu, Haokai Hong, and Weili Han. 2025. Pii-bench: Evaluating query-aware privacy protection systems. *arXiv preprint arXiv:2502.18545*.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180.

Chunlin Tian, Xinpeng Qin, Kahou Tam, Li Li, Zijian Wang, Yuanzhe Zhao, Minglei Zhang, and Chengzhong Xu. 2025. {CLONE}: Customizing {LLMs} for efficient {Latency-Aware} inference at the edge. In *2025 USENIX Annual Technical Conference (USENIX ATC 25)*, pages 563–585.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, and 1 others. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

WIPO. 2024. *WIPO Guide to Trade Secrets and Innovation*. WIPO, Geneva. WIPO Publication No. 2008.

Jiajun Xu, Zhiyuan Li, Wei Chen, Qun Wang, Xin Gao, Qi Cai, and Ziyuan Ling. 2024. On-device language models: A comprehensive review. *arXiv preprint arXiv:2409.00088*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Bowen Ye, Rang Li, Qibin Yang, Yuanxin Liu, Linli Yao, Hanglong Lv, Zhihui Xie, Chenxin An, Lei Li, Lingpeng Kong, and 1 others. 2026. Claw-eval: Toward trustworthy evaluation of autonomous agents. *arXiv preprint arXiv:2604.06132*.

Z.AI. 2025. Zclawbench: An end-to-end benchmark for evaluating agent performance. https://huggingface.co/datasets/zai-org/ZClawBench. Hugging Face Datasets.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, and 1 others. 2026a. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*.

Wenhao Zeng, Xuteng Zhang, Yuling Shi, Chao Hu, Yuting Chen, Beijun Shen, and Xiaodong Gu. 2026b. Glimprouter: Efficient collaborative inference by glimpsing one token of thoughts. *arXiv preprint arXiv:2601.05110*.

Guangyi Zhang, Yunlong Cai, Guanding Yu, Petar Popovski, and Osvaldo Simeone. 2026. Quantize-sample-and-verify: Llm acceleration via adaptive edge-cloud speculative decoding. *IEEE Communications Letters*, 30:852–856.

Kaiyan Zhang, Jianyu Wang, Ermo Hua, Biqing Qi, Ning Ding, and Bowen Zhou. 2024. Cogenesis: A framework collaborating large and small language models for secure context-aware instruction following. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4295–4312.

Arman Zharmagambetov, Chuan Guo, Ivan Evtimov, Maya Pavlova, Ruslan Salakhutdinov, and Kamalika Chaudhuri. 2026. Agentdam: Privacy leakage evaluation for autonomous web agents. *Advances in Neural Information Processing Systems*, 38.

Pengyan Zhu and Tingting Yang. 2025. Ce-lslm: Efficient large-small language model inference and communication via cloud-edge collaboration. *arXiv preprint arXiv:2505.14085*.

A Appendix

A.1 License

We strictly follow the original licenses associated with each dataset used to build ACE-BENCH. Claw-Eval, WildClawBench, QwenClawBench, LiveClawBench, and PinchBench are released under the MIT License; ClawBench is released under the Apache-2.0 License. All adapted tasks are used in accordance with their original licensing terms and with attribution to the corresponding sources.

A.2 Task Selection and Statistics

This section provides additional details on task selection, expert-curated task design, and statistics for the ACE-BENCH task suite. We first describe how tasks are selected from existing benchmarks and how additional expert-curated tasks are constructed, and then summarize the source and category distributions of the resulting task suite.

As described in Section 3.3, ACE-BENCH combines curated tasks from existing CLAW-style executable-workspace agent benchmarks with a small set of expert-curated tasks. Starting from an initial pool of candidate tasks collected from existing benchmarks, we adapt candidate tasks into the ACE-BENCH harness and validate the adapted instances as described in Appendix A.3. We then perform coverage-guided selection following the operation taxonomy used in prior work (Z.AI, 2025), so that the selected tasks cover diverse agentic operations rather than concentrating on a single task type. After adaptation validation and coverage-guided selection, we retain 121 tasks from existing benchmarks. Table 2 summarizes their source distribution.

Task Source	# Tasks	Percentage
Claw-Eval	68	53.12%
WildClawBench	22	17.19%
QwenClawBench	21	16.41%
LiveClawBench	4	3.12%
PinchBench	3	2.34%
ClawBench	3	2.34%
Expert-curated Tasks	7	5.47%
Total	128	100.00%

Table 2: Source distribution of the ACE-BENCH.

