Static Before Smart: Exact Cost Accounting and Baselines for Pay-per-Token LLM Routing

Arjun Subramanian
MIT Computer Science and Artificial Intelligence Laboratory (CSAIL)
Cambridge, Massachusetts, USA
arjun_s@mit.edu

Nithilan Karthik
Managent
Dallas, Texas, USA
nithilan@managent.tech

Abstract

Request-level LLM routing is a middleware control decision over metered model executions: a policy determines which backend serves each request and therefore which token charges are incurred. Yet routing is often evaluated using an aggregate shortcut—model-selection fraction multiplied by each model's average cost—and against baselines matched on routing rate rather than realized spend. These comparisons can misstate the economics of query-dependent routing because assignment can correlate with request and response length. We present an audit methodology for pay-per-token LLM routing built on exact request-level accounting, static baselines matched by realized dollars, held-out evaluation, and explicitly retrospective oracle headroom. On a frozen 243-query held-out test, query-only lexical routers fail to improve over the cost-matched static policy, while a hindsight oracle retains a 5.35-percentage-point quality advantage at nearly the same model cost (95% grouped-bootstrap interval: +2.17 to +8.81 points; permutation $p = 0.00430$), indicating complementarity that the evaluated features do not capture. An aggregate reconstruction of 1,307 released RouteLLM GSM8K examples shows that replacing routing-rate matching with realized-dollar matching reduces the mean descriptive routing edge from 0.756 to 0.566 percentage points; none of the nine interior operating points is individually significant under the stored Monte Carlo analysis. Finally, using hindsight assignments solely as an accounting stress test, 56 public xRouteBench slices exhibit a 4.81% median absolute difference between aggregate-mixture and exact assignment costs (6.55% mean), with the largest observed value, 52.80%, occurring on a 107-query AIME slice. These results support a simple evaluation contract for serverless and agentic LLM systems: before attributing economic benefit to a routing policy, account for every selected model execution at request granularity, compare against the strongest static policy at the same realized budget, and report retrospective oracle headroom separately from deployable routing gain.

CCS Concepts

• **Computing methodologies** → *Natural language processing*; • **Applied computing** → Economics.

Keywords

LLM routing, serverless inference, cost accounting, evaluation, benchmarking, inference economics

WoAIS '26, Tarragona, Spain
2026.

ACM Reference Format:
Arjun Subramanian and Nithilan Karthik. 2026. Static Before Smart: Exact Cost Accounting and Baselines for Pay-per-Token LLM Routing. In *Proceedings of the Second International Workshop on AI and Serverless (WoAIS '26), December 14, 2026, Tarragona, Spain.* ACM, New York, NY, USA, 6 pages.

1 Introduction

Large language models are increasingly consumed as metered services. In serverless and agentic systems, model selection is a middleware control-plane decision: it determines which backend executions occur, and a single decision can trigger retries, branches, or downstream fan-out. Infrastructure mechanisms can optimize how those invocations start, scale, and schedule [6, 10], while routing and cascade methods choose which models are invoked [2, 9]. These layers meet at a prerequisite question for any end-to-end efficiency claim: what did the control policy actually cause the system to spend at the quality it delivered? Exact attempt-level accounting is therefore not a substitute for serverless deployment measurement; it is the economic substrate on which such measurements must be interpreted. That question is easy to answer incorrectly. Consider a two-model router sending a fraction p of queries to a strong model. A common evaluation shortcut computes cost as $p\bar{c}_s + (1-p)\bar{c}_w$, where $\bar{c}_m$ is a model's average cost over the benchmark. This is exact for a query-independent random mixture, but not generally for a router. Routing decisions are functions of the query, while per-request LLM cost is also query-dependent through input and output tokens. If the router escalates queries that are systematically longer (or shorter), selection and cost are correlated. The routing fraction is therefore not sufficient to recover realized cost. A second problem is the comparator. Matching the fraction of strong-model calls does not necessarily match dollars, because the routed subset can have a different token distribution. A router should instead be compared against the best query-independent policy at the same realized budget. Otherwise an apparent "routing gain" can partly be a budget mismatch. Recent benchmark work likewise finds that sophisticated routers can fail to reliably beat simple baselines under unified evaluation [8]; xRouteBench, introduced with the LLM-Router infrastructure, further broadens routing evaluation across multiple modalities and domains [5]. This paper studies these issues through an audit of a three-model cost-aware router and two external panels. The goal is deliberately not to rescue the router. Negative and null results are retained because they are informative for evaluating middleware control policies under metered inference. We make four contributions:

