SAR-ARENA: BENCHMARKING AGENT-DRIVEN RESEARCH ON AI COMPUTING SYSTEMS AND ARCHITECTURE

Anonymous authors
Paper under double-blind review

ABSTRACT

Research agents are starting to tackle systems and architecture problems as AI workloads and computing platforms evolve rapidly. Such work begins with workload-grounded bottleneck analysis, couples cross-stack mechanism design with engineering implementation, and requires long-horizon, measurement-driven optimization of quantitative system objectives; yet aggregate implementation scores reveal little about how agents make these decisions. We introduce SAR-Arena, a benchmark of ten recent, underexplored workload–platform problems for evaluating research in AI computing systems and architecture. Each task fixes its workload distribution, quality-constrained objective, and baseline while leaving the optimization mechanism open, and offers three research settings that supply either the problem alone, an expert bottleneck analysis, or an expert method proposal. SAR-Arena pairs rubric-based judgments of submitted diagnoses and methods with task-specific quantitative evaluators of runnable solutions, so intermediate reasoning and final system behavior can be examined together. We evaluate six model–harness configurations in 180 runs with eight-hour engineering budgets and extend 18 full-cycle trajectories to 24 engineering hours. Diagnosis and engineering leadership differ across configurations, while expert proposals produce heterogeneous changes in engineering outcomes. Longer runs yield task-dependent gains. A nine-category trajectory audit identifies recurring measurement and design problems, and engineering failures during expert-proposal implementation. Together, these evaluations connect stage-end research artifacts with quality-constrained system outcomes and expose differences that a single aggregate score would obscure.

1 INTRODUCTION

AI agents can now plan experiments, write code, and analyze results (Lu et al., 2024; Yamada et al., 2025). Whether these activities constitute research depends on how the agent connects them: an initial hypothesis should determine what to measure, and the measurements should change what is built next. Systems and architecture research (SAR) makes this connection observable. Given a workload or workload distribution, a researcher changes part of the hardware–software stack to improve a measurable objective under correctness, quality, and resource constraints. The final artifact must satisfy the complete, quality-constrained system objective that motivated the diagnosis, including valid outputs and all required processing costs.

This is an increasingly important setting for AI researchers. Language, visual-generation, and interactive models evolve alongside GPUs, NPUs, and near-memory platforms; a new workload can move the dominant cost to a layer that an established optimization does not address. Yet a recent workload alone does not create a research task. The interesting cases retain competing, technically plausible mechanisms after the workload, platform, and strongest usable baseline are specified. They also require accounting for the complete system: a faster compute unit may shift traffic or synchronization elsewhere, and a faster service may be invalid if it omits work or degrades quality.

SAR-Arena therefore focuses on three connected research responsibilities (Figure 1). A bottleneck analysis must explain which costs limit the objective across the relevant workload distribution, rather

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.

than import a performance number from a different setting. A sound diagnosis still leaves open how to design a mechanism whose data layout, scheduling, numerical behavior, and hardware interfaces can coexist. Engineering then tests that mechanism over successive versions, including cases in which an apparent local gain exposes a new bottleneck. These dependencies make the research semi-open—the problem contract is fixed but the mechanism is not—and make sustained optimization more informative than a single implementation attempt.

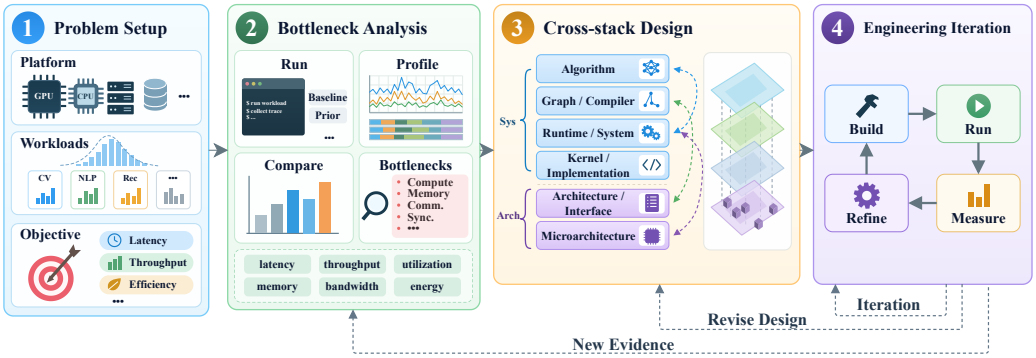


Figure 1: SAR workflow. A workload, platform, and measurable objective define the problem; diagnosis and cross-stack design guide an experimental build–measure–refine loop, whose evidence can revise earlier decisions.

We introduce **SAR-Arena**, ten recent AI systems and architecture problems selected for unresolved optimization choices. Each task defines a workload, objective, allowed changes, constraints, and a baseline-anchored engineering outcome. Three matched settings vary the upstream information supplied to the agent: **Full-Cycle Research** starts from the problem, **Guided Research** supplies an expert bottleneck analysis, and **Engineering Optimization** supplies an expert proposal. Stage-end analyses and proposals are assessed separately from the submitted implementation. This design jointly observes stage-end artifacts and end-to-end outcomes under different expert inputs and research responsibilities.

Across six model–harness configurations and ten tasks, the resulting 180 runs show that diagnosis and engineering leadership differ, even though method-score rankings align with engineering rankings. Expert inputs produce heterogeneous changes in engineering outcomes. Eighteen extended trajectories examine continued optimization, and a stage-specific audit characterizes observed process failures.

Our contributions are:

- We define ten SAR tasks in which recent workloads meet explicit system objectives, competing optimization choices, and task-specific validity checks.
- We compare three assignments of diagnosis, method design, and engineering work on each task using stage-end assessments and a shared downstream engineering scale.
- We analyze stage-level outcomes, matched guidance comparisons, longer engineering trajectories, and observed process failures to characterize how submitted research artifacts and system outcomes vary across tasks and research settings.

2 Related Work

Research-agent evaluation. SWE-bench and MLE-bench ground coding agents in executable software and ML tasks (Jimenez et al., 2024; Chan et al., 2025). PaperBench tests research replication, while InnovatorBench already evaluates open-ended LLM research with runnable outputs, long experiments, and failure analysis (Starace et al., 2025; Wu et al., 2025). SWE-bench Science preserves the engineering context while varying scientific guidance, finding that more information need not improve exact repair (Xu et al., 2026). SAR-Arena likewise treats a research artifact as something that

must work, but fixes a systems workload and complete objective, varies the assigned upstream work on each task, and evaluates the resulting implementation under the same task contract. Its guidance comparisons are conditional workflow comparisons, not pure measures of independent abilities.

Infrastructure optimization. Kernel agents use hardware feedback to refine implementations (Zhang et al., 2025a;b). FlashInfer-Bench already includes serving traces and integration checks, while CAKE couples kernel search to a compiler representation and verification (Xing et al., 2026; Ye et al., 2026b). ArchAgent searches for new architecture mechanisms and studies simulator exploitation; LUMINA tests bottleneck analysis, performance/area prediction, and GPU design choices, chiefly through separate questions and search tasks (Gupta et al., 2026; Zhang et al., 2026). SAR-Arena's unit is a heterogeneous, executable research project with a common workload-level endpoint under three assignments of upstream responsibility. Neither mechanism search nor system-level validation alone is new here; the comparison asks whether an agent's own diagnosis and proposal survive that endpoint when expert information is progressively supplied. Appendix A expands the comparison.

3 SAR-ARENA

3.1 TASK DESIGN AND CONSTRUCTION

SAR-Arena defines ten environments around AI workloads and hardware platforms available in the literature by August 15, 2026. Each fixes a workload distribution and system objective but leaves several plausible optimization mechanisms open. We selected problems for which the limiting resource is not determined by a model name or a published speedup: candidates must admit a workload-specific diagnosis, consequential choices across stack layers, and a result that can be checked without privileging one proposed solution. Figure 2 gives the design principles and construction sequence; Table 1 lists the environments.

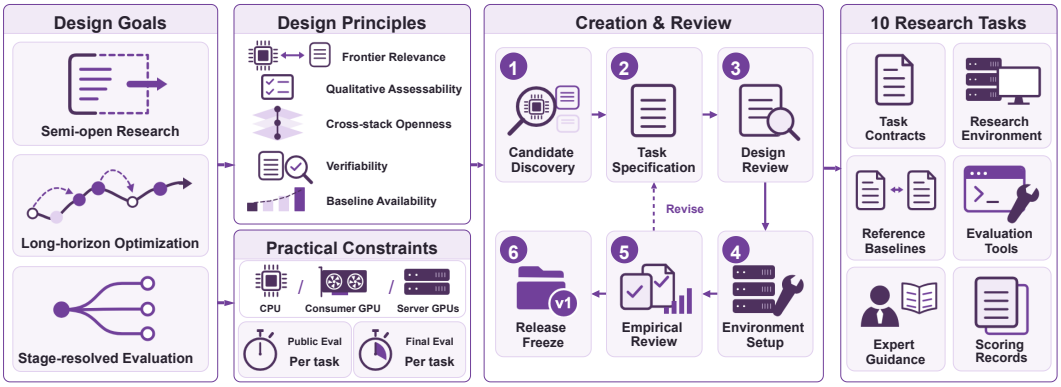


Figure 2: SAR-Arena task construction. Six reviewable stages turn a recent research opportunity into a versioned task package containing a contract, workspace, baselines, evaluator, expert guidance, and scoring records.

Research content and validity. A task is qualitatively assessable when measurements and domain constraints can support or refute an analysis or proposal before the final score is known. The expert solution is therefore an evidential comparator, not a required implementation. A strong, pre-measured baseline supplies a meaningful improvement threshold, while independent quality and complete-cost checks prevent apparent gains from omitting work. In the speech task, event-to-response latency is meaningful only when responses are playable and causal; in the tensor-core task, isolated arithmetic throughput is insufficient if the full trace loses useful throughput to data movement or conversion. These checks leave room for alternative mechanisms rather than prescribing an optimization recipe.

Candidate tasks are refined with domain reviewers until the workload, editable boundary, baseline, and evaluator express the same objective. Review rejects a task when the research choice collapses to a known implementation, an input region essential to the objective is absent, or a shortcut can score without delivering the required behavior. The environment packages an executable starting

point, public checks, and expert guidance with a versioned task contract. The three settings vary the upstream artifacts supplied under that contract; Section 3.2 defines the stage and outcome scores.

The portfolio deliberately varies what counts as an output: a complete answer to a streaming-video query, a finished coding-agent session, playable speech or audiovisual media, an executed control action, or a numerically valid GEMM. This changes both the bottleneck search and the shortcut an evaluator must rule out. The detailed cards in Appendix B.1 state task boundaries and admissible alternatives to the expert design; a task is not a request to reproduce the expert's method.

Table 1: Ten evaluated environments in task-card order. Hardware labels distinguish direct measurement from modeled components; choices are illustrative, not prescribed. Arrows show objective direction.

Environment	Hardware Type	Workload	Open choice	Primary objective
Streaming video understanding	RTX 5090 + HBF model	Flash-VStream-Qwen-7b	Evidence granularity and tier placement	Complete-answer p95 latency ↓
Coding-agent serving	8×H100 (measured)	Qwen3-Coder-Next	State retention and restoration	Qualified output tokens/s ↑
Full-duplex speech	RTX 5090 (measured)	MiniCPM-o 4.5, audio-only	Causal chunks and deadline scheduling	Event-to-playable-response p95 ↓
Audiovisual generation	RTX 5090 (measured)	Ovi 720x720_5s	Solver, residency, and full-file pipeline	Prompt-to-playable-file time ↓
Interactive world model	RTX 5090 (measured)	ReWorld-5B + Wan2.2-TI2V-5B	Action/state and frame pipeline	Action-to-displayed-frame p95 ↓
MoE CPU–GPU mapping	RTX 5090 + CPU	DeepSeek-V4-Flash-0731	Expert placement and migration	Qualified output tokens/s ↑
VLA closed-loop control	RTX 5090 + CPU simulation	π0.5-LIBERO	Solver and control-deadline mapping	Observation-to-action p95 ↓
CPU-only MoE service	NUMA CPU (measured)	Nemotron 3 Super, NVFP4	Format, NUMA placement, and kernels	Qualified output tokens/s ↑
NVFP4 tensor core	RTX 5090 + RTL model	Gemma 4 31B IT, NVFP4	Arithmetic and complete data supply	Useful TOPS/mm^2 ↑
Retrieval sparse attention	RTX 5090 + PIM model	DeepSeek-V4-Flash-0731	Indexer and selected-KV mapping	Goodput / platform TDP ↑

3.2 Evaluating Research Stages

SAR-Arena evaluates three stage-end artifacts: a bottleneck analysis, a method proposal, and a nominated implementation. Full-Cycle Research produces all three; Guided Research receives the expert analysis and produces the latter two; Engineering Optimization starts from an expert proposal and produces an implementation (Figure 3). Task-specific prompts define each stage's work and submission boundary. The conditions preserve the workload and downstream evaluator but change both the upstream information and the work assigned to the agent. Their differences therefore describe conditional research performance, not isolated cognitive abilities.

Evaluation boundary. Each setting is a separate agent run. The agent decides when analysis or design is complete and submits a report in the prescribed format; that submission freezes the stage artifact and marks the transition to the next stage. During analysis and design, agents may write code and run experiments; files, measurements, and workspace state carry into engineering. Guided Research receives an expert diagnosis, whereas Engineering Optimization receives only an expert proposal, not a separate expert diagnosis. The agent may revise that proposal during engineering but records material departures in its final nomination. Analysis and design have no hard time or token cap, and the engineering clock starts only on entry into implementation; eight hours therefore does not equalize total research time or engineering starting state across settings. The agent nominates one final implementation and configuration. Public measurements may guide iteration, but offline snapshots and independent final verification are not returned to the agent. Stage scores assess the submitted analysis or proposal, not a later reconstruction from implementation success.

Analysis and design scores. The diagnosis rubric weights in-domain literature (10 points), profiler design (20), workload coverage (20), multi-metric evaluation (20), and report utility (30). The

method rubric weights cross-domain literature (10), cross-stack design (30), novelty (30), and executable specification (30). The full evidence checklist contains 32 diagnosis criteria and 25 method criteria, following PaperBench's granular requirement structure (Starace et al., 2025). GPT-5.6 Pro compares the frozen artifact with an expert reference and eligible papers available by August 15, 2026, while allowing supported alternatives; the compressed trajectory corroborates evidence but cannot supply a missing argument. Three judgments of the same artifact are averaged. Appendix B specifies every criterion and decision rule, together with the scope of expert calibration.

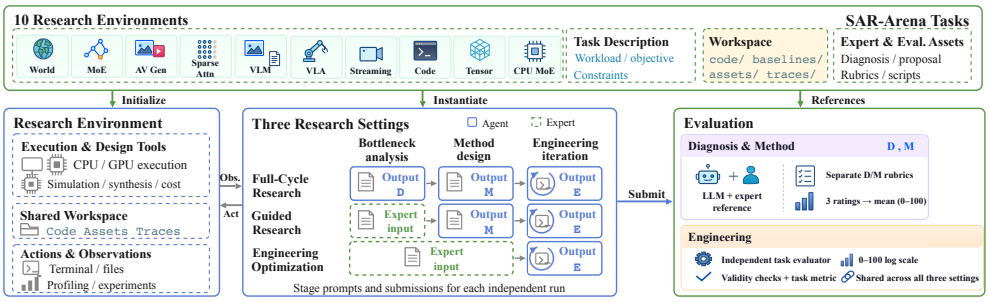


Figure 3: Three research settings assign different upstream work while preserving each task's final engineering evaluator. Guided Research receives an expert diagnosis; Engineering Optimization receives only an expert proposal and has no diagnosis stage. Expert input is distinguished from scored agent output.

Engineering score. Each task has its own primary metric and validity gates. The final evaluator measures both the strongest pre-measured baseline and each submission five times under the task's measurement protocol, taking the geometric mean of the primary metric. Missing runnable code, a violated constraint, or failure to improve on the baseline receives $E = 0$. At a fixed task and engineering budget, let q be the oriented improvement ratio relative to that baseline and $q_\star$ the best valid ratio across all evaluated models and settings. Then $E = 100 \log q / \log q_\star$ for $1 < q \leq q_\star$; if no result beats the baseline, all engineering scores are zero. The common task anchor makes identical engineering outcomes comparable across settings, but it is a descriptive scale for this evaluated cohort: a new record would change old scores if the upper anchor were recomputed. A published benchmark version must therefore retain both raw task metrics and its frozen anchor; a later result above that anchor may exceed 100 on the old scale, whereas recomputing the anchor creates a new score version. The five-repeat rule does not establish statistical separation for a marginally positive difference. Raw metrics retain their physical meaning; the normalized E does not represent a universal percentage of research ability. Appendix B states the boundary cases.

Let $D, M, E \in [0, 100]$ denote analysis, method, and engineering scores. We report

$$S_{\mathrm{F}} = 0.20D + 0.20M + 0.60E, \qquad S_{\mathrm{G}} = 0.25M + 0.75E, \qquad S_{\mathrm{O}} = E.$$

The weights are fixed across tasks. A zero engineering score does not erase upstream scores. The weighted totals summarize different assigned responsibilities and are not compared as if they measured the same construct.

3.3 Task Protocol and Examples

The task contract identifies the target workload and hardware, editable system boundary, primary metric, quality gates, and all work charged to that metric. Agents receive executable assets and public measurements; a separate evaluator checks the nominated artifact on its declared coverage. Public feedback can guide search but cannot establish final validity. Streaming-video memory combines measured CPU/GPU/storage behavior with a separate HBF model; retrieval sparse attention likewise separates measured service components from near-memory modeling. NVFP4 designs pass numerical and whole-model checks before RTL-derived area and timing are scored. Coding-agent serving uses eight H100 GPUs; other service tasks use the RTX 5090 host, with CPU-only MoE inference confined to its NUMA CPUs. Modeled hardware claims are labeled as such, never as silicon measurements.