In addition, we design a small set of expert-curated tasks to better cover privacy-sensitive edge-cloud scenarios that are underrepresented in existing task suites. We construct these tasks through a scenario-driven process. First, we identify common privacy-sensitive agent workflows, such as processing private records, handling internal documents, using credentials, or operating over organization-specific information. Second, for each scenario, we use LLM-assisted synthesis to create a realistic user instruction and instantiate an executable workspace containing the necessary files and tools. Third, we define a success checker to verify whether the intended task outcome is achieved. During construction, we ensure that sensitive context appears naturally as part of the workspace rather than as an artificial instruction-level constraint. All expert-curated tasks follow the same executable-workspace format as the curated benchmark tasks. The final benchmark contains 128 tasks in total.

The resulting task suite spans six agentic operation categories, including Office & Daily Tasks, Information Search & Gathering, Safety & Security, Development & Operations, Data Analysis, and Automation. Table 3 reports the category distribution, showing that ACE-BENCH is not concentrated on a single type of workspace-agent task.

Task Category	# Tasks
Office & Daily Tasks	36
Information Search & Gathering	34
Safety & Security	21
Development & Operations	14
Data Analysis	13
Automation	10
Total	128

Table 3: Distribution of ACE-BENCH tasks across agentic operation categories.

A.3 Task Adaptation Protocol

This appendix provides additional details on how candidate tasks are adapted into standardized executable instances for ACE-BENCH. Since tasks are collected from different executable-workspace agent benchmarks, their original formats differ in task configuration files, workspace layouts, tool interfaces, and verification procedures. We therefore apply a unified adaptation protocol to ensure that all tasks can be executed, logged, and evaluated under the same harness.

We first normalize each task into a unified task configuration file with a metadata header and standardized sections for prompts, grading criteria, automated checks, workspace paths, tools, environment settings, and warmup instructions. We then normalize the workspace layout and execution settings. Task-relevant files are organized into a

Domain	Category	# Units	Percentage	Weight	# Tasks
Personal Sensitive Information	P1 Identity identifier	276	8.15%	1.0–3.0	48
	P2 Documents and accounts	457	13.50%	1.0–3.0	66
	P3 Contact information	1178	34.80%	1.0–3.0	84
	P4 Financial information	209	6.17%	1.5–3.0	48
	P5 Sensitive attributes	248	7.33%	1.5–3.0	26
	Subtotal	**2368**	**69.96%**	–	–
Organizational Confidential Information	S1 Code or algorithm	6	0.18%	2.0–3.0	5
	S2 Credentials or keys	196	5.79%	1.5–3.0	32
	S3 Security vulnerability	26	0.77%	1.5–3.0	11
	S4 Business strategy	235	6.94%	1.5–3.0	54
	S5 Personnel or administration	97	2.87%	1.5–3.0	33
	S6 Customer or partner information	189	5.58%	1.0–2.5	46
	S7 Internal finance	268	7.92%	1.0–3.0	52
	Subtotal	**1017**	**30.04%**	–	–
Total	–	**3385**	**100.00%**	–	–

Table 4: Statistics of sensitivity-unit annotations by privacy domain and fine-grained category. The weight column indicates the assigned risk-weight range for units in each category, and # Tasks denotes the number of tasks containing at least one unit from that category.

standardized workspace structure, and required resources are placed in deterministic locations. We also standardize the tool and skill settings used by each task so that task actions can be executed and logged consistently by the harness. This step ensures that tasks from different sources can follow the same agent loop and logging mechanism.

For each task, we adapt its original verification logic into a success checker compatible with ACE-BENCH. Depending on the task, the checker may involve rule-based validation, file or application-state auditing, command-output inspection, or rubric-based LLM-as-a-judge verification. The adapted checker produces a completion score in $[0, 1]$, which is used for task-utility evaluation.

Finally, we validate that adaptation does not substantially change task semantics or difficulty. For each task, we run both the original and adapted versions using Qwen3.5-9B and compute the average completion score over three runs. Considering the variability of agent execution, tasks whose absolute average-score difference exceeds 0.3 are discarded. This helps ensure that the standardized instances preserve the intended task objective while remaining compatible with the ACE-BENCH harness.

A.4 Privacy Annotation

This section provides additional details on the privacy annotation process used in ACE-BENCH. Given the constructed task suite, we first identify tasks whose local workspace or execution context already contains privacy-relevant information, such as files, tool outputs, command outputs, or application states. For these tasks, we preserve the original content and annotate the corresponding spans as sensitivity units.

For tasks without explicit sensitive information, we examine whether synthetic sensitive content can be inserted naturally without changing the task objective, execution workflow, or success criteria. When suitable insertion points exist, such as user profiles, internal documents, account records, configuration files, or task-specific notes, we use LLM-assisted synthesis to generate realistic but fully synthetic sensitive content and insert it into these locations. For example, in an office-productivity task that requires summarizing an internal meeting note, synthetic employee records or compensation-related details can be added to the workspace document without changing the required workflow. All injected content is synthetic and does not reuse real private data.