1. We formulate an exact request-level economic accounting protocol for pay-per-token LLM routing.

2. We define a dollar-matched query-independent static baseline that asks whether query dependence adds value at the same realized budget.

3. We use a frozen held-out evaluation to separate realized learned-routing value from retrospective oracle headroom.

4. We stress-test the protocol on released RouteLLM material and 56 xRouteBench slices, exposing cost misestimation and a regret-sign reversal. The short covariance derivation in §2.1 is explanatory accounting algebra rather than a standalone theoretical contribution: it makes explicit why routing fractions cease to be sufficient once assignments depend on queries. Our scope is inference economics and evaluation. We do not report platform cold-start, autoscaling, or GPU-energy measurements; those require real deployment instrumentation and are intentionally separated from the claims here. Accordingly, the measured token charge is a controlled component of end-to-end cost, not a claim about total serverless TCO.

2 Exact Accounting for Query-Dependent Routing

2.1 Request-level formulation

Let $q_1, \ldots, q_n$ be queries and $m \in \{1, \ldots, K\}$ candidate models. Let $A_{im} \in \{0, 1\}$ indicate that the router chooses model m for query i, with $\sum_m A_{im} = 1$. Let C_{im} be the reconstructed token-charge cost had model m served query i. The exact mean model cost of policy A is

$$\bar{C}_{\text{exact}} = \frac{1}{n} \sum_{i=1}^{n} \sum_{m=1}^{K} A_{im} C_{im}. \tag{1}$$

Suppose instead we know only the model selection fractions $p_m = \frac{1}{n} \sum_i A_{im}$ and the unconditional per-model average costs $\bar{C}_m = \frac{1}{n} \sum_i C_{im}$. The aggregate mixture estimate is

$$\bar{C}_{\text{mix}} = \sum_{m=1}^{K} p_m \bar{C}_m. \tag{2}$$

Writing expectation over a uniformly sampled benchmark query gives the exact identity

$$\bar{C}_{\text{exact}} - \bar{C}_{\text{mix}} = \sum_{m=1}^{K} \text{Cov}(A_m, C_m). \tag{3}$$

Thus the shortcut is exact only when the covariance terms cancel. A query-independent mixture has zero covariance in expectation. A learned or heuristic router need not. The same logic applies to latency or any request-level quantity whose value depends jointly on the selected model and query. For dollars, the effect can arise even with fixed public token prices because response lengths vary substantially by query.

2.2 Cost-matched static baseline

Let model m have mean quality Q_m and mean realized cost $\bar{C}_m$ on an evaluation population. A query-independent static mixture chooses model m with probability π_m, giving expected quality and cost

Table 1: Evaluation panels. RouteLLM is an aggregate reproduction; xRouteBench uses hindsight assignments for accounting stress tests only.

Panel	Pool/test	Models	Purpose
Frozen diagnostic	364/364	3	Heuristic diagnostic
Held-out split	1200/243	3	Router selection
RouteLLM GSM8K	1307/1307	2	Reconstruction
xRouteBench	56 slices	18	Stress test

Table 2: Frozen model/API and pricing configuration. Rates in USD per million tokens. September 10, 2026 is the audit snapshot, not a separately archived pricing timestamp.