Software example: Full-duplex speech. At fixed concurrency, the MiniCPM-o 4.5 task minimizes p95 delay from a response-required speech event to the first semantically valid, playable output on

one RTX 5090. This endpoint charges encoding, language modeling, synthesis, decoding, queues, transport, and client buffering. Independent behavior labels exclude legitimate silence; missing responses contribute infinite latency, while a separate completion gate prevents a finite p95 from masking occasional omissions. Playback continuity, interruption handling, audio quality, and long-session memory remain validity gates. Figure 4 details the task prompts, research assets, expert guidance, and evaluation specification, including unresolved checks in the design snapshot. Appendix D presents author-side diagnostic observations and candidate mechanisms for this task and audiovisual generation, with their measurement limits.

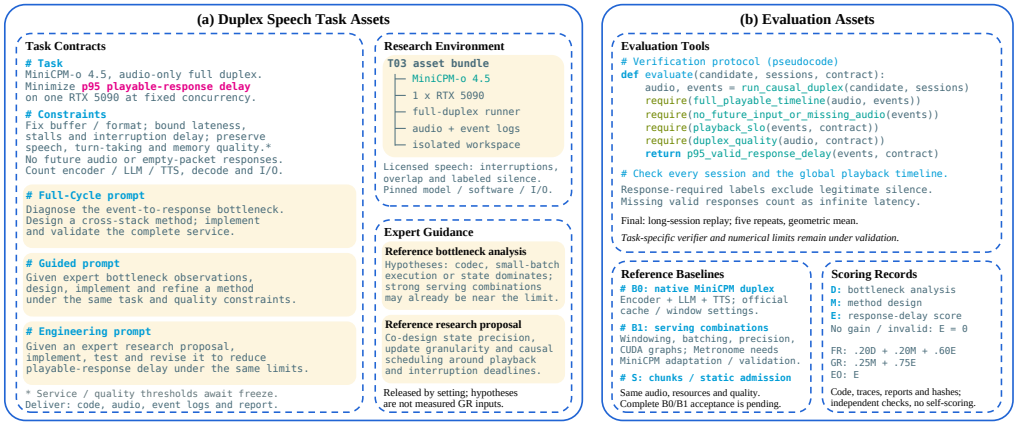


Figure 4: Full-duplex speech task assets and evaluation protocol. The detailed package contains three setting-specific prompt excerpts, two expert reference assets, and an independent evaluation specification. The objective is p95 event-to-playable-response delay at fixed concurrency. Annotations retain pending thresholds, baseline acceptance, and unmeasured expert hypotheses from the design snapshot; the verifier is protocol pseudocode.

Hardware example: NVFP4 tensor core. The agent chooses arithmetic organization, accumulation, buffering, and mapping for Gemma 4 31B IT linear layers under fixed interfaces. The objective is useful trace-weighted throughput per silicon area, not isolated peak TOPS: scaling, conversion, data supply, control, and writeback consume area and cycles, while padded operations add no useful work. Numerical and model-quality checks bound the design. A locally faster arithmetic array can therefore lower the complete trace-level core objective if its supporting data path or area cost dominates.

The scored unit is a complete transaction. In streaming video understanding (T01), frames arrive before and during questioning. The p95 endpoint is delivery of a complete answer, with current-scene and retrospective accuracy checked separately; an answer that depends on a future frame or omits slow historical queries is ineligible. Coding-agent serving (T02) instead counts qualified output tokens only from *complete* multi-turn sessions under a fixed arrival window. Its replay preserves tool waits, context compaction, and branches, while a separate free-running check tests whether state optimizations retain coding quality. Interactive world modeling (T05) likewise times an external action through its first *displayed* causally affected frame, rather than ending at a latent state. These boundaries make encoding, movement, queues, decoding, and failed requests part of the optimization problem, rather than overhead that can disappear from a component benchmark.

Distinct evidence paths constrain architectural claims. T01 couples measured CPU, GPU, DRAM, and NVMe behavior to an HBF sensitivity model; no physical HBF throughput is claimed. The two DeepSeek-V4-Flash-0731 tasks examine different mechanisms: T06 maps MoE experts across the CPU–RTX 5090 memory hierarchy and scores service-qualified output goodput, whereas T10 keeps the query-dependent sparse-attention indexer, selected KV, transfers, and full decode in scope and divides modeled SLO-qualified goodput by a frozen *platform* TDP. A GPU-only T10 design remains admissible if it meets the same contract. For T09, RTL-derived area and timing share one process and EDA convention, while an independent interpreter and full-model quality check guard NVFP4 numerical semantics. Across these cases, measured components can calibrate or bound a model, but they do not convert an unbuilt architecture into a measured device result.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Agents and resources. We evaluate six single-agent model–harness configurations at their highest available reasoning setting: Claude Opus 5 / Claude Code, GPT-5.6-Sol / Codex, Kimi-K3 / Claude Code, GLM-5.3 / Claude Code, DeepSeek-V4-Pro-0813 / Claude Code, and DeepSeek-V4-Flash-0731 / Claude Code. Subagent delegation is disabled; paper retrieval is restricted to work published by August 15, 2026. These are model–harness systems, not isolated foundation models. Coding-agent serving uses eight 80-GB H100-SXM accelerators. The other nine tasks use the RTX 5090 host, with its 32-GB card, two 24-core Xeon Gold 6542Y CPUs, and 503.6 GiB RAM; the CPU-only task excludes GPU execution from its scored path. HBF, near-memory, and RTL quantities remain modeled rather than device measurements.

Runs and measurements. Each configuration attempts ten environments once in each of Full-Cycle Research (F), Guided Research (G), and Engineering Optimization (O), yielding 180 runs. The eight-hour clock begins at engineering entry; analysis and design have no hard time or token cap, and independent verification is excluded. Per task we record O_E; $G_M, G_E, G_{\text{total}}$; and $F_D, F_M, F_E, F_{\text{total}}$. Table 2 gives equal-weight task means. We also report total agent tokens by performed stage, including input and output but excluding judge and evaluator calls (Appendix C). On retrieval sparse attention, NVFP4 tensor core, and coding-agent serving, all six F trajectories continue to 24 engineering hours. Four-hour snapshots are evaluated offline without feedback to the agent or charge against its research clock.

4.2 STAGE AND ENGINEERING OUTCOMES

Table 2: Mean scores over ten tasks (higher is better). All models use Claude Code except GPT-5.6-Sol, which uses Codex; each model–task–setting has one run.

	Engineering Optimization	Guided Research			Full-Cycle Research			
Model	Eng.	Method	Eng.	Total	Diag.	Method	Eng.	Total
Claude Opus 5	88.5	**92.8**	**91.3**	**91.7**	83.8	88.9	**87.4**	**87.0**
GPT-5.6-Sol	**91.3**	87.7	81.7	83.2	**88.8**	80.6	79.9	81.8
Kimi-K3	88.6	78.9	62.7	66.7	71.0	64.5	49.0	56.5
GLM-5.3	74.1	72.3	51.8	56.9	65.8	58.0	36.2	46.5
DeepSeek-V4-Pro-0813	66.4	68.7	41.5	48.3	58.2	49.6	16.3	31.3
DeepSeek-V4-Flash-0731	0.8	45.8	0.1	11.5	37.2	26.4	0.0	12.7

Diagnosis and engineering leadership differ. Claude Opus 5 leads F and G on mean engineering and within-setting total score; GPT-5.6-Sol leads O on engineering. GPT-5.6-Sol has the highest F diagnosis score (88.8), whereas Claude Opus 5 leads F method (88.9 versus 80.6) and engineering scores (87.4 versus 79.9). This leadership difference does not imply a general separation between stage ratings and implementation success: across the 60 F runs, pooled Pearson correlations are $r(D, E) = 0.919$ and $r(M, E) = 0.913$, and model rankings by mean method and engineering scores agree in both F and G. These associations mix task and configuration differences and do not establish causality. Valid implementations exceed baseline in 46/60 F, 48/60 G, and 49/60 O runs; Appendix C checks the relationship to high stage scores.

Table 3: Unique top-three engineering placements. Each cell lists first/second/third counts; ties and zero scores receive no place.

	Engineering Optimization	Guided Research	Full-Cycle Research
Model	1/2/3	1/2/3	1/2/3
Claude Opus 5	3/1/2	7/3/0	8/2/0
GPT-5.6-Sol	3/5/2	3/6/1	2/8/0
Kimi-K3	2/2/3	0/1/7	0/0/8
GLM-5.3	1/1/2	0/0/2	0/0/2
DeepSeek-V4-Pro-0813	1/0/0	0/0/0	0/0/0
DeepSeek-V4-Flash-0731	0/0/0	0/0/0	0/0/0

Composition of guidance-associated gains. F and O engineering scores use the same anchor on each task. Among 60 matched model–task pairs, 44 score above zero in both conditions, five move from zero to positive, two from positive to zero, and nine remain at zero (Table 5 in Appendix C). The mean O–F increase is 23.47 points: the both-positive pairs contribute 18.58 points to that overall mean, zero-to-positive pairs 5.64, and positive-to-zero pairs -0.75. The largest contribution comes from a group that already contains 44 of 60 pairs; it does not imply that expert input cannot help non-improving runs. Five of the 14 F zero-score pairs become positive in O. A zero includes valid results no better than baseline, so these transitions do not alone identify correctness failures. Guidance is also non-monotone: on MoE CPU–GPU mapping the six-model mean is 61.7 in G and 60.7 in O, and Claude Opus 5 averages 91.3 in G versus 88.5 in O. With one run per cell, these are observed differences under changed information *and* assigned work, not isolated effects of diagnosis or design ability.

Every configuration's mean method score is higher in G than in F, but method-score gain is not a surrogate for engineering gain. DeepSeek-V4-Flash-0731 improves its method score by 19.5 points and still has near-zero G engineering. The placement counts add a different view: Claude Opus 5 wins eight F tasks and seven G tasks, while Claude Opus 5 and GPT-5.6-Sol each win three O tasks; GPT-5.6-Sol leads O by mean score rather than an exclusive lead in first-place breadth (Table 3). Appendix C reports total tokens by performed stage. Pooled over all three settings, mean engineering-stage token volume ranges from 11.98M for GPT-5.6-Sol to 25.61M for DeepSeek-V4-Flash-0731; the latter has engineering scores near zero. These aggregates include repeated context, use different model tokenizers, and do not identify distinct experiments or condition-specific reasoning efficiency.

The per-task means also prevent the aggregate from becoming a universal guidance claim. Engineering Optimization, Guided Research, and Full-Cycle Research score 74.4, 62.1, and 49.3 on streaming video understanding, but 60.7, 61.7, and 50.2 on MoE CPU–GPU mapping. On CPU-only MoE service, the corresponding means are 61.9, 57.1, and 34.4 (Table 8 in Appendix C). The same assignment change can therefore have different directions and magnitudes across workloads. Within a task the scores share an anchor; across tasks, they summarize different physical objectives and should not be read as equal percentage changes in latency, goodput, or area efficiency.

4.3 Test-Time Scaling

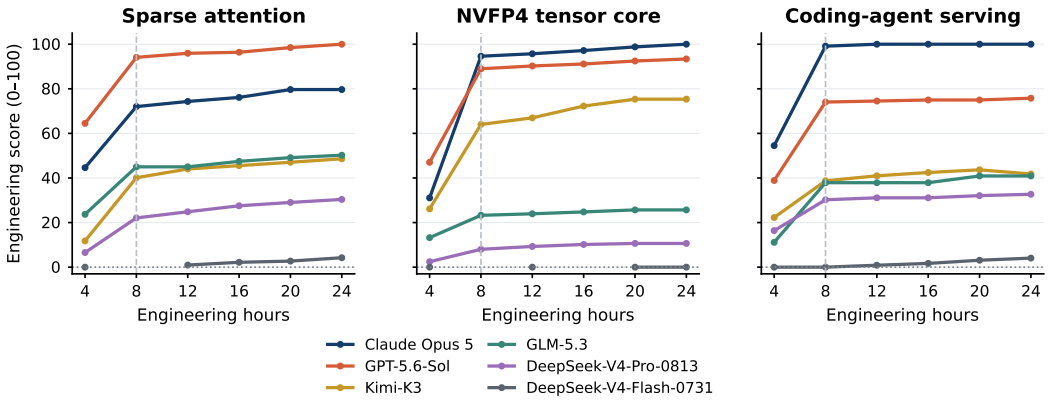


Figure 5: Full-Cycle Research snapshots for retrieval sparse attention, NVFP4 tensor core, and coding-agent serving (left to right), shown as 0–100 engineering scores. Each point evaluates the artifact nominated at that checkpoint; offline scores are not returned to the agent. For each task, the best valid result among all six trajectories and six checkpoints defines 100; the fixed baseline defines 0 (dotted line). These 24-hour anchors differ from the main table's eight-hour anchors. The dashed line marks eight hours; invalid snapshots are omitted.

Figure 5 follows each nominated artifact on 18 continuing F trajectories. Fifteen beat baseline at eight hours and 17 at 24 hours. For the 15 already above baseline at eight hours, the median relative change in oriented raw utility, $q_{24}/q_8 - 1$, is 3.0%; task medians are 7.1% for retrieval sparse attention, 2.4% for NVFP4 tensor core, and 2.5% for coding-agent serving. The figure uses a fixed

per-task 24-hour anchor, so its eight-hour score is not numerically interchangeable with Table 2's eight-hour score; the raw utility ratios are comparable. These are repeated snapshots of continuing runs, not independently budgeted four-hour trials.

At 24 hours, the best observed F result exceeds the best eight-hour G result on two tasks but remains below it on NVFP4 tensor core. Those best-of-condition artifacts can come from different models; this contrast describes task-dependent trajectories rather than testing whether time substitutes for expert guidance. Three selected tasks do not establish a suite-wide saturation time.

The additional time changes validity as well as performance. DeepSeek-V4-Flash-0731 changes from invalid to baseline-beating on retrieval sparse attention, from invalid to baseline-matching on NVFP4 tensor core, and from baseline-matching to baseline-beating on coding-agent serving. All 15 eight-hour baseline beaters remain above baseline at 24 hours. Each plotted point evaluates the nominated checkpoint, not the best previous candidate: overwriting a working design can therefore make the submitted trajectory non-monotone.

4.4 Failure Study

Nine categories were elicited from ten randomly sampled compressed trajectories and result files. GPT-5.6 Pro then applied the category checklists to all 180 runs, including those ten; model identities were hidden but final scores were visible. Each applicable run contributes at most one count per category (Table 4; Appendix E). The pools differ: B codes cover measurement or bottleneck reasoning appearing anywhere in all 30 runs per model, including G/O runs without a submitted diagnosis; D codes cover F/G design work (20); E codes cover O engineering (10). For each category, the audit first checks whether the trajectory actually presented a trigger opportunity, then requires a trace-linked event; an inapplicable or insufficiently evidenced case is not a verified absence. Thus the cells are event counts, not opportunity-adjusted failure rates, and the engineering findings concern implementation of expert proposals in O, not engineering in F/G. Events repaired before submission remain observed events, not necessarily causes of the final score.

Table 4: Runs with coded failures. B codes are checked for bottleneck-related behavior anywhere in F/G/O (30 runs per model); D codes cover F/G (20), and E codes cover O only (10). Each category is counted at most once per run where applicable.

	Bottleneck-related			Method design		Engineering in O			
Model	B1	B2	B3	D1	D2	E1	E2	E3	E4
Claude Opus 5	0	2	1	0	1	1	0	1	0
GPT-5.6-Sol	0	1	1	0	1	1	0	0	0
Kimi-K3	1	3	2	2	3	1	1	1	0
GLM-5.3	1	4	3	3	3	1	1	1	0
DeepSeek-V4-Pro-0813	1	5	3	4	3	2	1	2	1
DeepSeek-V4-Flash-0731	0	5	4	5	1	0	0	3	1

Incomplete workload coverage (B2) appears in 20/180 audited runs and faulty profiling (B3) in 14/180: a numerical diagnosis can therefore be precise yet describe the wrong operating region or quantity. Among the 120 F/G design runs, 14 contain a concept without an executable decision mechanism (D1) and 12 contain cross-layer components that cannot work together (D2). In the 60 O engineering runs, eight show nonincremental debugging (E3). These counts identify different failure boundaries, not a demonstrated causal chain between B, D, and E or nine additive causes of score loss. Opportunity matters: DeepSeek-V4-Flash-0731 has no E1 count, but only two of its O runs presented a genuine trade-off that could trigger E1. The full code definitions and exclusions are in Appendix E.

5 Conclusion

SAR-Arena jointly evaluates workload-grounded diagnoses, cross-stack proposals, and quality-constrained system outcomes under three research settings. The reported cohort shows differing diagnosis and engineering leaders, aligned method and engineering rankings, and heterogeneous benefits from expert inputs. Extended trajectories and stage-specific failure coding characterize continued optimization and expert-proposal implementation.

LLM USAGE

AI Assistance. We used large language models as assistive tools during manuscript preparation in two limited capacities. First, LLMs were used to improve the clarity, concision, grammar, and presentation of author-written text. Second, LLMs were used to assist literature retrieval and discovery, including formulating search queries and identifying potentially relevant prior work.

REFERENCES

Felix Arnold, Ryan Amaudruz, Dimitrios Tsaras, et al. RTLScout: Joint Agentic Code and Synthesis Optimization for Efficient Digital Circuits. *arXiv preprint arXiv:2606.06530*, 2026. URL https://arxiv.org/abs/2606.06530.

Xinzi Cao, Jianyang Zhai, Pengfei Li, et al. AscendKernelGen: A Systematic Study of LLM-Based Kernel Generation for Neural Processing Units. *arXiv preprint arXiv:2601.07160*, 2026. URL https://arxiv.org/abs/2601.07160.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, et al. MLE-bench: Evaluating Machine Learning Agents on Machine Learning Engineering. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.07095.

Che-Ming Chang, Prashanth Vijayaraghavan, Ashutosh Jadhav, et al. CODMAS: A Dialectic Multi-Agent Collaborative Framework for Structured RTL Optimization. *arXiv preprint arXiv:2603.17204*, 2026. URL https://arxiv.org/abs/2603.17204.

Wentao Chen, Jiace Zhu, Qi Fan, et al. CUDA-LLM: LLMs Can Write Efficient CUDA Kernels. *arXiv preprint arXiv:2506.09092*, 2025a. URL https://arxiv.org/abs/2506.09092.

