Hydra: Task-Aware Routing Across Language Models

Mindverse
Technical White Paper
Authors: Nils Rump and Noel Lorenz

11 September 2026

Abstract

HYDRA is a configurable routing architecture that separates task interpretation, model eligibility, route selection, and execution. Its standard configuration is quality-oriented; users can adjust selection priorities and apply portfolio constraints such as an OSS filter. A repeated internal 100-task benchmark compares four Hydra settings with six fixed model settings. Hydra adjustment D records 85.43 quality points at $0.02062 generation cost per 100 planned tasks. GLM 5.3 Flash records 87.95 points at $0.11406: D's observed generation cost is 81.9% lower, alongside a 2.52-point lower quality mean. The measurements illustrate quality–resource trade-offs, with historical controls, unequal scored coverage, and incomplete resource records limiting generalisation.

Keywords: model routing, language models, task interpretation, configurable inference, multi-model systems.

1 Introduction

Model selection depends on both the request and its execution environment. Tasks differ in reasoning, modality, and output requirements; users may also restrict the available portfolio. Hydra separates these concerns: the Coordinator interprets the task, eligibility defines admissible candidates, the Head selects a route, and orchestration executes it.

The standard configuration prioritises quality. Users can adjust preferences or restrict the portfolio without changing this architecture. The contribution is a controllable selection process whose outcomes are evaluated jointly in quality, generation cost, time, and output usage. The current benchmark illustrates several operating points of that process.

2 Related Work

Cost-aware cascades and learned routers investigate how model selection can reduce inference expenditure while retaining useful answer quality. FrugalGPT studies cascaded selection, while RouteLLM learns routing preferences from comparative feedback [2, 5]. AutoMix explores routing and self-verification, and LLM-Blender investigates combining model outputs [1, 3]. These approaches motivate both selective single-model execution and the broader possibility of coordinating specialised roles.

Hydra's emphasis is the decision boundary around such execution choices: which candidates are admissible, which task requirements matter, and how client priorities influence selection. Multi-metric evaluation is necessary to characterise these choices, as quality alone does not describe resource behaviour [4]. Automated judging supplies a practical quality measure, but remains sensitive to evaluator effects [6]. Accordingly, the measurements below are identified as internal evidence with explicit denominators and comparison limits.

3 System Architecture

3.1 Components and information flow

Hydra represents model selection as a pipeline from request interpretation to controlled execution (Figure 1). A *task statement* describes the work to be performed. A *portfolio* contains permitted models and routes. A *profile* expresses priorities within that portfolio. A *route* specifies a single model or a plan with specialised roles. These concepts are separate controls: adding a candidate changes what can be selected, while adjusting a profile changes how eligible alternatives are preferred.

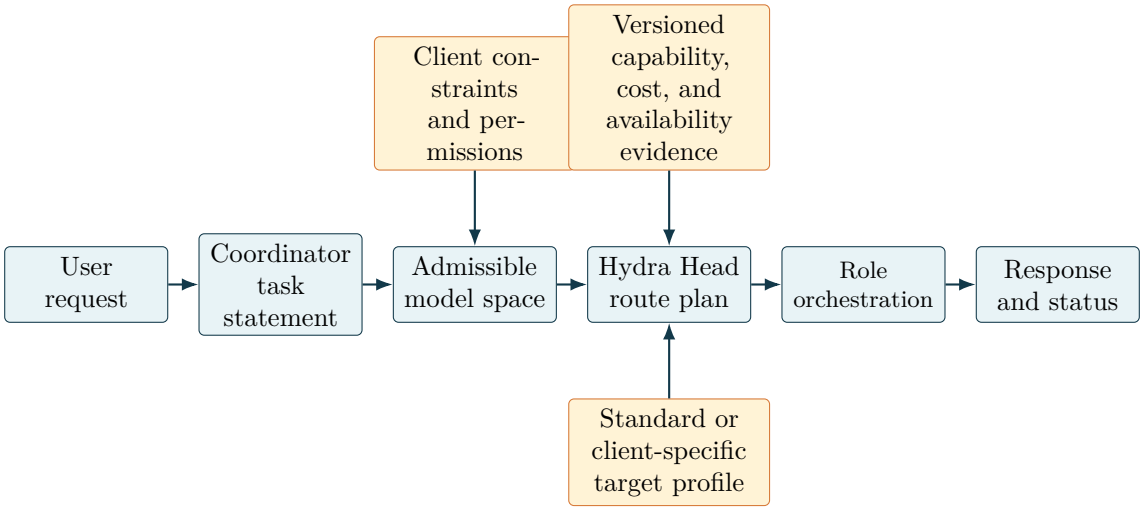

Figure 1: Hydra separates task interpretation, eligibility, route selection, and execution. Client constraints restrict the available routes; preferences guide selection.