Tier	Provider	Model/API	Date	Input	Output
T1	Together	Qwen3.5-9B	2026-09-10	.17	.25
T2	Together	gpt-oss-20b	2026-09-10	.05	.20
T3	OpenAI	GPT-4o	2026-09-10	2.50	10.00

$$Q(\pi) = \sum_m \pi_m Q_m, \quad C(\pi) = \sum_m \pi_m \bar{C}_m. \tag{4}$$

For a router with realized budget B, we compare it to the upper envelope of static mixtures satisfying $C(\pi) \le B$. With a small model pool this is the convex hull of fixed-model operating points. This baseline receives no query information; it answers whether query dependence adds value beyond simply spending the same amount differently. When the hull is constructed on the evaluation population, we treat it as an ex post diagnostic frontier, not a deployable policy learned out of sample. We also report a hindsight oracle that selects a correct low-cost model per query whenever possible. It uses information unavailable at inference time and is therefore only an upper bound. The gap between the static baseline and oracle measures available routing headroom; the gap between a deployable router and oracle measures uncaptured headroom.

3 Audit Protocol

3.1 Panels and models

Table 1 summarizes the evidence. The frozen diagnostic panel contains 364 prompts: 164 HumanEval [3], 100 MMLU [7], and 100 GSM8K [4]. Each prompt was run at temperature zero on three configured models. Provider usage fields supply prompt and completion token counts; request-level USD is reconstructed from those counts using the frozen rate table in Table 2. The complete matrix contains the response, correctness, token usage, reconstructed cost, and observed HTTP round-trip time for each model–query pair. The legacy audited router is a query-only keyword/complexity decision list. Its historical evaluation is not held out; we use it as a motivating failure case rather than generalization evidence. The accounting identity and evaluation protocol are priceagnostic; the particular dominance of a static model is specific to this frozen model/pricing snapshot and is not claimed to persist under arbitrary repricing or model changes. The benchmark charges no priced router step. For held-out evaluation we use a separate 1,200-query pool drawn evenly from GSM8K, MBPP [1], and MMLU. 2 Static Before Smart: Exact Cost Accounting and Baselines for Pay-per-Token LLM Routing WoAIS '26, December 14, 2026, Tarragona, Spain

We freeze a deterministic 60/20/20 train/validation/test split (720/237/243), group MMLU by subject, and merge exact or token-Jaccard-≥ 0.9 duplicate clusters before splitting. TF–IDF preprocessing is fit on training data only. Candidate query-only routers are logistic regression, a shallow boosted tree, a length threshold, and the frozen legacy heuristic. Hyperparameters and method selection use validation only; the final test is protected by frozen query IDs and a configuration hash. The selected method is logistic regression. Group-level bootstrap and permutation tests use 10,000 replicates over 170 test groups.

3.2 External checks

RouteLLM. We use the released GSM8K weak/strong responses from RouteLLM [9], after the project's contamination filtering, yielding 1,307 examples. The weak model is Mixtral-8x7B-Instruct-v0.1 and the strong model is GPT-4- 1106-preview. Historical BERT-router threshold aggregates provide 11 operating points. We reconstruct token-based dollars from response text and the frozen list-price assumptions used by the audit. Because query-level router scores are not retained in the frozen artifact, this is an aggregate reproduction: it supports the cost-matching/accounting observation, but we do not present it as a fresh query-level replication. xRouteBench. xRouteBench provides per-query model quality, token counts, latency, and costs across heterogeneous routing datasets [5]. We analyze 56 train/test/task slices with 18 candidate models each. Because our artifact has no learned-router assignments, we use a per-query qualitymaximizing hindsight assignment only to stress-test Eq. 3. This measures assignment-dependent accounting effects; it does not validate a deployable external router.

3.3 Reproducibility controls

The replication artifact freezes processed numeric inputs, the model/pricing configuration, policy outcomes, and scripts needed to regenerate the headline tables and statistical tests offline. It makes no paid API calls. Modeled energy, local dry-run serverless traces, and discrete-event simulations are excluded from empirical claims because they are not real cloud-platform measurements.