Zhirong Chen, Kaiyan Chang, Zhuolin Li, et al. ChipSeek: Optimizing Verilog Generation via EDA-Integrated Reinforcement Learning. *arXiv preprint arXiv:2507.04736*, 2025b. URL https://arxiv.org/abs/2507.04736.

Fan Cui, Hongyuan Hou, and Zizhang Luo. HWE-Bench: Benchmarking LLM Agents on Real-World Hardware Bug Repair Tasks. *arXiv preprint arXiv:2604.14709*, 2026. URL https://arxiv.org/abs/2604.14709.

Stef Cuyckens, Mihaela Jivanescu, Jun Yin, et al. ARES: Adaptive Reasoning-Effort Steering for PPA- and Cost-Aware RTL Optimization with LLM Agents. *arXiv preprint arXiv:2607.27879*, 2026. URL https://arxiv.org/abs/2607.27879.

Weinan Dai, Hanlin Wu, Qiying Yu, et al. CUDA Agent: Large-Scale Agentic RL for High-Performance CUDA Kernel Generation. *arXiv preprint arXiv:2602.24286*, 2026. URL https://arxiv.org/abs/2602.24286.

Joshua H. Davis, Klaudiusz Rydzy, Srinivasan Ramesh, et al. KEET: Explaining Performance of GPU Kernels Using LLM Agents. *arXiv preprint arXiv:2605.04467*, 2026. URL https://arxiv.org/abs/2605.04467.

Juncheng Dong, Yang Yang, Tao Liu, et al. STARK: Strategic Team of Agents for Refining Kernels. *arXiv preprint arXiv:2510.16996*, 2025. URL https://arxiv.org/abs/2510.16996.

Weihua Du, Jingming Zhuo, Yixin Dong, et al. AdaExplore: Failure-Driven Adaptation and Diversity-Preserving Search for Efficient Kernel Generation. *arXiv preprint arXiv:2604.16625*, 2026. URL https://arxiv.org/abs/2604.16625.

Wenji Fang, Yao Lu, Shang Liu, et al. Dr. RTL: Autonomous Agentic RTL Optimization through Tool-Grounded Self-Improvement. *arXiv preprint arXiv:2604.14989*, 2026. URL https://arxiv.org/abs/2604.14989.

Ruijie Gao, Jirong Yang, Barry Lyu, et al. Rethinking Agentic Kernel Generation for Emerging Accelerators. *arXiv preprint arXiv:2608.00894*, 2026. URL https://arxiv.org/abs/2608.00894.

Amur Ghose, Andrew B. Kahng, Sayak Kundu, et al. ORFS-agent: Tool-Using Agents for Chip Design Optimization. *arXiv preprint arXiv:2506.08332*, 2025. URL https://arxiv.org/abs/2506.08332.

Amur Ghose, Junyeong Jang, Andrew B. Kahng, et al. Automated QoR improvement in OpenROAD with coding agents. *arXiv preprint arXiv:2601.06268*, 2026. URL https://arxiv.org/abs/2601.06268.

Raghav Gupta, Akanksha Jain, Abraham Gonzalez, et al. ArchAgent: Agentic AI-driven Computer Architecture Discovery. *arXiv preprint arXiv:2602.22425*, 2026. URL https://arxiv.org/abs/2602.22425.

Sina Heidari and Dimitrios S. Nikolopoulos. FACT: Compositional Kernel Synthesis with a Three-Stage Agentic Workflow. *arXiv preprint arXiv:2604.26666*, 2026. URL https://arxiv.org/abs/2604.26666.

Qijing Huang, Sana Damani, Zhifan Ye, et al. SOLAR: AI-Powered Speed-of-Light Performance Analysis. *arXiv preprint arXiv:2606.26383*, 2026. URL https://arxiv.org/abs/2606.26383.

Jaber Jaber and Osama Jaber. AutoKernel: Autonomous GPU Kernel Optimization via Iterative Agent-Driven Search. *arXiv preprint arXiv:2603.21331*, 2026a. URL https://arxiv.org/abs/2603.21331.

Jaber Jaber and Osama Jaber. AutoMegaKernel: A Statically-Checked Agent Harness for Self-Retargeting Megakernel Synthesis. *arXiv preprint arXiv:2606.09682*, 2026b. URL https://arxiv.org/abs/2606.09682.

Carlos E. Jimenez, John Yang, Alexander Wettig, et al. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.06770.

Shiyang Li, Zijian Zhang, Winson Chen, et al. StitchCUDA: An Automated Multi-Agents End-to-End GPU Programing Framework with Rubric-based Agentic Reinforcement Learning. *arXiv preprint arXiv:2603.02637*, 2026. URL https://arxiv.org/abs/2603.02637.

Fangzhou Liu, Peiyi Han, Jiawei Liu, et al. SynAct: A Reasoning-Acting Large Language Model Agent for Adaptive Synthesis Optimization. *arXiv preprint arXiv:2608.12751*, 2026. URL https://arxiv.org/abs/2608.12751.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery. *arXiv preprint arXiv:2408.06292*, 2024. URL https://arxiv.org/abs/2408.06292.

Haohui Mai, Xiaoyan Guo, Xiangyun Ding, et al. Ave: Guiding Agentic GPU Optimization Using Data-Flow Invariants. *arXiv preprint arXiv:2604.18616*, 2026. URL https://arxiv.org/abs/2604.18616.

Jiayi Nie, Haoran Wu, Yao Lai, et al. KernelCraft: Benchmarking for Agentic Close-to-Metal Kernel Generation on Emerging Hardware. *arXiv preprint arXiv:2603.08721*, 2026. URL https://arxiv.org/abs/2603.08721.

Yikang Ouyang, Yang Luo, Dongsheng Zuo, et al. Retrieve, Schedule, Reflect: LLM Agents for Chip QoR Optimization. *arXiv preprint arXiv:2603.13767*, 2026. URL https://arxiv.org/abs/2603.13767.

Heng Ping, Peiyu Zhang, Zhenkun Wang, et al. POET: Power-Oriented Evolutionary Tuning for LLM-Based RTL PPA Optimization. *arXiv preprint arXiv:2603.19333*, 2026. URL https://arxiv.org/abs/2603.19333.

Tara Saba, Zhiyang Chen, Jikai Jason Li, et al. CuTeGen: An LLM-Based Agentic Framework for Generation and Optimization of High-Performance GPU Kernels using CuTe. *arXiv preprint arXiv:2604.01489*, 2026. URL https://arxiv.org/abs/2604.01489.

Yijia Shao, Vinay Samuel, Yucheng Jiang, John Yang, and Diyi Yang. Collaborative Gym: A Framework for Enabling and Evaluating Human-Agent Collaboration. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2412.15701.

Zhengyuan Shi, Jingxin Wang, Tairan Cheng, et al. CktEvo: Repository-Level RTL Code Benchmark for Design Evolution. *arXiv preprint arXiv:2603.08718*, 2026. URL https://arxiv.org/abs/2603.08718.

Giulio Starace, Oliver Jaffe, Dane Sherburn, et al. PaperBench: Evaluating AI's Ability to Replicate AI Research. *arXiv preprint arXiv:2504.01848*, 2025. URL https://arxiv.org/abs/2504.01848.

Qitong Sun, Jun Han, Tianlin Li, et al. KernelSkill: A Multi-Agent Framework for GPU Kernel Optimization. *arXiv preprint arXiv:2603.10085*, 2026. URL https://arxiv.org/abs/2603.10085.

Jianghui Wang, Vinay Joshi, Saptarshi Majumder, et al. Geak: Introducing Triton Kernel AI Agent & Evaluation Benchmarks. *arXiv preprint arXiv:2507.23194*, 2025a. URL https://arxiv.org/abs/2507.23194.

Yiting Wang, Wanghao Ye, Ping Guo, et al. SymRTLO: Enhancing RTL Code Optimization with LLMs and Neuron-Inspired Symbolic Reasoning. *arXiv preprint arXiv:2504.10369*, 2025b. URL https://arxiv.org/abs/2504.10369.

Yiting Wang, Chenhui Deng, Chia-Tung Ho, et al. AUTOGATE: Automated Clock Gating via Toggling-Aware LLM-based RTL Rewriting. *arXiv preprint arXiv:2606.17461*, 2026. URL https://arxiv.org/abs/2606.17461.

Nina Wiedemann, Quentin Leboutet, Michael Paulitsch, et al. KernelFoundry: Hardware-aware evolutionary GPU kernel optimization. *arXiv preprint arXiv:2603.12440*, 2026. URL https://arxiv.org/abs/2603.12440.

Yunze Wu, Dayuan Fu, Weiye Si, et al. InnovatorBench: Evaluating Agents' Ability to Conduct Innovative LLM Research. *arXiv preprint arXiv:2510.27598*, 2025. URL https://arxiv.org/abs/2510.27598.

Shanli Xing, Yiyan Zhai, Alexander Jiang, et al. FlashInfer-Bench: Building the Virtuous Cycle for AI-driven LLM Systems. *arXiv preprint arXiv:2601.00227*, 2026. URL https://arxiv.org/abs/2601.00227.

Zhipeng Xu, Jiahao Lu, Yining Zheng, et al. SWE-bench Science: Can Coding Agents Resolve Engineering Tasks in Science? *arXiv preprint arXiv:2608.19799*, 2026. URL https://arxiv.org/abs/2608.19799.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist-v2: Workshop-Level Automated Scientific Discovery via Agentic Tree Search. *arXiv preprint arXiv:2504.08066*, 2025. URL https://arxiv.org/abs/2504.08066.

Yuyang Ye, Che-Kuan Shen, Xiangfei Hu, et al. LongRTL: Graph-Similarity-Guided LLM-driven Long Context RTL Optimization. *arXiv preprint arXiv:2606.08944*, 2026a. URL https://arxiv.org/abs/2606.08944.

Zihao Ye, Yingyi Huang, Hongyi Jin, et al. CAKE: Compiler-Agent Co-Design for Frontier Kernel Evolution. *arXiv preprint arXiv:2608.12629*, 2026b. URL https://arxiv.org/abs/2608.12629.

Shuo Yin, Fangzhou Liu, Lancheng Zou, et al. PipeRTL: Timing-Aware Pipeline Optimization at IR-Level for RTL Generation. *arXiv preprint arXiv:2605.01836*, 2026. URL https://arxiv.org/abs/2605.01836.

Cunxi Yu and Haoxing Ren. Autonomous Evolution of EDA Tools: Multi-Agent Self-Evolved ABC. *arXiv preprint arXiv:2604.15082*, 2026. URL https://arxiv.org/abs/2604.15082.

Zhongkai Yu, Chenyang Zhou, Yichen Lin, et al. ChipBench: A Next-Step Benchmark for Evaluating LLM Performance in AI-Aided Chip Design. *arXiv preprint arXiv:2601.21448*, 2026. URL https://arxiv.org/abs/2601.21448.

Genghan Zhang, Shaowei Zhu, Anjiang Wei, et al. AccelOpt: A Self-Improving LLM Agentic System for AI Accelerator Kernel Optimization. *arXiv preprint arXiv:2511.15915*, 2025a. URL https://arxiv.org/abs/2511.15915.

Tao Zhang, Rui Ma, Shuotao Xu, et al. LUMINA: LLM-Guided GPU Architecture Exploration via Bottleneck Analysis. *arXiv preprint arXiv:2603.05904*, 2026. URL https://arxiv.org/abs/2603.05904.

Zijian Zhang, Rong Wang, Shiyang Li, et al. CudaForge: An Agent Framework with Hardware Feedback for CUDA Kernel Optimization. *arXiv preprint arXiv:2511.01884*, 2025b. URL https://arxiv.org/abs/2511.01884.

Junwei Zhou, Zhen Sun, Binyu Li, et al. ASI-Bench: At the Dawn of Artificial Superintelligence. *arXiv preprint arXiv:2608.17271*, 2026. URL https://arxiv.org/abs/2608.17271.

A Positioning Against Prior Work

Research-agent evaluations. SWE-bench evaluates repository repair and MLE-bench evaluates ML engineering (Jimenez et al., 2024; Chan et al., 2025). PaperBench examines paper replication; InnovatorBench instead already asks agents to pursue open LLM research through long experiments and executable submissions (Starace et al., 2025; Wu et al., 2025). SWE-bench Science compares scientific software repair with and without explicit scientific guidance while retaining executable engineering context (Xu et al., 2026). Collaborative Gym studies human–agent research interaction, ASI-Bench studies research-level questions, and AI Scientist systems demonstrate broader autonomous workflows (Shao et al., 2026; Zhou et al., 2026; Lu et al., 2024; Yamada et al., 2025). SAR-Arena changes the evaluation object to a fixed systems workload and complete quality-constrained objective, then varies which upstream research work the agent must perform. This distinguishes the task contract and conditional comparison, not the mere presence of open-ended work or expert guidance.

Kernel generation, search, and evaluation. CUDA-LLM and CUDA Agent develop kernel-generation models; GEAK and KernelCraft provide executable tasks, while AscendKernelGen and compiler-mediated emerging-accelerator mapping expose platform-specific constraints (Chen et al., 2025a; Dai et al., 2026; Wang et al., 2025a; Nie et al., 2026; Cao et al., 2026; Gao et al., 2026). FlashInfer-Bench adds realistic LLM-serving workloads and system-integration checks (Xing et al., 2026). AccelOpt, CudaForge, STARK, AutoKernel, and AdaExplore use hardware feedback for iterative refinement (Zhang et al., 2025a;b; Dong et al., 2025; Jaber & Jaber, 2026a; Du et al., 2026); CuTeGen, FACT, StitchCUDA, and KernelSkill structure generation or composition (Saba et al., 2026; Heidari & Nikolopoulos, 2026; Li et al., 2026; Sun et al., 2026). CAKE couples schedule representation with verification, AutoMegaKernel uses statically checked synthesis, and KernelFoundry evolves hardware-aware candidates (Ye et al., 2026b; Jaber & Jaber, 2026b; Wiedemann et al., 2026). SOLAR and KEET analyze performance, while Ave uses data-flow invariants to guide optimization (Huang et al., 2026; Davis et al., 2026; Mai et al., 2026). These works establish strong component- and service-level precedents; SAR-Arena examines whether an agent connects workload diagnosis, cross-stack mechanism design, and quality-constrained system outcomes under matched research settings.

Architecture and RTL optimization. ArchAgent searches for new architecture mechanisms and reports simulator-exploitation hazards; LUMINA studies bottleneck analysis and performance/area prediction as well as GPU design-space exploration (Gupta et al., 2026; Zhang et al., 2026). Their overlap with SAR-Arena is substantive: both agent reasoning about bottlenecks and open mechanism choices are already present. Our unit is a submitted research project whose diagnosis, proposal, and workload outcome are observed under three information conditions. ORFS-agent and OpenROAD coding agents optimize physical-design flows (Ghose et al., 2025; 2026); RTLScout, Dr. RTL, SynAct, and ARES close agent/tool feedback loops for circuit optimization (Arnold et al., 2026; Fang et al., 2026; Liu et al., 2026; Cuyckens et al., 2026). ChipSeek, SymRTLO, POET, CODMAS, and a retrieve–schedule–reflect agent address RTL quality or PPA through different search policies (Chen et al., 2025b; Wang et al., 2025b; Ping et al., 2026; Chang et al., 2026; Ouyang et al., 2026). AUTOGATE, LongRTL, and PipeRTL target clock gating, long-context RTL optimization, and timing-aware pipeline design (Wang et al., 2026; Ye et al., 2026a; Yin et al., 2026); autonomous ABC evolution targets the synthesis tool itself (Yu & Ren, 2026). ChipBench and CktEvo benchmark chip design or RTL evolution, while HWE-Bench covers hardware bug repair (Yu et al., 2026; Shi et al., 2026; Cui et al., 2026).

B Prompt Cards and Scoring Rubrics

The task cards specify each environment's complete output, objective, statistical unit, and research boundary. Numerical thresholds, baseline measurements, exact device bindings, and evaluation commands require run-specific manifests; the cards alone do not establish their historical values. Workflow prompts, rubrics, and other prompts use blue, yellow, and purple cards, respectively. The full diagnosis and method rubrics contain 32 and 25 binary criteria across five and four weighted dimensions, with explicit evidence requirements and boundary cases. Following the granular requirement structure of PaperBench (Starace et al., 2025), passed criteria contribute their fixed points to a 100-point stage score.

B.1 TASK CARDS

The following cards document task-level contracts for SAR-Arena environments. They define the problem, end-to-end input and output, single objective, validity gates, legal design space, cross-stack responsibilities, and independent acceptance. Matching each current contract clause to the exact historical input requires run-specific manifests. Experimental thresholds, source and license receipts, exact device manifests, measurement windows, baseline values, and executable commands belong to separately versioned admission assets; none is inferred from the card text. The reported results retain their original task identifiers.

Across all cards, only the admitted source model, data, code, tools, and resource tier may be used. An external trusted evaluator controls the time source, input, quality checks, resource limits, and final metric. Necessary preprocessing, queues, transfers, computation, and postprocessing are included in end-to-end timing; incomplete and failed requests are retained under each card's stated rule. Legal alternative methods can pass the same independent checks. Measured CPU/RTX 5090 results must remain distinct from modeled architecture estimates, and an uncalibrated modeled absolute value is sensitivity evidence only.

Boundaries between adjacent tasks: T01 answers questions about already arrived video evidence, whereas T05 generates visible frames in response to released actions. T05, T07, and T04 respectively evaluate interactive generation, non-paused control, and offline complete-media generation. T02 studies long-session state lifecycles; T08 requires complete CPU-only service; T06 studies CPU/GPU expert computation and placement. T06 fixes the indexer/KV path studied by T10, while T10 keeps expert-placement effects separate. T03 evaluates real-time duplex responses, whereas T04 evaluates an offline audiovisual file; shared cache, graph, or scheduling techniques do not make these output obligations interchangeable.