The main interfaces are a structured task statement, an eligible candidate set, a selected execution plan, and the resulting response with execution status. Capability, cost, and availability evidence informs the routing decision; execution records describe what actually occurred. A routing estimate and an observed result therefore have different roles in the system.

3.2 Coordinator and task interpretation

The Coordinator translates the request into a structured task statement covering semantic demand, required capabilities, context, and output requirements. It describes the work without selecting a model. A modality requirement, for example, constrains eligibility before the Head compares suitable routes.

3.3 Eligibility before preference optimisation

Hard constraints define which candidates may participate. These include deployment restrictions, permitted model classes, and required capabilities. Selection preferences operate within this admissible set. Required capability, availability, or price evidence cannot be inferred from missing records; unknown cost is not treated as zero.

3.4 Head selection and adjustment model

The Head combines task features, versioned route evidence, and an active profile to choose an admissible plan. Hydra provides a fixed, quality-oriented standard configuration. Users can adjust selection priorities, including the emphasis on quality, or apply an OSS filter. The model catalogue can also evolve as eligible candidates and their evidence change.

Preferences and eligibility are separate controls: one changes how candidates are compared, the other changes which candidates may participate. The evaluated adjustments illustrate this flexibility, without implying that every change improves an answer. Here, adaptation means task- and preference-conditioned selection; the benchmark does not establish online learning or autonomous policy improvement.

3.5 Execution and observability

Orchestration enforces the selected plan and records outcomes and resource usage. Completion, valid scoring, and complete cost accounting are distinct states and are reported separately. The architecture accommodates specialised roles, but this benchmark measures single-route generations; it does not establish a multi-role performance advantage.

4 Evaluation Methodology

4.1 Dataset and comparison design

The current September 2026 comparison presents four Hydra adjustments and six fixed comparator settings, each with 100 planned tasks. A and D have no OSS filter; E and H have an OSS filter. These labels identify experimental settings, not a quality ranking. An absent filter does not exclude OSS models, and the OSS designation is the benchmark's operational eligibility classification.

This is a post-hoc selection from the internal adjustment study, not an exhaustive sweep or a preregistered ranking. All arms use the same repeated task set. A and the fixed controls were measured on September 9; D, E, and H on September 10 in separate execution batches. The controls are therefore historical rather than contemporaneously randomised. Comparator names refer to the fixed control arms.

The benchmark measures direct response generation without tool calling or external tool execution. The comparator set targets this task scope rather than an exhaustive comparison of flagship models. It does not test agentic, multi-step workflows or establish that higher-capability models are unnecessary for all tasks without tools. The scoring model serves only as an evaluator, not as a generation comparator.

4.2 Quality assessment and uncertainty

A blind Fable 5.1 evaluator scores each available answer against the task and reference on an absolute internal 0–100 scale. Model identity, cost, and time are excluded from the scoring packet.

For arm a, let S_a be the set of score-eligible tasks and $q_{a,t}$ their quality scores. The reported mean is

$$\bar{q}_a = \frac{1}{|S_a|} \sum_{t \in S_a} q_{a,t}. \tag{1}$$

Missing scores are excluded rather than replaced with zero; $|S_a|$ is reported alongside the mean. The 95% percentile intervals use 2,000 task-bootstrap replicates without multiplicity correction. They describe task-sampling variation within this dataset, not evaluator bias or the effects of repeatedly using the dataset for development.