4 Results

4.1 Diagnostic: a static model dominates the legacy heuristic

Figure 1 shows the 364-query diagnostic panel. The legacy heuristic reaches 86.81% accuracy at $0.000788 per query. Always using gpt-oss-20b (T2) reaches 92.31% at $0.000115 per query: 5.49 percentage points higher accuracy while costing 6.87× less. The paired correctness difference favors T2 (exact McNemar $p = 3.25 \times 10^{-4}$), although the panel was not held out and therefore should not be read as evidence against routing in general. Qwen3.5-9B (T1), the uniform random policy, and GPT-4o (T3) are also more expensive than T2 under this snapshot. A hindsight oracle reaches 96.43% at $0.000129 per query, only modestly above T2's cost. This case illustrates why router evaluation must begin with fixed-model baselines. It is intentionally a snapshot-specific sanity check, not a general result about routing: one candidate happens to be simultaneously cheaper and more accurate on the observed workload. A simple

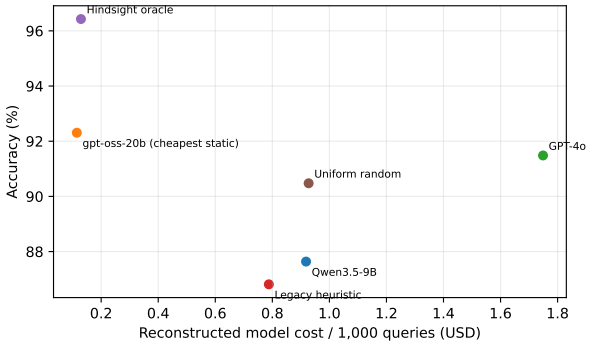


Figure 1: Reconstructed token-charge USD versus accuracy on the frozen 364-query panel. The legacy heuristic is dominated by the gpt-oss-20b static policy under the frozen price snapshot. The oracle is hindsight-only.

Table 3: Held-out 243-query test. USD is realized model cost per query; the hindsight oracle shows remaining complementarity.

Policy	Quality (%)	USD/query	Δ vs static (pp)
Cost-matched static (T2)	88.89	.000153	0.00
Selected logistic	88.89	.000153	0.00
Length threshold	88.89	.000201	0.00
Frozen heuristic	86.01	.000878	-2.88
Hindsight oracle	94.24	.000151	+5.35

repricing sensitivity reinforces that distinction. If T2's input and output prices are scaled together while frozen outputs and all other model rates are held fixed, Always-T2 remains both more accurate and cheaper than the legacy heuristic for multipliers below 8.49× (their costs are equal at 8.49×). T2 and T1 mean costs cross at 8.01×, so the static cost–quality frontier itself changes well before arbitrary repricing. The accounting protocol is invariant to that repricing even though the identity of the best static policy is not.

4.2 Held-out lexical routers do not capture oracle headroom

The held-out experiment asks whether a cleaner query-only learner can recover the oracle headroom. It does not. Validation selects logistic regression, but on the 243-query frozen test it sends every query to T2. Consequently, the selected router and the pre-frozen cost-matched static policy are identical: both obtain 88.89% quality at $0.000153/query. The shallow tree also collapses to the same point. A length threshold routes 4.53% to the strong model: quality remains 88.89%, while cost rises to $0.000201/query. The original heuristic obtains 86.01% at $0.000878/query. The oracle attains 94.24% at $0.000151/query, a 5.35-point quality gap over the static policy at essentially the same cost. Grouped inference gives a 95% bootstrap interval of +2.17 to +8.81 percentage points and permutation p = 0.00430 (10,000 draws each over 170 groups). The 13 test items corrected by the oracle but missed by static T2 are concentrated in MMLU (9 MMLU, 2 GSM8K, 2 MBPP). Their prompt lengths are

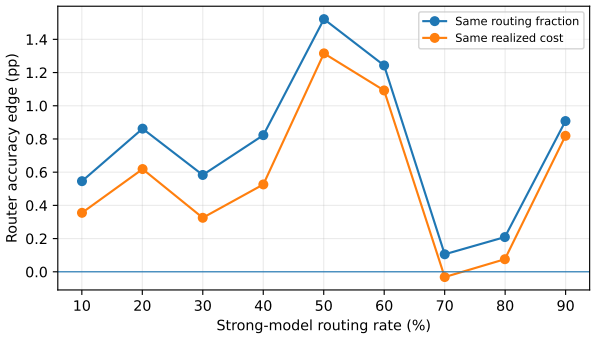


Figure 2: RouteLLM GSM8K aggregate reproduction. Matching strong-model fraction is not the same as matching realized dollars; dollar matching reduces the mean interior-point edge from 0.756 to 0.566 percentage points.