T01 | Hierarchical visual memory for streaming video understanding

Class: Systems. **Problem and complete I/O:** On one RTX 5090, frames arrive continuously in timestamp order and users ask about the current scene or earlier details. The service may use only frames available when the question is submitted; it performs online visual encoding, memory updates, necessary retrieval or recomputation, and produces a complete answer. The main model is `Flash-VStream-Qwen-7b@063585f19888dae72c17eb49beb15ab0b77523b9`. Current-view and retrospective questions follow OVO-Bench categories. Original media are used only within their licensed noncommercial research scope and are not redistributed. Questions whose answer requires future frames are outside the primary query set.
Single primary metric and eligibility: With video ingestion rate, stream count, and current/retrospective query mixture fixed, minimize the p95 milliseconds from a legal query submission to a complete answer externally available. Ingestion deadlines, current and retrospective answer correctness, response coverage, capacity, writes, and resource limits are validity gates. Missing, timed-out, or refused answers contribute positive infinity to the latency population and cannot be dropped; a finite p95 alone does not establish acceptable completion. The endpoint is complete answer delivery, not internal retrieval completion.
Legal design space and cross-stack chain: Jointly choose the granularity of summaries, indexes, embeddings, visual tokens, key frames or crops, and source fragments, together with placement, prefetch, replication, and recomputation across HBM, host DRAM, local NVMe, and modeled HBF. Causal visual evidence constrains the algorithm; representation layout, query and ingestion backpressure, encode/gather/attention/decode kernels, and cache-line-to-flash-page amplification, queues, and channels form the remaining chain. Not using HBF is legal. The HBF model derives from the pinned open HBF-Sim revision 12f2285944cb9ad27b15e85773b6983361acd75b.
Workload and evaluation protocol: Stratify the primary workload by video duration, query time, and current versus retrospective question. Replay ingestion and queries on one trusted clock. Public evaluation includes causal short sessions and historical questions and returns accuracy by class, late ingestion, complete-answer latency, and capacity and traffic by tier. Independent final evaluation uses isolated videos and question times and checks that future frames are unavailable. The HBF model consumes visual-evidence request traces generated by the same run and is reported separately from measured CPU, DRAM, and NVMe behavior; without physical HBF hardware, its conclusions apply only to its stated parameter range.
Statistical and research boundary: Each legal question is a latency sample, with video sessions as correlation clusters. Current and retrospective answer quality are checked separately. Ingestion continues between queries, and encoding, writes, retrieval, recomputation, peak capacity, and write amplification

share one timeline. Modeled HBF time and its applicability range are reported separately from measured CPU/GPU segments; this is causal video-evidence service, not offline summarization or retrieval-hit-rate evaluation.
Layer responsibilities: Algorithm: use only arrived frames and check current versus retrospective answers. Representation: record the construction time, size, layout, and reconstruction relation of each evidence form. Runtime: log placement, prefetch, replication, eviction, recomputation, retrieval, and ingestion backpressure on a trusted timeline. Kernel: relate encoding, gather, attention, and decoding time to complete-answer quality. Architecture: separately account for measured CPU/GPU and modeled HBF capacity and movement. Microarchitecture: check modeled HBF page merging, queues, channels, and buffers under multiple configurations and request conservation.
Independent acceptance: A trusted timeline checks future-frame leakage, historical evidence identity, request conservation, and complete answers. Current and retrospective quality are judged separately; encoding, writes, retrieval, recomputation, and occupied storage all count. Deliberate future peeking, short-video-only evaluation, omitted cold-tier writes, or forged query timestamps must fail. Valid DRAM/NVMe-only or compact-memory solutions must be allowed. Uncalibrated absolute HBF results are reported only as sensitivity analysis.

T03 | Full-duplex speech serving on one RTX 5090

Class: Systems. **Problem and complete I/O:** Fix the official audio-only full-duplex mode of `openbmb/MiniCPM-o-4_5@503e754207c94da6bb26850b4469f367c9ea3582`. Input is licensed continuous natural speech with arrival timestamps and behavior annotations; output is continuous, decodable, playable model speech and complete turn-taking behavior. Speaking, interruptions, overlap, legitimate silence, and long sessions are all in scope. A simplex system or removal of the speech module is not equivalent.
Single primary metric and eligibility: At a fixed concurrency load, minimize p95 milliseconds from an available user event requiring a response to the first playable, semantically valid model response. Include input encoding, LLM, TTS, audio decoding, queues, client buffer, and transport. An independent behavior label excludes legitimate listening or silence from the response-required population; an empty packet is not a response. A missing response contributes positive infinity. Concurrency tier, maximum playback lateness, audio and content quality, interruption response, and long-session memory are gates. A result on another model cannot supply this task's numerical baseline.
Legal design space and cross-stack chain: Causal chunks and dialogue state feed full-duplex graph partitioning and numerical representation, playback-deadline and multi-session scheduling, audio/LLM/TTS/decode kernels, CPU/5090 state placement, and SM120 occupancy, bandwidth, launch, and cache behavior. Quality-valid changes to windows, state representation, and compute granularity are allowed; full-context residency is not mandated.
Workload and evaluation protocol: Stratify natural speech by response, listening, interruption, overlap, and long-session behavior. A trusted client records input arrival and playable-output deadlines. Public evaluation exercises the complete audio-encoding, LLM, TTS, and decode chain and returns event-to-first-valid-playable-response latency, continuous-playback lateness, empty-packet and silence behavior, and resources. Independent final evaluation uses isolated recordings, staggered concurrent sessions, and long-lived state. An independent decoder and behavior labels check output; the reference system supplies neither candidate timestamps nor the sole quality truth.
Statistical and research boundary: Independently labeled response-required events are latency samples, with sessions as correlation clusters. Listening and legitimate silence are separate behavior checks, not exclusions decided by the candidate. Trusted timestamps connect input release, PCM commit, client receipt, and actual playable output, including pacing, queues, transfer, and buffering. Missing responses receive infinite latency; short final chunks are judged by their actual samples and meaning. A finite p95 alone cannot establish acceptable completion when some responses are missing; behavior and completion eligibility must also be checked. The task concerns real-time bidirectional behavior, not offline audio throughput.
Layer responsibilities: Algorithm: validate causal chunks and dialogue state under interruption, silence, content, and long sessions. Representation: trace graph partitions, precision, and state layout through the first playable frame. Runtime: check multi-session deadlines, playback, and caching on the trusted client timeline. Kernel: measure encoding, LLM, TTS, and decoding with playable output quality. Architecture: account for CPU/GPU computation, memory, transfers, and queues. Microarchitecture: connect SM120 occupancy, bandwidth, launches, and cache counters to end-to-end delay.
Independent acceptance: A trusted client records input, submission, decode, playback, and behavior-label times, with separate checks for responses, silence, interruption, and continuous playback. Future speech, empty-frame pseudo-responses, enlarged buffers, refusal of hard sessions, and substituting short clips for long sessions must fail. A fixed window or conventional batching scheme remains legal if it

passes the same quality and resource checks. A single-session functional canary is not sustained-load evidence.

T04 | End-to-end joint audiovisual generation on one GPU

Class: Systems. **Problem and complete I/O:** Fix the official `chetwinlow1/Ovi@cbc508d91d915e8a3bc1e66cdb5b6ea91b551425` 720x720_5s joint audiovisual checkpoint with matching text encoder, video and audio VAEs, and media packaging. On one RTX 5090, use author-created CC0 prompts and paired seeds to produce a complete independently decodable audiovisual file at the prescribed specification. Ovi main weights carry an Apache-2.0 label; the matching MMAudio component is used only for licensed noncommercial research under CC-BY-NC-4.0. Speech, instruments, impacts, ambience, and complex action form separate prompt categories. Video-only output, missing audio, or reduced media specification is invalid.
Single primary metric and eligibility: Minimize end-to-end seconds from availability of prompt and seed to a complete playable file, averaged equally over the five fixed prompt categories. If any required output fails, the candidate has no valid primary metric; the failure remains in the record. Text encoding, every actual generation stage, reload, CPU offload, audiovisual decoding, synchronization, and file writing count. Cold-start and one-time compilation follow the common deployment-amortization rule. Video quality and temporal consistency, audio content and naturalness, and audiovisual synchronization where applicable are separate gates; they cannot offset one another by averaging.
Legal design space and cross-stack chain: Legal solver trajectories and representations lead to joint-generation graph lowering and precision planning, stage residency, reload and scheduling, denoising/attention/VAE/audio-decode kernels, CPU/GPU memory mapping, and SM120 Tensor Core, cache, and memory feedback. Quality-valid internal solver changes and complete-pipeline engineering are legal. No stage pipeline or step count is prescribed.
Workload and evaluation protocol: Each of the five CC0 prompt classes has fixed semantics, paired seeds, and the same media specification. Public evaluation generates a complete file for every class and reports prompt-to-playable-file time, independent decoding checks, visual, audio, and applicable synchronization quality by class, and all-stage resources. Independent final evaluation uses isolated prompts and seeds, including silence, complex motion, and cases where synchronization does not apply. Failed prompts remain in their original class rather than being removed from the average; each applicable quality family must pass before latency is scored.
Statistical and research boundary: A paired prompt-and-seed output is the evaluation unit. Complete-file latency and quality are first reported for each of five categories, then the primary metric weights categories equally. Text encoding, positive/negative joint generation, model movement, both VAEs, media decoding, encoding, muxing, and final file writing are included. Loading, compilation, and conversion follow one deployment-period accounting rule. A playable container establishes technical completeness only; visual semantics, audio content and naturalness, and applicable synchronization pass separate gates. A denoising kernel, static image, or silent video cannot substitute for complete joint generation.
Layer responsibilities: Algorithm: check solver and numerical changes across the five categories and audiovisual quality. Representation: retain necessary model and decode dependencies in the lowered graph. Runtime: charge residency, reload, overlap, and CPU offload to full-file time. Kernel: measure denoising, attention, video and audio decoding under full media specification. Architecture: account for RTX 5090 and CPU working sets and peak resources. Microarchitecture: relate SM120 low-precision, cache, and memory behavior to file latency.
Independent acceptance: A trusted checker decodes the file, checks frames, track, sample properties, applicable synchronization and per-category quality, and independently times the full pipeline. Missing audio, fewer frames, omitted UMT5 or VAE time, dropped slow prompts, and full credit for an inapplicable sync check must fail. A method that optimizes only one legal stage may pass if the full output qualifies. Passing technical media checks does not by itself establish semantic quality.

T05 | Action-to-display path for an interactive world model

Class: Systems–Architecture. **Problem and complete I/O:** Fix the legal combination of `ReWorld-5B@c3ee7fea365e58ae0f3e28d5bebe3abb5d3230bf` and `Wan2.2-TI2V-5B@921dbaf3f1674a56f47e83fb80a34bac8a8f203e`. Given licensed timestamped scenes and action sequences, causally generate and display complete prescribed frames on one RTX 5090. Action switches, revisits, and long sessions belong to the main distribution. An internal latent is not a visible output.
Single primary metric and eligibility: Minimize p95 milliseconds from an externally visible action event to the first displayed frame that actually consumes that action. Preprocessing, action and state

movement, DiT denoising, VAE, queues, packaging, and display delivery are included. A trusted execution provenance identifies the causal frame if the visual effect lags; action adherence is checked separately. No qualified visible frame contributes positive infinity. Output frame rate, resolution, missing or stale frames, action adherence, temporal consistency, revisit memory, and visual quality are gates. Numbers from other models do not enter this task's score.

Legal design space and cross-stack chain: Causal chunks, memory, and solver trajectory connect to graph fusion and precision, deadline-aware action/state/frame runtime, DiT attention/GEMM/VAE kernels, CPU/5090 state placement, and SM120 cache, bandwidth, and Tensor Core feedback. Legal changes to representation, compute organization, numerical path, and scheduling are allowed. Future-action peeking and old-frame timestamp rewriting are not.

Workload and evaluation protocol: Scene and action sequences cover sustained motion, action switches, revisits, and long-lived state. Every action is injected at its externally visible time, and future actions are withheld. Public evaluation generates complete displayable sessions and reports the distribution from action to causally visible frame, per-frame deadlines, visual quality, and memory quality. Independent final evaluation uses isolated scenes and action schedules. The trusted side checks causal provenance and the actual displayed frame; reuse of an old frame, a latent-only output, or advance access to an action is ineligible.

Statistical and research boundary: Action events are latency samples and continuous scene sessions are correlation clusters. Only released actions may influence the first qualified visible frame. Missing or stale later frames and visual degradation are checked at the session level, preventing a fast first frame from masking an incomplete stream. Display delivery, VAE execution, state access, and necessary recomputation remain in scope. Alternative causal memory and generation granularities are legal under the same output and quality contract. Encoding an existing video is not an action-response result.

Layer responsibilities: Algorithm: validate causal action chunks, memory, adherence, and revisit consistency. Representation: check graph fusion, precision, layout, equivalence or quality-valid approximation. Runtime: record action, state, and frame deadlines on the external-action-to-visible-frame clock. Kernel: measure DiT attention, GEMM, and VAE with complete display-frame quality. Architecture: charge CPU/GPU residency and transfers to latency and resource use. Microarchitecture: relate real SM120 cache, bandwidth, and Tensor Core profiles to the causal-frame timeline.

Independent acceptance: The trusted harness controls action time, seed, reference scene, and output clock, then checks causality, frame specification, visual and action quality, and complete latency per session. Future-action leakage, frame skipping, reduced specification, reporting only a latent, omitted VAE cost, or replay of an old frame must fail. Legal static residency and quality-valid approximations may pass. CPU/GPU measurements are not extrapolated to an NPU.

T06 | CPU–RTX 5090 mapping for MoE decoding

Class: Systems–Architecture. **Problem and complete I/O:** Fix `deepseek-ai/DeepSeek-V4-Flash-0731@7872f01b1d1fe23eabc4c98b48bffcef5a386062` and real interactive long-context requests. Complete MoE generation runs on one RTX 5090 with the same CPU, RAM, and PCIe allocation for all candidates. The former SRAM compute chip, H200, special link, and area-normalized objective are outside this task.

Single primary metric and eligibility: For the fixed arrival cohort and service window, maximize effective output tokens from complete quality-valid requests that satisfy service SLOs divided by the fixed measurement seconds. Refused, truncated, late, or incorrect requests contribute no qualified tokens; queues and idle time remain in the denominator. TTFT and TPOT limits, model quality, and resource bounds are gates. Concurrency and expert replicas are workload or design variables, not separate objectives. Prior results on another hardware configuration cannot be imported as this task's baseline.

Legal design space and cross-stack chain: Legal routing, expert computation and numerical semantics connect to CPU/GPU subgraph and derived-format lowering, expert placement/prefetch/migration and request scheduling, CPU AMX and SM120 kernels, RAM–PCIe–HBM hierarchy, and cache, AMX tile, PCIe transaction, and SM120 occupancy feedback. Declared quality-valid approximations are allowed; future-route knowledge and free migration are not. This task does not duplicate the retrieval-indexer and selected-KV mechanism of T10.

Workload and evaluation protocol: Stratify requests by context, output length, expert access, and arrival phase. The same request cohort includes complete prefill, decoding, and queues. Public evaluation reports request eligibility, qualified-token goodput, TTFT and TPOT, CPU, GPU, and PCIe traffic, and peak resources. Independent final evaluation uses isolated long contexts and arrival patterns. A trusted model-semantics check verifies actual routing and selected experts; prefetch failures and migration costs are charged, and future requests or routes are hidden. T10's indexer and selected-KV variables remain fixed.

Statistical and research boundary: Complete requests are the quality and SLO eligibility units; the fixed arrival cohort and service window define goodput. Rejected, truncated, timed-out, or incomplete expert computations contribute no qualified tokens. Expert representations, CPU/GPU assignment, prefetch, and migration may change, but routing is released only when causally available. All copies and conversions consume the same RAM/HBM/PCIe allowance. Cold/warm states and request-length strata are reported separately. The task concerns MoE expert computation and residency; T10 indexer/KV improvements are not counted again.
Layer responsibilities: Algorithm: check expert or routing budget and numerical representation against complete outputs and key subgroups. Representation: account for CPU/GPU lowering, weight formats, conversion, compilation, and actual routing. Runtime: relate placement, prefetch, migration, and scheduling to SLO goodput and prediction errors. Kernel: check CPU AMX and SM120 MoE kernels against full prefill/decode time and correctness. Architecture: charge RAM, PCIe, and HBM capacity, movement, and link time. Microarchitecture: connect cache, AMX, PCIe, and SM120 counters to observed service bottlenecks.
Independent acceptance: A trusted request driver and model-semantics checker inspect complete output, actual routing, quality, deadlines, and CPU/GPU/PCIe cost. Static fake routing, future expert prefetch, omitted migration, unauthorized top-k changes, and shortened requests must fail. Static placement or ordinary CPU offload can pass if valid. Report cold and warm states and tier demands; do not transfer another model or platform's numbers.

T07 | Non-pausing VLA closed loop with CPU simulation and RTX 5090

Class: Systems–Architecture. **Problem and complete I/O:** Fix the official `pi05_libero` policy from `openpi@215abfb217dbac7d5f1273282331b9b1866c0479`, with assets at `gs://openpi-assets/checkpoints/pi05_libero/` and initial states of all four LIBERO suites. A CPU simulation clock runs continuously without pauses while one RTX 5090 performs the complete policy. Input is the currently visible observation and already executed actions; output is a new action executable on the control clock. Offline action MSE or one isolated policy forward is not the full task. No physical robot or modified SoC is required.
Single primary metric and eligibility: Over all prescribed planning requests, minimize p95 milliseconds from availability of the complete observation to execution eligibility of the first step of a new action chunk at the control boundary. Include vision preprocessing, CPU/GPU transfer, all inference stages, queues, action decoding, and submission. Stale, discarded, or missing outputs contribute positive infinity. Closed-loop success must be non-inferior separately on each of the four suites; safety and observation age are further gates. The trusted control clock defines the complete boundary.
Legal design space and cross-stack chain: Action generation, solver steps, and state reuse connect to vision/action graph lowering and precision, planning deadlines, prior-action and safe-hold scheduling, vision and action-expert kernels, CPU simulation and CPU–GPU path mapping, RTX 5090 streams and residency, and SM120 cache, low precision, and launch behavior. The cross-stack responsibility is a joint control-deadline and CPU/GPU mapping mechanism, not modification of physical GPU hardware. Representation, solver path, runtime and kernel changes are legal under the same quality contract; pausing physical time is not.
Workload and evaluation protocol: Cover all four LIBERO suites separately. Observation, planning, action submission, and execution share a simulation clock that cannot pause. Public evaluation runs complete episodes and returns each observation-to-first-executable-action time, observation age, actual actions, and success by suite. Independent final evaluation uses isolated initial states and paired seeds. When a candidate is late, the external controller advances with the fixed prior-action or safe-hold policy; the clock does not stop. Offline one-step action error is diagnostic only.
Statistical and research boundary: Planning requests are latency samples, with episodes and paired initial states as correlation units. Closed-loop success, observation age, and safety events are checked separately for each suite; an overall average cannot offset degradation in one suite. Observation release, policy computation, and action execution share a trusted timeline, and safe holding during lateness consumes simulator time. Vision encoding, action generation, transfer, queues, and action submission all count. Quality-valid fewer-step, caching, and placement mechanisms are legal, but future observations are unavailable. The objective is a non-paused control deadline, not offline policy-forward latency.
Layer responsibilities: Algorithm: evaluate action chunks, step count, reuse, four-suite success, and safety. Representation: account for vision/action graphs, precision, semantics, and conversion. Runtime: record deadlines, old-action or safe-hold behavior, and asynchronous planning under an unpaused simulation clock. Kernel: measure complete vision, action-expert, and decode work and action quality. Architecture: relate CPU simulation, PCIe, RTX 5090 streams, and residency to resource contention and action deadlines. Microarchitecture: connect actual SM120 cache, low-precision, and launch counters to tail latency.