4.3 Resource accounting

Generation cost is the confirmed total over 100 planned tasks, including recorded attempts and retries. Unresolved attempt costs make that value a lower bound, marked $\geq$. It is neither cost per scored completion nor the total cost of a deployed Hydra service: judging, Coordinator work, and other service overheads are outside this generation-cost measure. Coordinator and routing overhead is expected to be small in the intended deployment, but its contribution is not quantified here. Evaluation cost is separate experimental expenditure; it is not assumed negligible.

Time is successful-task wall time, including recorded retry or recovery elapsed time. Both mean and median are shown because long executions can affect the mean substantially. This excludes unsuccessful tasks from the timing summary and is not full end-to-end system latency. Token usage sums known completion tokens across generation attempts; the chart divides that sum by 100 planned tasks. It is not mean answer length. Unknown usage remains incomplete and is marked separately in the resource view.

5 Benchmark Results

5.1 Quality, cost, and coverage

Table 1 presents quality together with its generation-cost trade-off. GLM 5.3 Flash has the highest observed quality mean, 87.95, at \$0.11406 per 100 planned tasks. Hydra D records 85.43 at \$0.02062: 81.9% lower generation cost with a 2.52-point lower quality mean. Against DeepSeek V4 Pro 0813 (87.72; \$0.39692), D's generation cost is 94.8% lower with a 2.29-point lower mean. All three cost totals are complete within the recorded generation boundary.

The four Hydra means span 83.83–85.66. Costs for A, E, and H are lower bounds, so their apparent price differences cannot establish a complete-cost saving. Percentages above use unrounded totals and equal planned-task denominators; they are descriptive generation-cost comparisons, not full-service savings or evidence of equivalent quality.

Table 1: Selected internal N=100 comparison (A, D, E, H and six fixed controls). Cost is confirmed generation USD per 100 planned tasks; ≥ marks incomplete cost. OSS filter applies only to Hydra; – identifies fixed comparators.

Arm	OSS filter	Quality	95% interval	Scored	Gen. USD
GLM 5.3 Flash	–	87.95	82.78–92.70	99/100	0.11406
DeepSeek V4 Pro 0813	–	87.72	82.31–92.73	99/100	0.39692
Hydra adjustment E	Yes	85.66	80.01–90.94	96/100	≥ 0.23388
Hydra adjustment D	No	85.43	79.71–90.55	98/100	0.02062
GPT-5.6 Luna max	–	84.43	78.54–89.94	99/100	0.18695
Hydra adjustment A	No	84.06	77.87–90.00	99/100	≥ 0.03989
MiniMax M3	–	83.91	78.32–89.18	99/100	0.20855
Hydra adjustment H	Yes	83.83	77.50–89.89	95/100	≥ 0.04837
GPT-OSS 120B high	–	82.82	76.34–88.37	95/100	≥ 0.14708
GPT-5.6 Luna low	–	81.77	75.55–87.73	99/100	0.02974