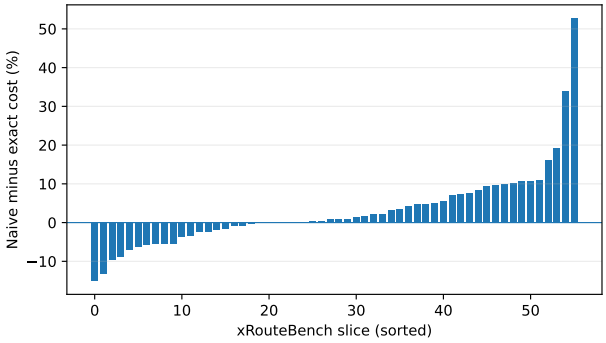


Figure 3: Naive aggregate-mixture cost minus exact assignment cost across 56 xRouteBench slices, sorted by error. Hindsight assignments are used only as an accounting stress test.

not obviously separated from the remaining test items (median 168 versus 163 tokens); their T2 completion lengths are longer (median 544 versus 294 tokens), but completion length is unavailable before routing and this comparison is descriptive. The null learned-router result therefore does not imply that model complementarity is absent. It says only that the evaluated query-only lexical features do not capture that complementarity under this frozen protocol. For system designers, oracle headroom should not be confused with deployable savings.

4.3 Cost matching changes the apparent RouteLLM edge

On the released RouteLLM GSM8K panel, the nine interior threshold points have a mean accuracy edge of +0.756 percentage points over a random baseline with the same strongmodel fraction. Once the baseline is instead matched to the router's realized dollar cost, the mean edge falls to +0.566 percentage points, a reduction of 0.189 percentage points. Figure 2 shows that the difference is systematic over most thresholds. Eight of nine interior points remain positive after cost matching, but none individually reaches $p < 0.05$ in the 20,000-draw Monte Carlo baseline used by the audit. We therefore treat this as descriptive external evidence, not a significance claim. The accounting shortcut also differs from reconstructed exact cost by up to 6.31% on this sweep. At the approximately 30% strong-model point, naive and exact accounting place the policy on opposite sides of the chosen cost-regret reference: naive regret is $+3.21 \times 10^{-5}$ USD/query whereas exact regret is -9.23×10^{-5}. Eq. 3 reconciles every threshold to floatingpoint precision. This is the operationally important case: even a modest percentage cost error can reverse a nearboundary decision about whether routing pays.

4.4 The covariance effect is not confined to one router

xRouteBench lets us ask whether assignment-dependent accounting differences occur across a broader public corpus. Because our xRouteBench artifact lacks learned-router assignments, we use hindsight assignments only to induce a deterministic query-dependent selection pattern, then compare aggregate-mixture and exact costs. Across 56 slices, the signed relative error has mean +2.98% and median +0.84%; the median absolute error is 4.81% and the mean absolute error is 6.55%. Errors are positive on 33 slices and negative on 23, consistent with Eq. 3: the direction depends on which queries the assignment sends to each model. The observed range is -15.13% to +52.80%, with the largest positive value on a small AIME training slice (107 queries). Even restricting attention to larger slices does not eliminate the effect: the 4,489-query generic training slice has +19.18% relative error. These results do not imply that every learned router will incur a large bias. They show the more basic point that routing fractions are not sufficient statistics for cost once assignment depends on the query. Reporting query-level realized cost removes the ambiguity at negligible analysis complexity. 4 Static Before Smart: Exact Cost Accounting and Baselines for Pay-per-Token LLM Routing WoAIS '26, December 14, 2026, Tarragona, Spain