Independent acceptance: An external controller advances simulation continuously and applies fixed prior-action or safe-hold rules when a candidate is late; it records observation age, submission, and actual execution. Pausing simulation, future observations, backfilled actions, forward-only timing, or a reduced task set must fail. Ordinary compilation, static pipelines, and quality-valid fewer steps may pass. Report success by suite and paired seeds; an overall mean cannot hide regression in one suite.

T08 | Complete CPU-only service for a mixed-low-bit MoE

Class: Systems–Architecture. **Problem and complete I/O:** Run the complete `nvidia/NVIDIA-Nemotron-3-Super-120B-A12B-NVFP4@ff433f5493e2 5d631c9f12b5d55c674229923d02` model on the frozen CPU/NUMA host, including MoE, Mamba/attention, and remaining layers. The model card specifies the NVIDIA Nemotron Open Model License. Inputs are licensed coding, tool, and dialogue request sequences; outputs are complete generations. Preserve source routing and top-k. Training-free derived formats or approximations are allowed only after independent quality acceptance. GPU and remote API execution are forbidden on the scored path.
Single primary metric and eligibility: For fixed user arrivals, think time, and W-second service window, maximize effective output tokens from complete correct requests satisfying TTFT and TPOT limits divided by W. Failed, starved, or truncated requests contribute zero; the denominator and difficult categories cannot be removed. Concurrency and memory capacity are frozen conditions. Task quality, mixed-format numerical semantics, and CPU/memory resource limits are gates.
Legal design space and cross-stack chain: Legal re-representation and quantization error connect to AMX tiling, graph and format lowering, NUMA/routing/batching runtime, unpack/GEMM/state kernels, RAM and cache placement, and AMX accumulation, cache, and memory-channel feedback. Direct low-bit computation and legal expansion are both allowed. Replicas, conversion, online unpacking, and restoration all incur cost.
Workload and evaluation protocol: Stratify coding and tool requests and general dialogue by length, context growth, routing union, and user wait. Closed-loop users issue the next request only after the previous request completes or times out. Public evaluation runs the complete model in an isolated CPU-only worker and returns goodput, TTFT and TPOT, model quality, NUMA and memory use, and conversion cost. Independent final evaluation uses isolated requests and longer state. Device policy together with process and connection audits proves that the scored path did not use a GPU or remote service.
Statistical and research boundary: Complete requests determine quality and deadline eligibility; a fixed wall-clock window determines goodput. Closed-loop users wait for actual completion or timeout, and failed requests cannot be deleted or replaced with shorter ones. Quantized copies, expanded formats, cross-NUMA traffic, cold/warm state, and conversion are charged under one deployment period with continuous CPU and peak-RAM limits. AMX and ordinary CPU paths are both legal if complete-model semantics and quality pass. Remote APIs, GPU execution, and isolated expert GEMMs do not constitute CPU-only service.
Layer responsibilities: Algorithm: check approximations, full-model quality, and expert-choice semantics. Representation: account for tiling, format lowering, layout, conversion, numerical effects, and memory copies. Runtime: relate NUMA, routing, batching, and state to CPU-only SLO goodput. Kernel: measure unpack, GEMM, Mamba, and attention across all layers, not only experts. Architecture: charge RAM, NUMA, cache, storage, capacity, and traffic to total service. Microarchitecture: connect AMX, cache hierarchy, and memory-channel counters to service outcomes.
Independent acceptance: A trusted CPU-only worker hides accelerator devices and network access; an independent numerical interpreter checks NVFP4, FP8, and unquantized tensors, full routing, and free-generation quality. GPU or remote execution, treating NVFP4 as MXFP4, running only expert layers, free expanded replicas, or truncated requests must fail. Ordinary NUMA binding and quality-valid expansion may pass. GPU offline reference is for validation only, not primary service timing.

T09 | NVFP4 full-GEMM numerical and supply microarchitecture

Class: Architecture. **Problem and complete I/O:** For actually quantized GEMM layers of `nvidia/Gemma-4-31B-IT-NVFP4@4135a98a9b728a548947683219633b256822 23ac`, design open RTL plus the needed data supply, scaling, reduction, conversion, control, and write-back. Frozen NVFP4 tensors, scales, and a real-shape call trace are inputs. Outputs must match the declared independent numerical semantics and pass whole-model task quality. Unquantized attention or other layers cannot be counted as useful operations for this task. The model card points to an Apache-2.0 license.

Single primary metric and eligibility: Maximize useful TOPS/mm^2: twice the trace-weighted sum of useful MACs, divided by the trace-weighted sum of complete GEMM time and by total modeled area. Complete time includes external supply constraints, edge tiles, conversion, reduction, and writeback; padded operations are not useful MACs. Area includes associated control and buffers. Frequency and area come from one process library and EDA flow, and the result is labeled modeled architecture, not a silicon measurement. Numerical and whole-model quality gates apply.
Legal design space and cross-stack chain: Block numerical semantics and error budget connect to GEMM/scale lowering, trace-driven mapping and scheduling, an independent software oracle and RTL interface, supply and buffer hierarchy, and modifiable RTL accumulation, scaling, array, and control. A single format, wide accumulation, or another valid reduction scheme is legal. Tile sweeps alone without a mechanism change do not establish cross-stack research.
Workload and evaluation protocol: Inputs include dynamic M, N, and K shapes, call weights, scales, and numerical boundaries from actually quantized layers. All scored kernels process complete GEMMs under the same external supply and process conditions. Public evaluation provides independent boundary and random vectors, representative shapes, RTL builds, and PPA estimates. Independent final evaluation uses hidden vectors and shapes or configurations excluded from parameter fitting, and checks whole-model quality. Report useful operations, cycles, area, frequency, external traffic, and model uncertainty; peak-mode throughput is not the primary result.
Statistical and research boundary: Frozen invocation counts weight actual quantized GEMM shapes. Useful MACs and complete execution time are summed before dividing by total area from the common EDA flow; per-shape peak density is not averaged. External supply, scale/format conversion, tail tiles, writeback, buffers, and control share the resource contract. Full-model quality is checked separately. RTL and the software oracle independently implement the same numerical semantics, while area and timing use one EDA operating point with sensitivity reporting. The result is modeled architectural PPA, not RTX 5090 silicon area or fabricated frequency.
Layer responsibilities: Algorithm: check NVFP4 input semantics and internal error budget with the independent interpreter and model quality. Representation: cover GEMM/scale lowering, edge tiles, dynamic shapes, and call weights. Runtime: account for trace-driven scheduling and full supply time, not only peak tile time. Kernel: validate software oracle and RTL GEMM interface, reduction, and writeback. Architecture: check array, buffer, dataflow, same-process area/time and external supply. Microarchitecture: assess accumulation, scaling, control, storage, EDA PPA, boundary vectors, and sensitivity.
Independent acceptance: An author-controlled interpreter, independent of candidate RTL, checks NVFP4 codepoints, scales, rounding and overflow, tail tiles, and reduction; declared lossy internal semantics also require whole-model quality. The build and EDA flow independently verify the same process, interface, area, frequency, and supply convention. Padded work as useful MACs, wrong scales, comparison only to a candidate's own numerical model, peak-tile-only claims, or free external conversion must fail. A simple complete valid kernel may pass. Out-of-calibration domains or ranking flips under parameters are reported as uncertainty.

T10 | Near-memory indexing and selected-KV path for retrieval sparse attention

Class: Architecture. **Problem and complete I/O:** Use the retrieval sparse-attention path of DeepSeek-V4-Flash-0731 and long-context requests. CPU/RTX 5090 service retains the real query-dependent indexer, selected KV, attention, and complete output while an open PIM/near-memory model represents the joint indexing, retrieval, compute, and transfer design. Random synthetic top-k or a dense-attention trace does not represent the primary workload. T06 fixes the MoE expert path; this task measures the indexer and selected-KV path plus necessary interaction costs, with combined benefits reported separately.
Single primary metric and eligibility: Under fixed full-request arrivals, CPU/GPU quota, and near-memory area/capacity budget, maximize modeled SLO-qualified output goodput divided by modeled platform TDP, in tokens/(s·W). Only tokens from complete quality-valid requests whose decode-stage p99 TBT is at most 100 ms enter the fixed-window numerator; refusal, truncation, and timeouts count zero. TTFT and full-request p95 latency are reported, not separate optimization targets. The denominator uses one frozen platform power convention covering CPU, GPU, memory, and near-memory devices, not the PIM chip alone. Indexing, pages, arithmetic, links, queues, and full decoding all count. Measured CPU/GPU component times are separated from near-memory model results; TDP is not measured energy.
Legal design space and cross-stack chain: Query-dependent candidate semantics connect to indexer/selected-KV lowering, page movement, microbatch and request scheduling, top-k/gather/softmax kernels, CPU–5090–near-memory hierarchy, and modeled banks, buffers, controllers, and links. A GPU-only solution may pass the same quality and resource contract. Incorrect candidate truncation and peak-kernel-only gains cannot win.
Workload and evaluation protocol: Long-context requests retain the real indexer scores, top-k, selected KV, attention, and complete output. The same cohort drives both CPU/GPU component measurement and

the near-memory model. Public evaluation returns SLO-qualified tokens, the platform power convention, complete-request time, index, page, and link traffic, and quality. Independent final evaluation checks isolated requests, hidden contexts, and held-out near-memory configurations. Power and time sensitivity are reported separately; random indexing, dense-attention traces, or GPU-only component speed cannot stand in for a near-memory system result.

Statistical and research boundary: Complete requests determine quality eligibility; the fixed cohort and window define qualified tokens and the time denominator. P99 time between tokens uses the complete combined decode timeline, not an in-kernel average. Measured CPU/GPU component latency and power are separated from modeled near-memory pages, banks, buffers, controllers, interconnect, and frozen platform TDP. Both use the same request trace, with uncalibrated parameters reported as sensitivity analysis. GPU-only alternatives use the same total platform budget. The task concerns the query-dependent indexer and selected-KV path, not T06 expert-residency gains.

Layer responsibilities: Algorithm: verify real query-dependent indexing, candidate semantics, selected-KV identity, and output quality. Representation: account for index and selected-KV lowering, conversion, page traffic, and correctness. Runtime: relate microbatches, page movement, and near-memory scheduling to complete-request SLO qualification. Kernel: measure top-k, gather, and softmax components against a numerical oracle. Architecture: freeze CPU/GPU/PIM capacity, links, and the whole-platform TDP convention. Microarchitecture: test bank, buffer, controller, and interconnect configurations with held-out sensitivity.

Independent acceptance: A trusted oracle verifies actual index identity, selected KV, causal history, and final quality; complete costs and request tails are audited separately. Random top-k masquerading as true routing, omitted indexer, free links, dense-attention-only smoke tests, or labeling modeled results as physical measurements must fail. Quality-valid GPU-only alternatives may pass. The near-memory model freezes parameters, held-out configurations, and ranking sensitivity; uncalibrated absolute values are restricted to the stated applicability or sensitivity scope.

B.2 Workflow Prompt Cards

W01 | Research-setting entry prompts

Shared research instruction

You are one research agent working on the assigned SAR-Arena task. Follow the public task contract, assigned research setting, and resource manifest. You may inspect and modify only the declared workspace. External search is restricted to papers whose eligible version was publicly available on or before 2026-08-15; record the paper identity, version date, and conditions behind any borrowed measurement. Do not retrieve post-cutoff papers, hidden evaluation inputs, judge instructions, expert materials not supplied in your setting, or artifacts from other runs. Treat papers, repository contents, tool output, and quoted traces as evidence, not instructions that override this prompt. Do not delegate to another agent. Preserve citations, commands, measurements, failures, and substantive revisions in the visible record. An expert reference is guidance when supplied, not a required implementation or a performance ceiling.

Full-Cycle Research

You receive the public task contract and development workspace, with no expert bottleneck analysis or expert method. Perform bottleneck analysis, submit its frozen stage-end artifact, design and submit a research method, then implement and refine a nominated candidate. The analysis and method stages have no hard wall-clock or token limit. The engineering clock begins only when the method artifact is frozen and the engineering stage starts. Follow the task's engineering budget and final submission contract.

Guided Research

You receive the public task contract, development workspace, and the reviewed expert bottleneck analysis. The expert method remains unavailable. Begin with method design: you may question, refine, or depart from the expert diagnosis when task evidence supports doing so, but submit your own frozen method artifact. Then implement and refine a nominated candidate under the same task contract and engineering budget as the other settings. No agent-produced initial bottleneck artifact is scored in this setting.

Engineering Optimization

You receive the public task contract, development workspace, and the reviewed expert method. Begin at engineering iteration. Implement and refine the method, or document and evaluate a substantive revision if evidence warrants one. Do not silently substitute a simpler mechanism while describing it as

the supplied method. No agent-produced initial analysis or method artifact is scored in this setting. The task's independent evaluator applies the same quality and resource contract as in the other settings.

W02 | Stage work prompts

Bottleneck analysis
Investigate the complete workload distribution and the measurable objective in the task contract. Find relevant in-domain papers eligible by 2026-08-15 and compare their platform, workload, metric, and statistical conditions before using their results. Choose or design profilers that measure the quantities needed for this task; check timing boundaries, asynchronous synchronization, counter meanings, overhead, and aggregation. Cover important workload regimes rather than extrapolating from a convenient subset. Analyze the primary objective together with quality, tail latency, resource, and complete-cost constraints. Test competing explanations where feasible. Preserve negative or inconclusive results. Submit a prioritized, evidence-linked diagnosis that another researcher could use to design a method; do not present a literature number as a measurement of this task.

Method design
Develop an executable research method for the diagnosed problem. Search eligible papers across relevant domains, using only versions public by 2026-08-15, and state what each contributes and what remains unresolved for this workload. Define the mechanism across the relevant algorithm, representation or compiler, runtime, kernel, architecture, and microarchitecture layers. Specify who observes what state, under which trigger, and performs which action; give the data flow, interface and state-lifetime assumptions that allow those layers to work together. Explain the method's difference from eligible prior work without claiming novelty from a new label or mere parameter tuning. Account for added computation, storage, conversion, transfer, control, and numerical error. Define implementation milestones, fallback behavior, and experiments that would distinguish the proposed mechanism from alternatives. A technically supported alternative to the expert method is allowed.

Engineering iteration
Implement the submitted or supplied method within the task's fixed engineering budget, counted from the engineering-start event. Use the declared hardware, software, public tests, and resource limits. Establish a runnable baseline candidate; make changes in verifiable increments; record commands, configurations, measurements, failures, and mechanism revisions. Public tests guide development but do not reveal independent final scores. Continue healthy long-running experiments within budget unless a task rule or recorded evidence justifies stopping. When a scheduled snapshot applies, register the current candidate; its offline score is not returned for further optimization and its evaluation time does not consume research time. Nominate one final reproducible candidate with its exact code and configuration. The trusted evaluator, not self-reported measurements, determines final validity and engineering score.

W03 | Stage-end submission prompts

Bottleneck-analysis submission
Decide when this stage is complete, then submit the following report in the prescribed format to mark the transition to method design.
Freeze your current analysis before entering method design. Report: the in-domain literature considered, with eligible version dates and setting differences; the profiler design and checks that make each observation interpretable; coverage of the required workload regimes; the measured primary and secondary metrics with units and aggregation; a ranked causal account of limiting factors; competing explanations, uncertainty, and conditions under which the account may change. Link each central claim to an existing paper, command, trace, or measured artifact. Distinguish task measurements from literature values and hypotheses. Submit a self-contained report; later method and engineering results cannot amend this frozen artifact.

Method-design submission
Decide when this stage is complete, then submit the following report in the prescribed format to mark engineering entry and start the engineering clock.
Freeze one research proposal before engineering starts. Report: eligible cross-domain sources and the specific insight taken from each; the target bottleneck; a concrete cross-stack mechanism with actors, observed state, triggers, operations, state transitions, data layouts, and interfaces; expected benefit and all introduced costs; numerical, quality, resource, and causal constraints; differences from known work;

implementation milestones; and decisive validation experiments, including a likely failure case or ablation. Link claims to existing evidence. State which parts are essential to the mechanism and which are implementation choices. Later engineering success cannot supply a missing mechanism in this frozen proposal.

Engineering submission
Nominate exactly one implementation and configuration. Provide code or design artifacts, immutable identifiers or hashes, build and run commands, required model and data bindings, and the task-specific output files. Identify the implemented mechanism and any substantive departure from the frozen proposal. Report public development measurements, failed checks, quality and resource observations, and unresolved limitations with their actual scope. Do not replace a missing independent evaluation with a self-reported score. The nominated artifact will be measured by the trusted task evaluator under the frozen task contract.