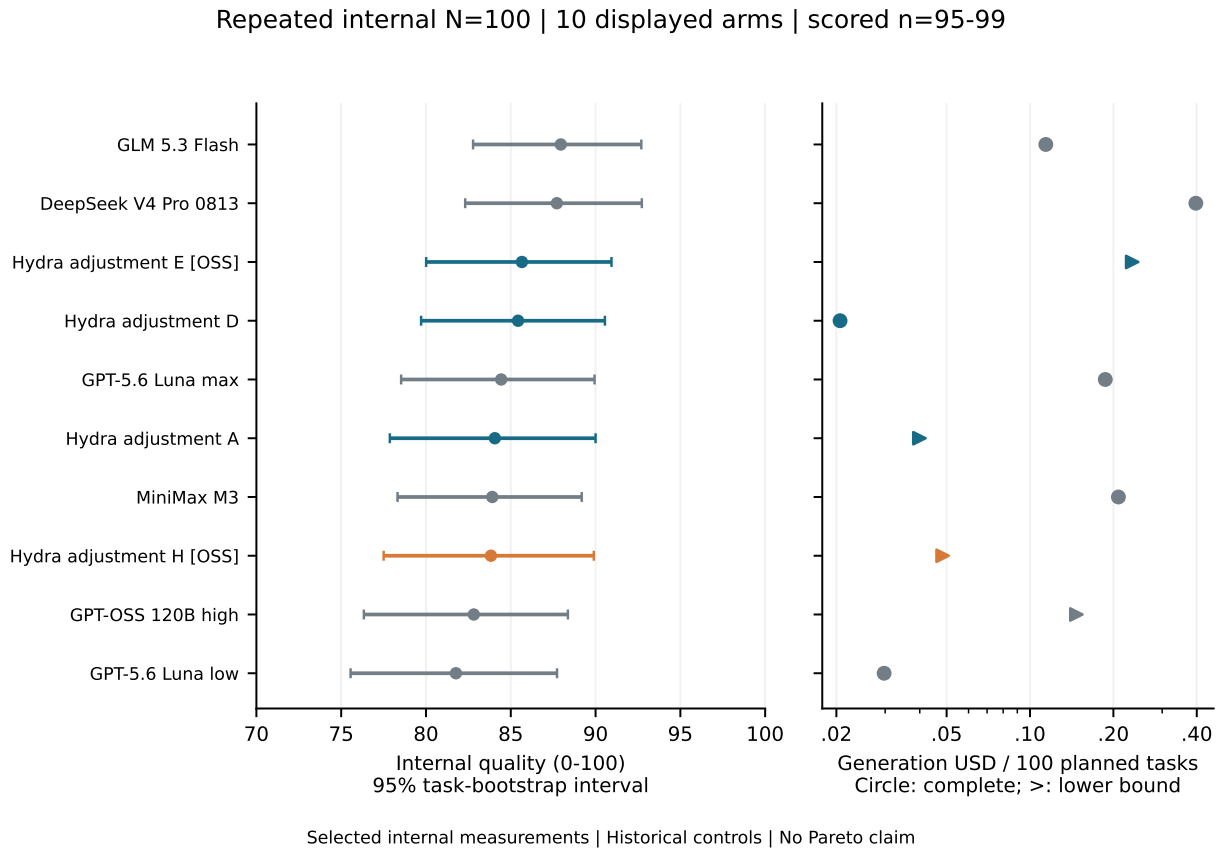

Figure 2: Selected internal quality and generation cost (A, D, E, H and six fixed controls); repeated N=100 dataset, scored n=95–99 per arm. Intervals are task-bootstrap intervals. Right-pointing cost markers indicate lower bounds. OSS denotes an active filter.

The paired quality difference between D and GLM is -2.76 points over 98 jointly scored tasks (95% interval: -6.59 to 0.82), alongside D's 81.9% lower generation cost per 100 planned tasks. The paired and arm-mean differences use different task subsets. This exploratory comparison establishes

neither superiority nor non-inferiority; no cost saving is asserted for every Hydra adjustment.

5.2 Latency and output usage

Table 2 shows the latency–usage trade-off. D's median successful-task time is 3.02 seconds versus GLM's 4.79 seconds, with 80% fewer completion tokens (44,295 versus 221,475 over 100 planned tasks). Both token records are complete. D is not the fastest setting: MiniMax has a 1.69-second median.

Mean times reveal long executions that medians conceal: D averages 7.93 seconds, while E averages 34.68 seconds despite a 4.28-second median. Typical response time alone therefore understates resource variability. These are descriptive comparisons of successful-task generation time; token totals include recorded attempts and do not measure answer length or end-to-end service latency.

Table 2: Resource denominators and incompleteness (September 2026 snapshot). Times are seconds per successful task. Tokens are known completion-token totals across 100 planned tasks. Unknown counts refer to attempts with unresolved generation cost. Displayed Hydra settings: E and H with OSS filter; A and D without OSS filter.

Arm	Completed	Median s	Mean s	Tokens total	Unknown
GLM 5.3 Flash	100/100	4.79	28.92	221,475	0
DeepSeek V4 Pro 0813	100/100	7.73	32.40	218,088	0
Hydra adjustment E	98/100	4.28	34.68	204,250	29
Hydra adjustment D	100/100	3.02	7.93	44,295	0
GPT-5.6 Luna max	100/100	3.98	16.16	151,889	0
Hydra adjustment A	100/100	4.48	20.19	24,617	2
MiniMax M3	100/100	1.69	8.70	208,225	0
Hydra adjustment H	96/100	7.36	39.18	72,785	56
GPT-OSS 120B high	96/100	35.82	370.82	888,680	51
GPT-5.6 Luna low	100/100	2.49	3.94	20,878	0

5.3 OSS-filtered settings

E and H illustrate two measured settings under the OSS filter. E records 85.66 quality points on 96 scored tasks; H records 83.83 on 95. Their confirmed generation costs are at least $0.23388 and $0.04837, respectively, with 29 and 56 unresolved attempt costs. These lower bounds do not establish a cost ranking. Different coverage and execution batches also prevent attributing the observed quality difference to adjustment alone.