5 Implications for Serverless and Agentic AI Control Planes

For a serverless audience, the role of this audit is deliberately narrow but foundational. A router is a middleware control decision because it chooses the metered backend invocations that infrastructure must execute; in agentic workflows, that choice can also be replicated through retries and fan-out. End-to-end serverless efficiency still depends on cold starts, queueing, autoscaling, router compute, and platform billing, none of which we measure here. Exact attempt-level model accounting therefore provides a necessary economic layer, not a sufficient deployment evaluation. The audit suggests a minimal evaluation contract for costaware routing in pay-per-use AI systems. 1. Account at request granularity. Log the selected model, billable input/output tokens, retry attempts, and realized charge for each request. Aggregate only afterward. If provider usage fields are unavailable, reconstruct tokens with the correct tokenizer and state the pricing snapshot. Router compute, if nonnegligible, should be added separately rather than hidden in model

cost. 2. Match dollars, not routing fractions. A "30% strong model" random baseline can consume a different budget from a query-dependent router. Compare quality against a query-independent mixture at the same realized cost, and compare cost at matched quality when feasible. The two views make dominance explicit. 3. Separate headroom from prediction. A hindsight oracle answers whether complementary model errors exist; it does not show that a router can predict them. Report both oracle gap and deployable-router gain. Our held-out result exhibits a large oracle gap while the learned router adds zero value over static routing. 4. Freeze evaluation. Select router features and thresholds on train/validation only, using duplicate-aware splits where appropriate. Freeze final test IDs, pricing, and result hashes before inspection. Retain negative test outcomes rather than retuning post hoc. 5. Keep economic and platform measurements distinct. Token cost, API latency, function cold-start delay, GPU energy, and queueing are different quantities. A local stub or discrete-event simulator should not be labeled as a measured serverless deployment. For WoAIS-style systems work, real platform experiments should record region, instance configuration, lifecycle evidence for cold starts, concurrency, retries, and full billed cost. This study deliberately stops short of those claims. These rules are especially relevant for agentic/serverless workloads, where a routing decision can fan out into repeated calls and highly variable generations. Request-level accounting composes naturally with such workflows: sum the actual cost of every invoked step. Aggregate model averages do not. A future deployment study should add measured router latency/compute, retry amplification, cold-start and queueing effects, and full platform billing to the same perattempt ledger rather than treating token charges as total TCO.

6 Related Work

Cost-aware LLM selection. FrugalGPT studies prompt adaptation, approximation, and cascades for reducing API cost [2]. RouteLLM learns routers from preference data and reports substantial cost reductions at comparable quality in some settings [9]. Our work is complementary: rather than proposing a stronger router, we focus on whether a reported operating point is economically well measured and whether it beats a cost-matched query-independent alternative. Router benchmarking. Unified benchmarking finds that several routers do not reliably beat simple baselines [8]. xRouteBench, introduced with the LLMRouter infrastructure, extends evaluation across generic, memory, vision, time-series, and personalized settings [5]. We use its per-query tables to stress-test assignment-dependent cost accounting without claiming learned-router performance when assignments are absent. Serverless LLM systems. ServerlessLLM targets low-latency model loading, migration, and scheduling for elastic LLM inference [6]. FaaScale accelerates on-demand model scaling and studies bursty inference traces [10]. Those systems optimize the infrastructure that executes requests; our study targets the economic control decision that chooses which model a request invokes. A complete production evaluation needs both layers.

7 Code and Reproducibility

Code and replication materials supporting the paper's quantitative results are available upon request. The replication package contains the frozen configurations, processed numeric inputs, evaluation scripts, and derived outputs used in the paper and supports offline reproduction without paid API calls. The RouteLLM component is an aggregate reconstruction, and the xRouteBench component remains a hindsight diagnostic, as described above.