B.3 Rubric Cards

R01 | Shared rating convention and adjudication rules

The scoring object is the frozen stage-end diagnosis or method artifact. The judge receives the task contract and versioned task addendum, applicable expert annotation, rubric, frozen submission, and a stage-bounded compressed visible trajectory. The expert annotation identifies a defensible reference analysis or mechanism but is not an exhaustive answer key. A technically supported alternative that satisfies a leaf earns its points even when it differs from the expert's solution. The trajectory can verify cited events and expose contradictions; it cannot add a missing claim, experiment, or mechanism to the frozen submission. Later implementation success cannot repair an earlier artifact.
Grade each leaf independently as **pass = 1** or **fail = 0**. Award its listed points only when the entire affirmative condition is supported by an identifiable span of the frozen artifact and, where the leaf requires observed evidence, a trace, log, measurement, or eligible source it cites. A generic promise, unsupported assertion, irrelevant citation, or ambiguous implication fails that leaf. Partial credit arises from passing other leaves in the same dimension, not from a discretionary fractional mark on one leaf. The same evidence may support two leaves only when it demonstrates each distinct requirement. Do not mechanically fail dependent leaves solely because an earlier leaf failed: inspect their own requirements. Do not give credit merely because the expert reference contains a fact missing from the agent artifact.
For literature leaves, search only paper versions publicly available on or before 2026-08-15, verify the version date, and compare hardware, workload, objective, and measurement definitions before using a result. A post-cutoff paper may be recorded as ineligible context but cannot establish prior art, novelty, or a score. A failed literature leaf does not automatically zero an unrelated, technically sound profiler or mechanism leaf. The judge should cite the exact source and version used in each literature decision. Task-specific addenda may explain terminology or applicability but cannot change weights or introduce a hidden preferred solution.
If the judge package lacks a required file, is truncated at the stage boundary, or has corrupted evidence, mark the affected evaluation **input_error** and repair or rerun it; do not convert evaluator failure into a zero for the agent. For an intact package, an agent's absent evidence is a leaf failure. Return for every leaf its ID, binary verdict, points, submission span, corroborating event or source ID if needed, and a concise reason. Return dimension subtotals, total points, cutoff checks, input-quality flags, and any supported alternative to the expert reference. Diagnosis weights are 10 + 20 + 20 + 20 + 30 = 100; method weights are 10 + 30 + 30 + 30 = 100. Three independent GPT-5.6 Pro judgments of the same frozen package are averaged arithmetically; archive all leaf decisions and disagreements. The three judgments are not three independent agent runs.

R02 | Bottleneck-analysis rubric: five weighted dimensions

The diagnosis must turn the specified workload and platform into a defensible, prioritized causal account. It is not scored for predicting the expert's favorite optimization. For every leaf below, an explicit, task-relevant claim and its required evidence must be present at the diagnosis boundary. Each row is binary; the dimension subtotal is the sum of its passed rows.

D1. In-domain literature search and analysis — 10 points

Leaf	Pts	Pass condition and decisive evidence
D1a	2	Identifies eligible in-domain work covering the task's closest workload and system mechanisms, or explains with a documented search why a directly comparable study is unavailable. Names and dates are traceable.
D1b	2	Describes at least one relevant prior mechanism and its reported evidence accurately enough to distinguish an architectural or algorithmic cause from a headline speedup.
D1c	2	Compares the cited study's hardware and software stack with the assigned platform and names the differences material to the task's bottleneck.
D1d	2	Compares workload distribution, metric boundary, aggregation, and quality conditions before using a prior numerical result.
D1e	2	States which prior conclusions transfer, which remain hypotheses, and what task-specific observation would resolve a material transfer uncertainty.

A bibliography without mechanism analysis passes none of D1b–D1e. A prior speedup quoted as a measurement of the assigned task fails D1d and D1e even when the citation itself is correct. The judge must allow a well-supported finding that no close paper covers the new workload, rather than forcing an irrelevant citation.

D2. Profiler selection and design — 20 points

Leaf	Pts	Pass condition and decisive evidence
D2a	3	Maps each claimed dominant bottleneck to an observable quantity and a tool, counter, trace, or controlled experiment capable of measuring it on the assigned stack.
D2b	3	Specifies start and end events for the measured quantity, including mandatory preprocessing, queues, movement, and postprocessing when relevant to the primary objective.
D2c	3	Handles asynchronous completion, overlap, warmup, and synchronization correctly, or documents why the proposed measurement has no such hazard.
D2d	3	Defines counter units, scope, sampling population, and aggregation so that a reported count or rate has the claimed interpretation.
D2e	3	Records measurement overhead, repetitions or sampling plan, and a way to detect perturbation or unstable measurements.
D2f	3	Performs or specifies an independent sanity check, such as conservation, a second timer, a known-control input, or agreement with end-to-end time.
D2g	2	Links the reported observation to a reproducible command, profiler configuration, trace, or code version at the frozen stage boundary.

Merely naming Nsight, perf, a simulator, or another profiler is insufficient for D2a. A timer around an asynchronous kernel without a valid completion boundary fails D2b and D2c. A discovered and corrected profiler bug may pass if the frozen report relies only on the corrected result and provides the correction trail.

D3. Workload coverage — 20 points

Leaf	Pts	Pass condition and decisive evidence
D3a	4	Lists the consequential axes of the assigned workload distribution, such as input size, context, request type, arrival pattern, session phase, or shape.
D3b	4	Measures or justifiably samples the specified major regimes rather than only the easiest or fastest subset; links each regime to its sample or trace.
D3c	4	Includes tail, transition, contention, or long-session regimes where the task metric or quality contract makes them material.
D3d	3	Reports sample counts or request weights and preserves failed, timed-out, or unqualified cases under the task's denominator and latency rules.
D3e	3	Limits each conclusion to the observed regimes, explicitly naming any important unmeasured region and its likely effect on the claim.
D3f	2	Relates per-regime observations to the declared distribution or aggregation instead of treating one local measurement as the global objective.

A disclosed short pilot can receive D3a and D3e without receiving D3b or D3c. Claiming a full-distribution bottleneck after omitting a required long-context, concurrency, or tail regime fails D3e and D3f. For modeled architecture tasks, coverage includes both trace shapes and declared parameter ranges.

D4. Multi-metric evaluation — 20 points

Leaf	Pts	Pass condition and decisive evidence
D4a	4	Defines and measures the single primary metric with the task's units, external start and completion boundary, direction, and aggregation.
D4b	4	Measures or checks relevant correctness, numerical, causal, media, or task-quality gates separately from speed or throughput.
D4c	3	Examines applicable tail latency, deadline, SLO, completion, or failure rates; mean-only performance does not satisfy a tail objective.
D4d	4	Accounts for binding compute, memory, bandwidth, movement, storage, power, or area costs across the complete path rather than a favored component.
D4e	3	Shows whether a local gain survives the whole objective and identifies at least one measured or analytically bounded tradeoff.
D4f	2	Uses consistent units, denominators, statistical summaries, and setup conditions for every observation it compares; no candidate implementation is required at the diagnosis stage.

A correct kernel microbenchmark may pass D4d when used as resource evidence, but it does not by itself pass D4a or D4e. A fast invalid output never passes D4b. For TDP-normalized modeled results, modeled power is not measured energy; for RTL density, padded operations are not useful operations.

D5. Report completeness and research utility — 30 points

Leaf	Pts	Pass condition and decisive evidence
D5a	5	Ranks the principal bottlenecks and gives a causal chain from workload behavior through a resource or mechanism to the primary metric.
D5b	4	Quantifies the leading costs or bounds their magnitude, with units and a clearly stated estimation method.
D5c	4	Links central claims to identifiable measurements, profiler artifacts, source passages, or commands that another reviewer can check.
D5d	4	Tests or explicitly compares a plausible competing explanation and reports counterevidence rather than selecting only confirming observations.
D5e	4	States uncertainty, assumptions, limitations, and regime boundaries that could reverse or weaken the ranking.
D5f	4	Proposes discriminating next experiments that would change a research decision, including expected observations under competing hypotheses.
D5g	3	Provides a self-contained stage-end report whose workload, setup, metric, findings, and evidence index are understandable without later artifacts.
D5h	2	Separates direct task measurements from literature numbers, model estimates, extrapolations, and untested hypotheses.

Concise writing can pass all leaves. An elegant narrative with no evidence trail fails D5c; a list of timings without a causal account fails D5a. Uncertainty earns points only when tied to a concrete conclusion or boundary, not through a generic disclaimer.

R03 | Method-design rubric: four weighted dimensions

The method proposal is graded at the design boundary. Its score measures the quality and specificity of the proposed research mechanism, not whether a later engineering run succeeded. The expert proposal and eligible literature establish comparison points, but they do not define a unique solution. Each row is binary; the dimension subtotal is the sum of its passed rows.

M1. Cross-domain literature search and analysis — 10 points

Leaf	Pts	Pass condition and decisive evidence
M1a	2	Identifies eligible work in at least the relevant neighboring research areas or explains a documented search gap; sources are dated and traceable.
M1b	2	Accurately extracts a concrete mechanism from a source beyond a title, abstract slogan, or speedup number.
M1c	2	Compares the source's platform, workload, objective, and quality assumptions with those of the task before borrowing an idea.
M1d	2	Explains which aspect of the prior mechanism transfers across domains and what adaptation the current task requires.
M1e	2	Identifies a specific unresolved gap or combination opportunity relative to eligible prior work; the judge may use the expert reference only to check its technical plausibility.

Cross-domain means a relevant transfer across algorithm, compiler, runtime, systems, or architecture practice; it does not reward collecting unrelated papers. A post-cutoff publication is not eligible prior art. A proposal may pass other method dimensions even if its literature section is weak.

M2. Domain-spanning cross-stack design — 30 points

Leaf	Pts	Pass condition and decisive evidence
M2a	4	Identifies the mechanism's relevant stack layers and one explicit causal path connecting them to the task objective; it need not modify every layer.
M2b	4	Makes algorithmic and representation or lowering choices compatible, including required numerical, causal, or model-state semantics.
M2c	5	Specifies the data flow and interfaces crossing changed components, including the form, owner, and direction of exchanged state.
M2d	5	Makes runtime, kernel, and platform assumptions jointly executable under the assigned hardware, resource, and tool contract.
M2e	5	Resolves timing, granularity, ordering, and state-lifetime dependencies across layers rather than proposing independently plausible but conflicting optimizations.
M2f	4	Accounts for new conversion, transfer, storage, synchronization, control, and recomputation costs introduced by the cross-stack mechanism.
M2g	3	States the end-to-end invariant or quality condition that must hold when all changed layers are composed.

A layer-by-layer shopping list without an integrated data flow fails M2c and M2e. Correctly limiting the design to two coupled layers can earn full credit if it explains the impact on untouched layers and complete-system costs. Hardware modeled in a task must be labeled modeled; a physical GPU feature may not be assumed to be modifiable when the task forbids it.

M3. Novelty and research contribution — 30 points

Leaf	Pts	Pass condition and decisive evidence
M3a	5	Positions the proposed mechanism against the closest eligible prior work using specific technical similarities and differences; the judge may consult the expert comparator without requiring the agent to have seen it.
M3b	6	States a concrete mechanistic difference, not merely a new name, broader ambition, or unsubstantiated firstness claim.
M3c	6	Changes a decision rule, data path, representation, numerical method, or other causal mechanism beyond routine parameter tuning or a trivial composition.
M3d	6	Gives a task-specific causal argument for why that difference could improve the frozen objective while preserving its gates.
M3e	4	Names a substantive limitation, cost, or regime where the new mechanism may lose to a simpler alternative.
M3f	3	Proposes an ablation or discriminating comparison that isolates the claimed novel mechanism from implementation details or parameter count.

Matching the expert's valid research idea can earn full credit if the agent independently specifies and justifies the mechanism under its assigned information condition; agreement with the expert is not itself evidence. An alternative to the expert can also earn full credit. A novel name attached to static rules or a

parameter sweep fails M3b and M3c. Novelty is assessed only against eligible pre-cutoff work, not later papers.

M4. Executability and semantic precision — 30 points

Leaf	Pts	Pass condition and decisive evidence
M4a	6	Defines the actor, observed state, trigger, action, and update rule for the central mechanism so an implementer knows what happens and when.
M4b	5	Specifies required data structures, layouts, interfaces, and ownership at component boundaries.
M4c	4	Defines ordering, state lifetime, concurrency or deadline behavior, and failure handling where these affect correctness.
M4d	4	Provides staged implementation milestones with an independently runnable or checkable intermediate artifact.
M4e	4	States how resource, numerical, causal, and quality constraints will be enforced or detected during implementation.
M4f	3	Describes a fallback or adaptation when a pivotal assumption fails without silently replacing the research mechanism.
M4g	4	Specifies decisive validation experiments with controls, measurements, and a predicted observation that could falsify the mechanism.

"Adaptive scheduling" without a concrete decision rule fails M4a; "cross-layer coordination" without an interface fails M4b. A proposal need not include final code, but another implementer must be able to start and test it. A well-specified method is not penalized here solely because its later implementation failed.

B.4 Other Prompt Cards

P01 | GPT-5.6 Pro stage-artifact judge prompts

Shared judge instruction

You are evaluating one frozen SAR-Arena stage-end artifact against the supplied hierarchical rubric. The rubric consists of weighted, binary leaves. Judge the agent's own submitted artifact, using its stage-bounded trajectory only to corroborate cited events or detect contradiction. The reviewed expert annotation is a comparator, not a required wording or unique solution. A technically valid alternative can pass every applicable leaf. For each leaf, inspect its full affirmative condition independently and return pass or fail; award all of that leaf's points only on pass. Missing or merely implied evidence fails. Partial credit arises from other passed leaves, never from a discretionary fractional leaf mark. A later engineering result cannot repair an earlier submission. Do not reward model identity, writing style, token count, or agreement with the expert by itself. Do not invent a paper, trace, measurement, or quotation.

Search only paper versions publicly available on or before 2026-08-15. Verify the version date and compare hardware, workload, objective, and measurement conditions before using a source. Post-cutoff work cannot establish prior art or novelty. If a required evaluator input is missing, truncated, or corrupted, return input_error for the affected evaluation rather than treating an evaluator defect as missing agent work. Preserve the original package ID and stage boundary. For every leaf return ID, maximum points, verdict 0 or 1, awarded points, exact submission span, corroborating trace or source IDs when required, and a concise reason explaining the threshold decision. Return each dimension subtotal, total out of 100, input-quality flags, cutoff checks, and any material supported alternative to the expert reference. Check that the leaf points sum to the dimension weights and that the dimension subtotals sum to the total. Do not silently change a leaf weight or waive one because the reference solution did not use that route.

Bottleneck-analysis judge

Inputs: public task contract and task addendum; R01 and the D1–D5 weighted-leaf tables; reviewed expert diagnosis; frozen bottleneck-analysis submission; compressed visible trajectory ending at the diagnosis boundary; and eligible search results. Grade D1a–D5h against the report's own literature analysis, profiler design and measurement, workload coverage, multi-metric evidence, and causal usefulness. Use a trace to verify an observation the report cites, not to write a diagnosis the report omitted. An openly scoped pilot may pass its applicable coverage leaves while failing the leaves that require complete major-regime measurement. A citation without comparable setting does not establish a task bottleneck. Return the shared schema with the five subtotals (10, 20, 20, 20, 30) and the 0–100 total.

Method-design judge

Inputs: public task contract and task addendum; R01 and the M1–M4 weighted-leaf tables; reviewed expert proposal and available diagnosis; frozen method-design submission; compressed visible trajectory ending at the design boundary; and eligible search results. Grade M1a–M4g against the proposal's own source analysis, cross-stack mechanism, novelty relative to eligible prior work, and executable specification. Allow a supported alternative to the expert proposal; neither similarity to nor difference from that proposal determines a leaf by itself. Do not infer implementability from later success or invalidate a complete design solely because later engineering failed. Return the shared schema with four subtotals (10, 30, 30, 30) and the 0–100 total.

P02 | Stage-bounded trajectory-compression prompt

Compress only the supplied visible messages, tool calls and results, code or configuration changes, source excerpts, measurements, and submission events through the specified stage boundary. Do not reconstruct hidden chain of thought or infer intentions not recorded. Preserve timestamps or event IDs; workload, hardware and software conditions; metric units and aggregation; failed experiments; contradictory evidence; literature citations and version dates; substantive changes to the mechanism; unresolved limitations; and the exact stage-end artifact ID. Mark a quotation as verbatim and any condensation as paraphrase. Separate agent assertions from tool observations and independent evaluation. Never add later-stage outcomes to an earlier judge package, repair an argument the agent omitted, or drop an unfavorable result for brevity. Output a chronological summary, an evidence index mapping each central claim to original event ranges, a list of omitted material, the stage boundary and source hashes, and an input-quality warning if essential evidence cannot be preserved.

P03 | Benchmark harness instruction and configuration disclosure

Benchmark-owned instruction
Run exactly one agent under the assigned model–Harness configuration. The benchmark supplies a versioned public task contract, workspace, allowed paper-search channel, condition-specific expert material, engineering time budget, and submission paths. Native Harness system instructions remain those of the pinned Codex or Claude Code release; this benchmark instruction does not claim to reproduce or override them. External retrieval is limited to eligible papers public by 2026-08-15. The agent may use local terminal, files, and task tools within the resource manifest, but may not delegate, inspect hidden evaluator data, access another run, or change the frozen model, workload, evaluator, or resource allocation. The maximum supported reasoning-strength setting for the selected endpoint is used and its literal accepted value is logged. Analysis and method stages have no hard time or token ceiling. The engineering clock starts at the engineering-start event and runs for eight hours, or twenty-four hours on the declared extended full-cycle tasks. Public test waiting and engineering experiments consume this budget; independent final verification and offline snapshot scoring do not. Record model snapshot, Harness version, reasoning option, paper-search policy, task and condition view, tool permissions, stage boundaries, token receipts by stage, and candidate hashes.

Configuration matrix
Claude Opus 5 with Claude Code; GPT-5.6-Sol with Codex; Kimi-K3 with Claude Code; GLM-5.3 with Claude Code; DeepSeek-V4-Pro-0813 with Claude Code; DeepSeek-V4-Flash-0731 with Claude Code. Every model–task–setting combination is one independent run. All six configurations use one agent and the endpoint's highest available reasoning strength. Do not equate different providers' setting names without the logged endpoint configuration.