We then annotate all privacy-relevant content as sensitivity units. Each unit records its *value*, *source location*, *privacy category*, and *risk weight*. Table 4 reports the privacy taxonomy, annotation statistics, risk-weight ranges, and task coverage. Our taxonomy contains two domains, *Personal Sensitive Information* and *Organizational Confidential Information*, with 12 fine-grained categories in total. The category taxonomy is derived from prior taxonomies (Ngong et al., 2026; Shen et al., 2025), and further extended to organizational confidential information following WIPO guidance on trade secrets (WIPO, 2024). Risk weights are assigned to distinguish leakage events with different potential

harms, so that exposing highly sensitive units contributes more to the privacy penalty than exposing low-risk units.

A.5 Implementation Details of Evaluated Strategies

This section provides additional implementation details for the strategies evaluated in ACE-BENCH.

Sketch-Guided Edge Execution. Following the implementation described by Zhang et al. (2024), Sketch-Guided Edge Execution instantiates pipeline-style collaboration. Before the main execution begins, the cloud-side model receives the initial task context and generates a high-level execution sketch using the following prompt:

Prompt for Sketch Generation

Role: You are an organizer responsible for providing only the skeleton of how an executor agent should complete the task.
Task: Provide the skeleton as a list of points. Each skeleton point should be concise, using fewer than 10 words.

After this upfront stage, all subsequent agent steps are executed by the edge-side model conditioned on the generated sketch.

Task-Level Routing. Task-Level Routing makes a one-time model-use decision before execution and assigns the entire trajectory to either M_e or M_c. We instantiate it with the matrix-factorization-based router from RouteLLM (Ong et al., 2024). To adapt the router to agentic tasks, we train it using routing data from Li et al. (2026) and trajectory-level preference labels derived from validation-set completion scores of the evaluated models. Following prior routing settings (Panda et al., 2025; Ong et al., 2024), we calibrate the routing threshold on the validation set so that approximately 25% of tasks are routed to the cloud, and apply the same threshold during evaluation.

Step-Level Routing. Step-Level Routing uses an edge-first escalation strategy. At each step, M_e first generates an output, and we compute an uncertainty score using the entropy of its next reasoning-token distribution, following GlimpRouter (Zeng et al., 2026b). If the uncertainty score exceeds a threshold, the step is escalated to M_c; otherwise, the edge-side output is used directly. For fair comparison, we tune the threshold on the validation set so that roughly 25% of steps are escalated to the cloud, and apply the same threshold during evaluation.

Adaptive Cloud Assistance. We instantiate assistive collaboration through an advisor tool backed by the cloud-side model. The edge-side model can invoke the advisor during execution; when invoked, the current conversation history, including previous tool calls and observations, is forwarded to the cloud-side model. The cloud-side model then returns high-level advice, and the edge-side model continues execution conditioned on this advice. The advisor instruction encourages invocation before committing to a substantive approach, when the agent is stuck or considering a change of approach, and before declaring the task complete. Full prompts are included in the released implementation.

A.6 Pricing and FLOPs Estimation

This section describes how we estimate cloud-side monetary cost and edge-side computation in ACE-BENCH.

For each cloud invocation, the harness records raw input tokens, cache-read input tokens, and output tokens. We aggregate these token counts over all cloud calls and compute cost using the corresponding OpenRouter prices at the time of our experiments:

$$C_{\text{cloud}} = \frac{p_{\text{raw}} T_{\text{raw}} + p_{\text{cache}} T_{\text{cache}} + p_{\text{out}} T_{\text{out}}}{10^6}, \quad (7)$$

where T_{raw}, T_{cache}, and T_{out} are aggregated token counts, and p_{raw}, p_{cache}, and p_{out} are prices per million tokens. Table 5 reports the pricing used in our experiments.

Cloud Model	Input	Cache Input	Output
GPT-5.4	2.50	0.2500	15.00
DeepSeek-v4-Flash	0.14	0.0028	0.28
DeepSeek-v4-Pro	1.74	0.0145	3.48
GLM-5.1	0.98	0.1820	3.08

Table 5: Cloud-side pricing used for monetary cost estimation. Prices are reported in USD per million tokens.

For edge-side execution, we report estimated FLOPs as a reference for local computation burden. Following Kaplan et al. (2020), we estimate inference FLOPs as:

$$\text{FLOPs}_{\text{edge}} = 2 \cdot N \cdot T_{\text{edge}}, \quad (8)$$

where N is the number of model parameters and T_{edge} is the total number of tokens processed by the edge-side model across the execution trace.