8 Limitations and Threats to Validity

The frozen 364-query legacy-router result is diagnostic and in-sample; it cannot establish population-level inferiority of the heuristic. The held-out pool is cleaner but derives from data used in earlier development of a predecessor router, so method-level contamination cannot be ruled out even though the final split is ID-disjoint and test-blind. Our static convex-hull calculations on external panels are ex post evaluation baselines rather than policies trained on a separate deployment trace. The RouteLLM result is an aggregate reproduction: the frozen artifact does not retain query-level BERT router scores, so we cannot rerun its full paired threshold analysis from scratch. We therefore do not elevate its Monte Carlo observations into a primary statistical claim. xRouteBench assignments in this paper are hindsight and are used only to demonstrate the magnitude and sign variability of the accounting covariance term. Finally, this paper studies model/API economics rather than cloud-platform overhead. We exclude modeled energy and do not claim measured cold-start, autoscaling, or queueing behavior. Real serverless deployment can introduce additional router overhead, retries, and billing components; all of those make exact attempt-level accounting more, not less, important.

9 Conclusion

A cost-aware LLM router should earn its complexity. That requires two checks before any architectural story: compute what the selected requests actually cost, and compare the resulting quality to the strongest static policy at the same budget. Exact accounting differs from selection-fraction accounting by an assignment–cost covariance term, which is non-zero whenever routing decisions correlate with query-dependent token cost. In our audit, this correction reaches decision-relevant magnitude, while a legacy heuristic is dominated and a held-out learned router collapses to its static baseline despite clear oracle headroom. For serverless AI, the practical lesson is simple: static before smart, and exact before aggregate.

References

[1] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program Synthesis with Large Language Models. arXiv preprint arXiv:2108.07732 (2021).

[2] Lingjiao Chen, Matei Zaharia, and James Zou. 2024. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. Transactions on Machine Learning Research (2024). https://openreview.net/forum?id=cSimKw5p6R

[3] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating Large Language Models Trained on Code. arXiv preprint arXiv:2107.03374 (2021).

[4] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training Verifiers to Solve Math Word Problems. arXiv preprint arXiv:2110.14168 (2021).

[5] Tao Feng, Fangxu Yu, Haozhen Zhang, Zhongjie Dai, Liangqi Yuan, Zijie Lei, Weizhi Zhang, Kunlun Zhu, Haodong Yue, Keyang Xuan, Ge Liu, and Jiaxuan

You. 2026. LLMRouter: Unified Infrastructure for Developing, Evaluating, and Deploying LLM Routers. arXiv preprint arXiv:2608.06867 (2026).

[6] Yao Fu, Leyang Xue, Yeqi Huang, Andrei-Octavian Brabete, Dmitrii Ustiugov, Yuvraj Patel, and Luo Mai. 2024. ServerlessLLM: Low- Latency Serverless Inference for Large Language Models. In 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24). USENIX Association, 135–153.

[7] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. In International Conference on Learning Representations.

[8] Hao Li, Yiqun Zhang, Zhaoyan Guo, Chenxu Wang, Shengji Tang, Qiaosheng Zhang, Yang Chen, Biqing Qi, Peng Ye, Lei Bai, Zhen Wang, and Shuyue Hu. 2026. LLMRouterBench: A Massive Benchmark and Unified Framework for LLM Routing. In Findings of the Association for Computational Linguistics: ACL 2026. 37733–37754. doi:10.18653/v1/2026.findings-acl.1881

[9] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. 2025. RouteLLM: Learning to Route LLMs with Preference Data. In International Conference on Learning Representations. https://openreview.net/forum?id=8sSqNntaMr

[10] Minchen Yu, Rui Yang, Chaobo Jia, Zhaoyuan Su, Sheng Yao, Tingfeng Lan, Yuchen Yang, Zirui Wang, Yue Cheng, Wei Wang, Ao Wang, and Ruichuan Chen. 2026. FaaScale: Unlocking Fast LLM Scaling for Serverless Inference. In Proceedings of Machine Learning and Systems, Vol. 8.