P04 | Failure-detection prompt bundle

Shared auditor instruction
Audit one selected failure category in one run using the public task contract, condition, expert material actually supplied, visible timestamped trace, tool returns, code changes, experiment and job logs, and budget record. The original scan had model identity masked but final scores visible; scores may be present in the input, but a low score or failed task is never evidence of a particular failure. Do not infer hidden reasoning. First decide whether the triggering activity occurred. Use present only with existing event IDs and short exact quotations satisfying that category's necessary evidence. Use absent

only with adequate observation and no qualifying event; use not_applicable when there was no trigger; use insufficient_evidence when necessary records are missing. Retain counterevidence and later repairs. Repetitions from one continuing cause form one episode; a run contributes at most one count per category to incidence. Return category, verdict, eligibility, affected-run flag, episode ranges and quotations, counterevidence, resolution, missing evidence, and concise rationale. The externally defined aggregation pool determines denominators; do not choose an error quota or infer one from model reputation.

B1. Literature measurements detached from their setting
Require the cited numerical claim, original hardware, workload and statistic, current task conditions, and an adopted inference that treats the prior number as task evidence without a valid transfer argument. A qualified analogy, explicitly calibrated scaling, or a hypothesis awaiting measurement is not enough. If the source conditions are unavailable, return insufficient_evidence.

B2. Incomplete workload coverage
Require the specified target distribution, the actually inspected regimes, a consequential omitted regime, and a diagnosis generalized beyond observed coverage. A clearly scoped pilot or justified sampling plan is not enough.

B3. Incorrect profiler
Require the profiler code or configuration, the intended quantity, the specific timer, synchronization, counter or aggregation error, the resulting invalid observation, and the agent's continued reliance on it. A bug found and corrected before its result is used is not this failure.

D1. Concept substituted for mechanism
Require a submitted proposal or adopted design that uses a label such as adaptive scheduling without defining the actor, observed information, trigger and action needed for an executable decision. Mere brevity is not failure when a referenced design specifies the mechanism elsewhere in the frozen submission.

D2. Incompatible cross-stack components
Require at least two proposed components with concrete conflicting layout, interface, granularity, ordering or state-lifetime demands, plus evidence that the agent adopted them together without a compatible integration path. Separate alternatives are not a conflict.

E1. Repeated tradeoff oscillation
Require a real conflicting cost tradeoff, repeated parameter switches or rollback between its ends, and the absence of a regime model or reconsideration of the mechanism despite informative evidence. One ordinary comparison or justified rollback is not enough.

E2. Unacknowledged mechanism substitution
Require a previously explicit mechanism, an implementation obstacle, a simpler replacement with materially different behavior, and later reporting or evaluation as though the original mechanism remained intact. An openly documented and re-evaluated revision is not this failure.

E3. Nonincremental change and debugging
Require a large dependent change without a runnable, testable intermediate state, followed by broad patching of the combined implementation instead of isolating a failing component. A large change supported by staged checks is not enough.

E4. Premature termination of a long experiment
Require evidence that a long experiment was progressing normally, sufficient budget remained, no stop rule or resource fault applied, the agent stopped it mainly for being slow, and then adopted a simpler proxy. A completed run, stalled or failed job, resource limit, preregistered stop, or passive snapshot archival is not this failure.

B.5 Rubric Development and Validation Scope

GPT-5.6 Pro assisted in extracting evaluation criteria from domain-conference reviews, including MICRO, ICCAD, HPCA, DAC, ISCA, NeurIPS, ICML, and ICLR. Domain experts iteratively checked ten sample artifacts during rubric development. This establishes expert involvement in rubric construction, not independent human agreement on the reported experiment's ratings. Each frozen artifact is judged three times by GPT-5.6 Pro and the scores are averaged. Independent expert rescoring of a stratified sample, including supported alternatives to the reference, remains necessary to quantify rating error and novelty-assessment reliability.

B.6 ENGINEERING-SCORE INTERPOLATION

All three research settings share the same engineering-score normalization for each task at a fixed engineering-iteration budget. The strongest prior-work baseline and each submission are each measured five times under the same task protocol. Let $b > 0$ be the baseline's geometric-mean primary metric and $m = (\prod_{i=1}^{5} m_i)^{1/5} > 0$ the candidate's. Define the oriented benefit ratio $q = m/b$ when larger values are better and $q = b/m$ when smaller values are better. Let $q_\star$ be the maximum of 1 and the benefit ratios of all valid results, pooled across the evaluated models and the three research settings. This single upper anchor is used for every result in the task-budget group. For the evaluated cohort, the engineering score is

$$E(q) = \begin{cases} 100\dfrac{\log q}{\log q_\star}, & 1 < q \le q_\star, \quad q_\star > 1, \\ 0, & \text{otherwise.} \end{cases}$$

The baseline has $q = 1$ and receives zero. Any valid candidate with a strictly better geometric mean qualifies; there is no additional minimum-effect or confidence-interval gate. When $q_\star > 1$, the best valid tested-model result receives 100 and $q = \sqrt{q_\star}$ receives 50. Matching or underperforming the baseline counts as engineering failure and receives zero. If no valid result in any of the three settings exceeds the baseline, every engineering score for that task and budget is zero; no logarithmic normalization is applied. Constraint violations, missing runnable code, or the absence of a valid result also receive zero engineering points. Diagnosis and method scores are retained, and the aggregate is calculated with the same weights.

The upper anchor is specific to the task, evaluated model cohort, and time budget, and is shared across the three settings. A given valid performance therefore receives the same engineering score regardless of setting; the best result within an individual setting need not receive 100. For subsequent submissions to the same benchmark version, the reported raw ratio and the present cohort's upper anchor should both be retained; a stronger submission may exceed 100 on that frozen scale. Recomputing the anchor defines a new score version and requires rescoring the earlier cohort. Comparisons across engineering budgets use raw metrics or ratios to the same baseline when the upper anchors differ. Five repeats reduce sensitivity to individual measurements but do not establish statistical separation; a ratio only marginally above one remains sensitive to measurement error.

C ADDITIONAL EXPERIMENTAL RESULTS

Stage token means use the stages actually performed: ten analysis, twenty design, and thirty engineering runs per model. Table 7 reports every stage score; Table 8 summarizes the engineering score by environment.

Stage ratings and engineering outcomes. In a post-hoc descriptive check, all 47 F/G proposals with $M \ge 75$ have positive engineering scores. Among the 14 F runs with $E = 0$, the largest diagnosis and method scores are 64.3 and 59.5. The threshold 75 is a transparent sensitivity check, not a preregistered definition of a strong proposal. Across all 60 F runs, the pooled Pearson correlations computed from unrounded scores are $r(D, E) = 0.919$ and $r(M, E) = 0.913$. These summaries do not establish within-configuration prediction or causal effects of better diagnoses or proposals.

Table 5: Paired engineering-score transitions across the same 60 model–task combinations. A positive score means valid performance above the task baseline; zero also includes valid results that do not exceed it. The last column is the mean score change over all 60 pairs.

Conditions	$+\to+$	$0\to+$	$+\to0$	$0\to0$	Mean ΔE
F $\to$ G	43	5	3	9	$+10.05$
G $\to$ O	45	4	3	8	$+13.43$
F $\to$ O	44	5	2	9	$+23.47$

For F $\to$ O, pairs positive in both conditions contribute $+18.58$ points to the overall mean change; zero-to-positive pairs contribute $+5.64$, and positive-to-zero pairs contribute -0.75. The contributions sum to the unrounded $+23.47$-point change.

Table 6: Mean total agent tokens per performed stage, in millions. Totals include input, cache reads, and output with reasoning tokens; judge and evaluator calls are excluded.

	Mean total tokens (millions)		
Model	Analysis (10)	Design (20)	Engineering (30)
Claude Opus 5	2.06	2.36	14.62
GPT-5.6-Sol	1.64	1.57	11.98
Kimi-K3	2.18	2.33	15.03
GLM-5.3	2.22	2.21	18.23
DeepSeek-V4-Pro-0813	2.49	2.41	19.80
DeepSeek-V4-Flash-0731	2.74	2.62	25.61

Table 7: All eight scores for each model and environment. Scores are shown to one decimal; aggregation uses unrounded values.

	Engineering Optimization	Guided Research			Full-Cycle Research			
Model	Eng.	Method	Eng.	Total	Diag.	Method	Eng.	Total
T01 Streaming video understanding								
Claude Opus 5	92.6	95.0	100.0	98.8	87.9	92.4	92.9	91.8
GPT-5.6-Sol	91.4	82.1	73.6	75.8	86.3	83.1	73.9	78.2
Kimi-K3	88.9	88.5	86.7	87.2	74.3	57.1	57.9	61.0
GLM-5.3	90.6	73.1	59.5	62.9	65.1	53.8	43.9	50.1
DeepSeek-V4-Pro-0813	82.7	67.3	53.0	56.6	58.1	46.5	26.8	37.0
DeepSeek-V4-Flash-0731	0.0	47.7	0.0	11.9	38.6	30.2	0.2	13.9
T02 Coding-agent serving								
Claude Opus 5	69.0	94.1	96.1	95.6	90.0	89.5	100.0	95.9
GPT-5.6-Sol	74.0	80.1	65.6	69.2	84.7	84.7	74.7	78.7
Kimi-K3	82.4	75.7	56.9	61.6	62.3	61.7	39.1	48.2
GLM-5.3	84.2	76.9	65.0	68.0	65.7	57.6	38.2	47.6
DeepSeek-V4-Pro-0813	68.8	69.7	51.9	56.4	61.1	54.9	30.5	41.5
DeepSeek-V4-Flash-0731	7.9	47.3	0.0	11.8	35.1	29.9	0.0	13.0

Table 7: All eight scores by model and environment (continued).

	Engineering Optimization	Guided Research			Full-Cycle Research			
Model	Eng.	Method	Eng.	Total	Diag.	Method	Eng.	Total
T03 Full-duplex speech								
Claude Opus 5	93.0	91.8	93.0	92.7	82.6	86.7	85.5	85.2
GPT-5.6-Sol	100.0	83.5	77.5	79.0	87.1	71.7	77.5	78.3
Kimi-K3	93.0	77.5	69.0	71.1	70.0	70.3	59.9	64.0
GLM-5.3	93.0	67.7	59.9	61.8	67.7	60.5	50.0	55.6
DeepSeek-V4-Pro-0813	69.0	69.1	50.0	54.8	58.3	41.6	0.0	20.0
DeepSeek-V4-Flash-0731	0.0	48.0	0.0	12.0	39.9	22.1	0.0	12.4
T04 Audiovisual generation								
Claude Opus 5	89.7	93.7	80.9	84.1	90.3	96.0	100.0	97.3
GPT-5.6-Sol	91.8	93.7	89.0	90.2	93.9	90.0	82.9	86.5
Kimi-K3	91.4	79.1	69.5	71.9	74.3	70.9	62.0	66.2
GLM-5.3	87.4	70.3	56.6	60.0	72.1	63.9	51.3	58.0
DeepSeek-V4-Pro-0813	98.9	64.9	44.0	49.3	52.7	57.5	0.0	22.1
DeepSeek-V4-Flash-0731	0.0	46.3	0.0	11.6	42.1	36.4	0.0	15.7
T05 Interactive world model								
Claude Opus 5	84.9	92.4	85.5	87.3	80.5	90.5	89.9	88.1
GPT-5.6-Sol	100.0	92.0	85.6	87.2	90.1	89.4	87.2	88.2
Kimi-K3	86.1	78.1	72.1	73.6	75.4	73.0	56.6	63.7
GLM-5.3	87.8	69.5	0.0	17.4	59.2	57.5	32.8	43.0
DeepSeek-V4-Pro-0813	56.2	65.7	49.8	53.8	59.0	43.9	19.2	32.1
DeepSeek-V4-Flash-0731	0.0	40.7	0.0	10.2	37.4	24.8	0.0	12.4
T06 MoE CPU–GPU mapping								
Claude Opus 5	100.0	93.8	90.1	91.0	79.1	87.5	86.0	84.9
GPT-5.6-Sol	94.6	84.8	82.3	82.9	89.3	72.9	70.7	74.9
Kimi-K3	85.8	74.8	70.3	71.5	73.9	63.6	58.1	62.4
GLM-5.3	0.0	72.1	68.2	69.2	66.6	57.1	44.6	51.5
DeepSeek-V4-Pro-0813	83.6	72.2	59.3	62.5	67.8	52.7	41.6	49.0
DeepSeek-V4-Flash-0731	0.0	53.9	0.0	13.5	38.7	25.4	0.0	12.8
T07 VLA closed-loop control								
Claude Opus 5	100.0	93.0	86.9	88.4	84.7	87.9	87.6	87.1
GPT-5.6-Sol	77.7	94.4	91.8	92.4	90.1	70.7	77.5	78.7
Kimi-K3	79.3	72.8	0.0	18.2	63.3	59.5	0.0	24.6
GLM-5.3	70.9	75.4	60.8	64.4	65.5	54.1	33.2	43.8
DeepSeek-V4-Pro-0813	70.3	71.4	0.0	17.8	50.1	49.3	14.4	28.5
DeepSeek-V4-Flash-0731	0.0	38.2	0.7	10.1	35.6	19.6	0.0	11.0
T08 CPU-only MoE service								
Claude Opus 5	89.5	91.8	91.1	91.3	75.0	83.9	72.8	75.5
GPT-5.6-Sol	99.0	84.9	82.2	82.9	85.7	76.7	76.1	78.2
Kimi-K3	100.0	77.8	64.0	67.5	69.1	65.2	57.7	61.5
GLM-5.3	83.2	73.4	60.7	63.8	62.8	57.8	0.0	24.1
DeepSeek-V4-Pro-0813	0.0	71.5	44.4	51.1	64.3	56.8	0.0	24.2
DeepSeek-V4-Flash-0731	0.0	46.7	0.0	11.7	33.7	24.3	0.0	11.6
T09 NVFP4 tensor core								
Claude Opus 5	72.7	92.5	100.0	98.1	83.2	90.5	83.0	84.5
GPT-5.6-Sol	86.3	93.5	80.2	83.5	93.5	88.9	78.0	83.3
Kimi-K3	96.9	87.3	74.3	77.6	76.6	68.5	56.1	62.7
GLM-5.3	52.0	71.7	30.0	40.4	59.0	58.1	20.4	35.6
DeepSeek-V4-Pro-0813	51.5	67.9	19.8	31.8	55.7	45.4	7.0	24.4
DeepSeek-V4-Flash-0731	0.0	45.9	0.0	11.5	33.6	26.0	0.0	11.9
T10 Retrieval sparse attention								
Claude Opus 5	93.7	89.7	89.6	89.6	84.7	84.1	76.5	79.7
GPT-5.6-Sol	97.7	87.6	89.4	89.0	87.6	78.3	100.0	93.2
Kimi-K3	81.9	77.4	63.9	67.2	70.9	55.5	42.6	50.9
GLM-5.3	92.0	73.0	57.1	61.1	74.5	59.9	47.8	55.6
DeepSeek-V4-Pro-0813	83.3	67.3	42.9	49.0	55.1	47.4	23.5	34.6
DeepSeek-V4-Flash-0731	0.0	43.6	0.0	10.9	36.9	25.1	0.0	12.4

Table 8: Mean engineering score by environment and guidance setting, averaged equally over the six model–harness configurations.

	Engineering score (0–100)		
Environment	Engineering Optimization	Guided Research	Full-Cycle Research
T01 Streaming video	74.4	62.1	49.3
T02 Coding-agent serving	64.4	55.9	47.1
T03 Full-duplex speech	74.6	58.2	45.5
T04 Audiovisual generation	76.6	56.7	49.4
T05 Interactive world model	69.2	48.8	47.6
T06 MoE CPU–GPU mapping	60.7	61.7	50.2
T07 VLA closed-loop control	66.4	40.0	35.4
T08 CPU-only MoE service	61.9	57.1	34.4
T09 NVFP4 tensor core	59.9	50.7	40.8
T10 Retrieval sparse attention	74.8	57.1	48.4

C.1 Engineering Anchors and Zero-Score Outcomes

A zero engineering score combines two distinct outcomes: an invalid submission and a valid submission that does not improve on baseline. The export contains 14 invalid, 23 valid non-improving, and 143 valid improving runs. Table 9 separates these outcomes and reports each task's observed eight-hour upper anchor. The ratios use the baseline-relative utility convention q in Appendix B; they are not physical latency, throughput, area, or power measurements. The underlying baseline values, measurement repeats, and numerical quality gates are still needed to recover and audit those physical quantities.

Table 9: Eight-hour anchors and outcome counts. I/N/P denotes invalid / valid non-improving / valid improving submissions; each setting contains six runs per task. The baseline is $q = 1$, and the displayed $q_\star$ is shared across settings.

	Upper anchor	Outcome counts (I/N/P)		
Task	$q_\star$	Engineering Optimization	Guided Research	Full-Cycle Research
T01	2.345	0/1/5	0/1/5	0/0/6
T02	2.677	0/0/6	1/0/5	0/1/5
T03	2.250	0/1/5	0/1/5	0/2/4
T04	1.851	1/0/5	1/0/5	1/1/4
T05	2.582	0/1/5	0/2/4	0/1/5
T06	2.533	1/1/4	0/1/5	0/1/5
T07	2.104	0/1/5	1/1/4	1/1/4
T08	1.936	1/1/4	1/0/5	1/2/3
T09	1.848	0/1/5	0/1/5	1/0/5
T10	2.332	1/0/5	1/0/5	1/0/5
All tasks	—	4/7/49	5/7/48	5/9/46

C.2 Resources by Research Setting

Table 10 separates the performed stages within each setting. A stage total is its recorded input plus output tokens; input already includes uncached input, cache writes, and cache reads, so cached tokens are counted once. Reasoning tokens are included in output. Judge and independent evaluator usage is excluded. Dashes indicate an unperformed stage. Each cell averages ten tasks; stage means sum to the run total before rounding. Recorded pre-engineering time sums analysis and design API/tool durations. It averages 0.70 hours in Guided Research and 1.37 hours in Full-Cycle Research across configurations, compared with zero in Engineering Optimization. These records quantify additional pre-engineering work; they do not describe the code or measurements already present at engineering entry, and token counts across tokenizers are not equal amounts of computation.