6 Discussion

6.1 What adjustment provides

A quality-oriented standard profile supplies a consistent starting point; user-adjustable priorities and an optional OSS filter allow other operating points. The model catalogue is another configurable boundary. The four settings illustrate this design space rather than a sequence of improvements. D's 81.9% lower recorded generation cost than GLM, with a 2.52-point lower quality mean, is one observed trade-off; it is not a guarantee for every profile.

6.2 System-level interpretation

Generation costs exclude Coordinator, orchestration, and evaluation overhead; timing covers successful tasks. The results characterise the measured inference step, not complete service pricing or end-to-end latency.

6.3 Continuous development

Hydra is continuously developed to improve routing quality, efficiency, and support for diverse workloads. Its configurable profiles and model catalogue allow the system to evolve while preserving the separation of task requirements, eligibility, selection, and execution. The benchmark in this paper describes one defined direct-response workload, not the full scope of system use.

7 Limitations

The task set has been reused, scored coverage differs, and controls were not rerun contemporaneously. Provider, timeout/recovery, and judge-batch differences prevent clean causal attribution to adjustment. Bootstrap intervals describe variation conditional on observed scores; they do not account for missing outcomes, systematic judge error, or repeated dataset use. Post-hoc selection also limits interpretation of the displayed settings.

The paper reports aggregate internal measurements for the stated task set and execution conditions. These results do not establish performance on an independent holdout or across all workloads. No Pareto or external-ranking claim is made.

8 Conclusion

Hydra separates task interpretation, model eligibility, route selection, and execution in a configurable routing system. A quality-oriented standard configuration provides a consistent starting point, while adjustable priorities and portfolio constraints, including an OSS filter, support different task and user requirements. The contribution is this controlled flexibility, assessed through observed quality and resource use rather than one fixed model choice.

The repeated internal N=100 benchmark illustrates four Hydra operating points alongside six fixed controls. D records 85.43 quality points at 81.9% lower generation cost than GLM's 87.95-point result, and 94.8% lower cost than DeepSeek's 87.72-point result. Against GLM, D also uses 80% fewer completion tokens and has a lower median successful-task time (3.02 versus 4.79 seconds). These observations show why quality, cost, and latency should be considered together: a modest difference in observed quality can accompany a substantial difference in generation resources.

The evidence does not establish equivalent quality, universal cost savings, or a single optimal configuration. Costs remain incomplete for several settings; repeated dataset use, unequal scored coverage, and historical controls constrain generalisation. Reported savings concern generation, not the complete service.

Hydra provides a consistent routing framework with a quality-oriented default and controls for adapting selection to user requirements. Continuous development builds on this system to improve quality, efficiency, and workload coverage. The guiding principle remains stable: select an admissible route for the task and assess its realised quality and resource use together.

Acknowledgements

This work was carried out by Nils Rump and Noel Lorenz at Mindverse.

References

[1] Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. AutoMix: Automatically mixing language models. *arXiv preprint arXiv:2310.12963*, 2024. URL `https://arxiv.org/abs/2310.12963`.

[2] Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023. URL `https://arxiv.org/abs/2305.05176`.

[3] Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. LLM-Blender: Ensembling large language models with pairwise ranking and generative fusion. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023. URL `https://arxiv.org/abs/2306.02561`.

[4] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *Transactions on Machine Learning Research (TMLR)*, 2023. URL `https://arxiv.org/abs/2211.09110`.

[5] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs with preference data. *arXiv preprint arXiv:2406.18665*, 2024. URL `https://arxiv.org/abs/2406.18665`.

[6] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023. URL `https://arxiv.org/abs/2306.05685`.