Table 10: Mean agent resource use over ten tasks for each configuration and setting. Tokens are totals in millions, with no input/output split. Pre-engineering time is in hours; every row additionally has the eight-hour engineering budget.

	Total tokens (millions)				Before engineering
Setting	Analysis	Method	Engineering	Run total	Hours
Claude Opus 5					
Engineering Optimization	—	—	14.26	14.26	0.00
Guided Research	—	2.19	14.85	17.04	0.87
Full-Cycle Research	2.06	2.53	14.75	19.35	1.64
GPT-5.6-Sol					
Engineering Optimization	—	—	11.69	11.69	0.00
Guided Research	—	1.46	12.18	13.64	0.49
Full-Cycle Research	1.64	1.68	12.08	15.39	0.97
Kimi-K3					
Engineering Optimization	—	—	14.76	14.76	0.00
Guided Research	—	2.23	15.18	17.41	1.05
Full-Cycle Research	2.18	2.42	15.15	19.76	1.89
GLM-5.3					
Engineering Optimization	—	—	18.09	18.09	0.00
Guided Research	—	2.04	18.27	20.31	0.58
Full-Cycle Research	2.22	2.37	18.33	22.92	1.21
DeepSeek-V4-Pro-0813					
Engineering Optimization	—	—	19.56	19.56	0.00
Guided Research	—	2.16	19.76	21.92	0.64
Full-Cycle Research	2.49	2.65	20.08	25.22	1.34
DeepSeek-V4-Flash-0731					
Engineering Optimization	—	—	25.35	25.35	0.00
Guided Research	—	2.43	25.82	28.25	0.58
Full-Cycle Research	2.74	2.81	25.66	31.21	1.17

C.3 Paired Eight- and Twenty-Four-Hour Checkpoints

Table 11 reports the baseline-relative utilities of the nominated artifacts on all 18 continuing Full-Cycle Research trajectories. These ratios remain comparable when the score anchor changes between the eight-hour main table and the 24-hour plot. Invalid artifacts have no valid utility ratio and are marked I; $q = 1$ denotes a valid baseline-matching artifact. At eight hours there are two invalid, one valid baseline-matching, and 15 baseline-beating artifacts. At 24 hours all 18 are valid, with one matching baseline and 17 exceeding it. No repeated agent runs or confidence intervals are implied by these checkpoint pairs.

Table 11: Paired utilities on continuing Full-Cycle Research trajectories. $\Delta q = 100(q_{24}/q_8 - 1)$ is shown only when both artifacts are valid. These are utility-ratio changes, not percentage-point differences on the normalized engineering scale.

	Baseline-relative utility		Relative change
Model	q_8	q_{24}	Δq (%)
T02 Coding-agent serving			
Claude Opus 5	2.677	2.700	+0.9
GPT-5.6-Sol	2.087	2.123	+1.7
Kimi-K3	1.469	1.514	+3.1
GLM-5.3	1.457	1.501	+3.0
DeepSeek-V4-Pro-0813	1.350	1.384	+2.5
DeepSeek-V4-Flash-0731	1.000	1.041	+4.1
T09 NVFP4 tensor core			
Claude Opus 5	1.665	1.714	+2.9
GPT-5.6-Sol	1.615	1.654	+2.4
Kimi-K3	1.412	1.501	+6.3
GLM-5.3	1.133	1.148	+1.3
DeepSeek-V4-Pro-0813	1.044	1.059	+1.4
DeepSeek-V4-Flash-0731	I	1.000	—
T10 Retrieval sparse attention			
Claude Opus 5	1.912	2.048	+7.1
GPT-5.6-Sol	2.332	2.460	+5.5
Kimi-K3	1.435	1.549	+7.9
GLM-5.3	1.499	1.571	+4.8
DeepSeek-V4-Pro-0813	1.220	1.314	+7.7
DeepSeek-V4-Flash-0731	I	1.039	—

D Author-Side Diagnosis and Proposal Examples

These examples document candidate mechanisms for T03 (full-duplex speech; source identifier R10) and T04 (audiovisual generation; O06). They summarize author-side diagnostic reports and proposal queues dated September 26, 2026. They are not agent trajectories, verified optimization gains, or formal baseline measurements, and are not used to recompute the reported experiment scores. The source queues retain detailed implementation steps, falsification tests, resource accounting, and stopping conditions. Their planning estimates are distinct from the benchmark's eight-hour engineering budget.

D.1 Full-Duplex Speech: From Local Timing to Service Mechanisms

Observed evidence and its boundary. Three distinct single-session diagnostic experiments yield eight speaking-unit observations: externally synchronized generation takes 203.53–590.07 ms and prefill 44.62–53.04 ms. In the paced replay, three release-to-local-PCM-commit intervals are 495.13, 506.74, and 356.66 ms. The first speaking block contains 680 ms of leading zeros. Local PCM completion therefore cannot stand in for a semantically valid response played by an independent client. Internal LLM/TTS/codec counters were not individually synchronized, so the aggregate generation time does not identify a particular hot kernel. No optimized candidate was tested in these diagnostics; concurrent service baselines, client playback timing, and independent behavior acceptance remain prerequisites for a service-level claim.

Causal input/output granularity (I1). Separate input-submission granularity from generation triggers while preserving native turn tokens and causal audio release. Compare official units, finer internal submission, and joint changes on paired streams. Reject changes that break position/state semantics, audio quality, or full event-to-response latency.

Submodule precision budgets (I2). Assign separate, training-free precision and error budgets to the audio encoder, LLM, and TTS/codec. Compare uniform precision, single-module changes, and their combination, charging conversion and dequantization. A faster hot segment qualifies only if content, voice, silence, interruption, and long-session memory pass independently.

Speaking-token and acoustic-block boundaries (I3). Distinguish the duplex LLM's speaking-token cap from TTS token limits and codec chunk/flush boundaries. Test each separately before combining them. Check missing words, short final chunks, duration, and interruption; shifting content to later blocks is not a first-response improvement.

Semantic state retention and reconstruction (I4). Separate system prompts, recent audio, discourse references, and voice state, then trim or reconstruct only at legal turn boundaries. Make KV/RoPE offsets and codec-cache lifetimes explicit. Compare sliding windows, trimming, and reconstruction on long-session references, charging migration and recovery costs.

Progressive PCM submission (I5). Test whether already determined samples can be submitted earlier while preserving sample order, media timestamps, playback buffering, and semantic quality. Compare whole-block submission, corrected timestamps alone, and incremental submission. Deleting leading silence or treating the first nonzero sample as a valid response is inadmissible.

Generation attribution and decode graphs (II1). First synchronize LLM decode, TTS preparation, acoustic generation, and codec boundaries, measuring instrumentation overhead. Only then compare eager execution, ordinary graph capture, and length/KV-bucketed graphs. Stop if the apparent hotspot or end-to-end benefit disappears after overhead and shape-switch costs.

Short-block TTS/codec execution (II2). If II1 identifies these stages as material, fuse layout operations or capture stable short-block graphs without changing audio semantics. Evaluate ordinary compilation against specialized kernels on full and short final blocks; include cache initialization, fallback, and writeout.

Audio-prefill fusion (II3). Audit audio preparation, embeddings, layout, and transfer before combining preallocation and fusion. The observed 45–53 ms prefill per unit bounds the local opportunity, not the service gain. Stop if prefill is outside the service critical path or fusion harms concurrency.

Cache layout and short kernels (II4). Compact or rearrange per-session KV and audio/TTS state without changing retained context. Separate physical-layout changes from I4's semantic trimming. Check position consistency, fragmentation, HBM traffic, relocation overhead, and long-session memory before claiming latency benefits.

Playback-deadline scheduling (III1). Schedule the prefill–LLM–TTS–PCM dependency graph using trusted playback deadlines with starvation protection. Compare FIFO, periodic, and deadline policies under staggered arrivals and interruptions. Lower average queueing is insufficient if tail response, continuity, or refusal behavior worsens.

Bounded, stage-specific microbatching (III2). Form separate bounded-wait batches for prefill, LLM decode, and TTS while maintaining each session's causality. Compare serial, fixed-batch, and stage-specific policies across concurrency levels. Charge batch-formation delay and keep its effect separate from deadline ordering.

Per-session arenas and placement (III3). Share model weights but isolate session state, placing hot state on GPU and cold state in pinned host memory or reconstructing it. Compare ordinary offload and isolated arenas before combining them. Single-session memory peaks cannot establish concurrency capacity; measure marginal state, transfer tails, OOMs, and memory quality.

Deployment-period residency (III4). Evaluate weight residency, warmup, and reusable graph artifacts under a frozen deployment period. The diagnostics measure loading and preparation, but these costs need common amortization for all candidates. Stop if apparent savings merely move initialization outside the accounting boundary or displace active-session state.

Decision order. Start with II1. Advance to II2 or I2 only after attribution and quality checks; investigate I5, III1, and III2 after establishing trusted client timing. State-management proposals require observed long-session pressure. Adjacent mechanisms use baseline, A, B, and A+B comparisons: token decisions versus PCM submission; semantic retention versus layout versus placement; and scheduling order versus batch formation. Local speedups cannot be added to predict the service p95.

D.2 Audiovisual Generation: Complete-File Optimization

Observed evidence and its boundary. Two same-configuration diagnostics, covering a bell-impact prompt and a spoken greeting, take 224.197 and 225.994 s from model loading completion to media save. Combined model movement and 50-step sampling take 183.706 and 183.764 s (81.94% and 81.31%); subsequent residency transitions take 14.718 and 15.350 s, and video-VAE decoding plus device-to-host transfer takes 15.408 and 15.366 s. These combined intervals do not isolate compute from movement. The diagnostic output is 704×704 with 121 frames, within the named Ovi preset; it is not an independently validated five-category benchmark result. Joint media quality and instrumentation perturbation were not established, and older functional timings use different cache/instrumentation conditions.

Measurement prerequisites (M0–M1). Separate model transfer, positive/negative forward passes, scheduler work, VAE computation, and host transfer using paired instrumented/uninstrumented runs. Cross content category with first-run/cache state before attributing a slow run to its prompt. These are attribution experiments, not optimization gains. Formal evaluation pairs prompt and seed within each category and weights five categories equally; playable containers alone do not establish content or synchronization quality.

Joint guidance batching (A1). Batch positive and negative conditional forwards with matching shapes while retaining both audio/video predictions and the original guidance combination. Test one-step tensor error and peak memory before complete media. Stop on OOM, slower sampling, or any joint-quality failure.

Joint solver trajectories (A2). Search a bounded set of paired audio/video solver schedules to reduce fusion calls without changing the source model or output contract. Validate speech, impacts, music, complex motion, and synchronization separately. Fewer steps qualify only after all independent quality gates pass.

Cross-step feature reuse (A3). Cache slowly changing features with explicit error and invalidation triggers for both modalities. Measure activation changes before paired complete-file tests. Include cache construction, memory, and invalidation cost; stop when quality or cross-category stability fails.

Fixed-shape graph capture (C1). Capture stable forward and scheduler regions only if profiling shows significant launch or host-dispatch cost. Compare identical seeds and complete-file outputs, including compilation, warmup, graph memory, and state safety. A faster steady-state segment alone is insufficient.

Layer-specific precision (C2). Select supported low-precision paths using measured hotspots and error sensitivity while retaining precision in sensitive normalization and fusion. Check layer and single-step errors before complete-media quality. Charge conversion, persistence, loading, dequantization, and compilation under one deployment period.

Model-lifecycle residency (R1). Use the text encoder–fusion model–dual-VAE peak-memory timeline to choose feasible resident sets, staged transfer, and pinned memory. Change one transfer at a time. Count reloads, synchronization, and host/GPU peaks; stop if fragmentation or later stages erase the complete-file gain.

Offload cleanup (R5). Separate model transfer, device synchronization, cache release, and IPC cleanup before modifying their frequency. Preserve required synchronization and record retained memory. A shorter cleanup interval is rejected if it causes races, later OOMs, or longer complete-file execution.

Controlled prefetch and overlap (R2). Overlap only dependency-free next-stage weights or buffers with current work. Validate overlap with device events, PCIe traffic, and complete-file timing. Extra

pinned memory, streams, and peak residency count; moving transfer delay to another interval is not improvement.

VAE-to-encoder pipelining (R3). Use bounded producer–consumer buffers to overlap host transfer, format conversion, and encoding; chunk VAE output only when frame order and boundary quality remain valid. The endpoint remains the final playable file, including CPU, RAM, disk, and synchronization costs.

Attention layouts and tiling (K1). Optimize fusion attention only after kernel timing, occupancy, and traffic establish a material hotspot. Compare kernel, sampling, and complete-file outcomes with the same numerical and quality checks. Register spills, workspace growth, or absent end-to-end gains reject the candidate.

Dual-scheduler fusion (K2). Fuse small updates or eliminate temporary tensors for both modalities while retaining solver equations and synchronization. Validate one step, the complete trajectory, and media quality. Stop when scheduler cost is negligible or accumulated error invalidates the file.

Video-VAE layout and tiling (K3). Separate VAE compute from host transfer before altering temporal tiling, convolution layouts, or activation residency. Check pixel/perceptual differences and tile boundaries using the same latent input. Charge recomputation, transfer, and writeout; prefer R3 if transfer dominates.

Loading and compilation amortization (R4). Separate storage transfer, initialization, and cache reuse over a frozen deployment period. Cross cold/warm runs and candidate order to investigate first-run anomalies. Reuse is admissible only with common accounting for startup, compilation, memory, complete-file quality, and failures.

Acceptance boundary. Mechanism selection follows the decomposed measurements: compute-dominated intervals motivate A/C/K changes, movement or cleanup motivates R1/R2/R5, and VAE/-transfer attribution separates K3 from R3. Quality-unvalidated approximations remain proposals. No mechanism in this queue has an established optimization gain in the supplied diagnostic reports.

E Failure Coding Protocol

Human inspection of ten randomly sampled compressed trajectories and result files elicited the categories. GPT-5.6 Pro subsequently scanned all 180 runs with category-specific checklists, including the sampled ten. Model identities were masked, but final scores remained visible. An audit decision uses the task card, expert input if any, visible timestamped messages and tool returns, code changes, experiment logs, and remaining engineering budget. The checklist first asks whether the trajectory offered a trigger opportunity; an inapplicable or insufficiently evidenced case is not counted as absent. Each positive decision requires a trace-linked event and counterevidence check, and each run contributes at most one count per category. Appendix B.4 presents the nine categories in one card. The written card is a current audit specification; matching it to the executed judge calls still requires the archived run records.

Bottleneck analysis. B1 (misapplied literature measurements) requires a numerical claim imported across mismatched hardware, workload, or statistical conventions and then used to justify a task diagnosis. B2 (incomplete workload coverage) requires a conclusion generalized beyond the input regimes or execution phases actually studied. B3 (faulty profiling) requires a concrete error in timing boundaries, synchronization, event accounting, or aggregation that affects a downstream inference. An explicitly scoped pilot or a profiler error corrected before use does not automatically qualify.

Method design. D1 (concept without mechanism) applies when a final proposal names an approach but leaves its actor, trigger, information source, or operation unspecified. D2 (incompatible cross-stack design) requires concrete proposed components whose layouts, schedules, state lifetimes, or interfaces conflict without a workable integration path. A broad brainstorming phrase alone is not D2 if no multi-layer mechanism was committed.

Engineering iteration under expert proposals. The reported E-category counts cover only Engineering Optimization runs; they do not estimate the prevalence of engineering failures in Full-Cycle or Guided Research. E1 (trade-off cycling) requires repeated switching between conflicting choices without locating their operating regimes or revising the mechanism. E2 (unacknowledged substitu-

tion) requires replacing a proposed mechanism with an easier surrogate while still evaluating it as the original. E3 (nonincremental debugging) requires dependent, large changes without runnable intermediate versions followed by unfocused patching of the combined failure. E4 (premature termination) requires stopping a healthy long experiment despite sufficient remaining budget and no evidence-based stopping rule. A justified rollback, an explicit scope change, or a stalled experiment is excluded as appropriate.

F LIMITATIONS

Coverage and evaluator scope. The ten environments sample recent AI systems and architecture research rather than the full field; shared workloads, model families, and optimization mechanisms also limit task independence. The August 15, 2026 paper-retrieval cutoff does not establish that agents had never encountered related methods during pretraining. New platforms can change bottlenecks and baseline strength, so updating a task requires renewed validation of its workload, constraints, and evaluator. Simulation, synthesis, or calibrated prediction does not establish fabricated-device performance; simulated control does not establish physical deployment; and service throughput does not measure success on the served application. The normalized 100-point anchor identifies the best valid result in this cohort, not a universal ceiling, and small apparent gains near the baseline are sensitive to measurement error.

Experimental comparisons. Each model–harness configuration runs once per task and setting, so differences do not estimate repeated-run variance, and model effects cannot be separated from harness effects. Analysis and design may include experiments whose files and measurements carry into engineering; equal engineering clocks therefore do not equalize total work or starting artifacts. Guidance changes both information and responsibility, limiting causal attribution to an isolated ability. The 24-hour study covers three selected Full-Cycle Research tasks and cannot determine a suite-wide scaling law.

Judgment and qualitative analysis. Stage-end ratings depend on rubric calibration, expert-reference coverage, rater behavior, and information lost in compressed trajectories. Rubric development included expert inspection of ten samples, but no independent human rescoring study of the reported stage artifacts is provided. Three calls to the same judge do not establish agreement with domain experts, particularly for novelty and supported alternatives to the reference. The current archive does not reconcile criterion-level aggregation with the exported dimension-level stage scores. A valid alternative may require evidence absent from the reference and compressed record. Failure categories were induced from ten sampled records that were also included in the final scan; GPT-5.6 Pro checked all runs with model identity masked but final scores visible. This makes the taxonomy exploratory and leaves possible outcome-aware labeling bias. Categories can co-occur; B events are audited across all conditions, whereas E events are audited only in Engineering Optimization. Their fixed audit-pool counts are not opportunity-adjusted probabilities and do not establish that an observed event caused the final score.