CHAPTER 8 / 3 MINUTE BRIEF

Measuring AI Workloads

Read the full chapter → / Read this brief online →

The main idea

Ten million tokens sent through a highly rated model do not establish a known amount or quality of completed work. A smaller model may be sufficient for the task; a larger one may succeed where it cannot. Only the application's acceptance results can settle the comparison.

This chapter also proposes **benchmark-weighted consumption**: an unvalidated, descriptive index that joins token counts to model benchmark scores. It can show a change in model mix that a raw token total hides. It cannot turn tokens into a measured quantity of intelligence or substitute for counting accepted work.

Explore this section in the full chapter →

What the workload actually consumes

For an AI workload, the unit of consumption is not the user request. It is the **model call**. One task — a support question answered, a report drafted — typically triggers several calls: a cheap model to interpret the question, a capable model to produce the answer, another model to check the result, occasionally a retry when something failed. Each call consumes input tokens and output tokens on a particular model.

Explore this section in the full chapter →

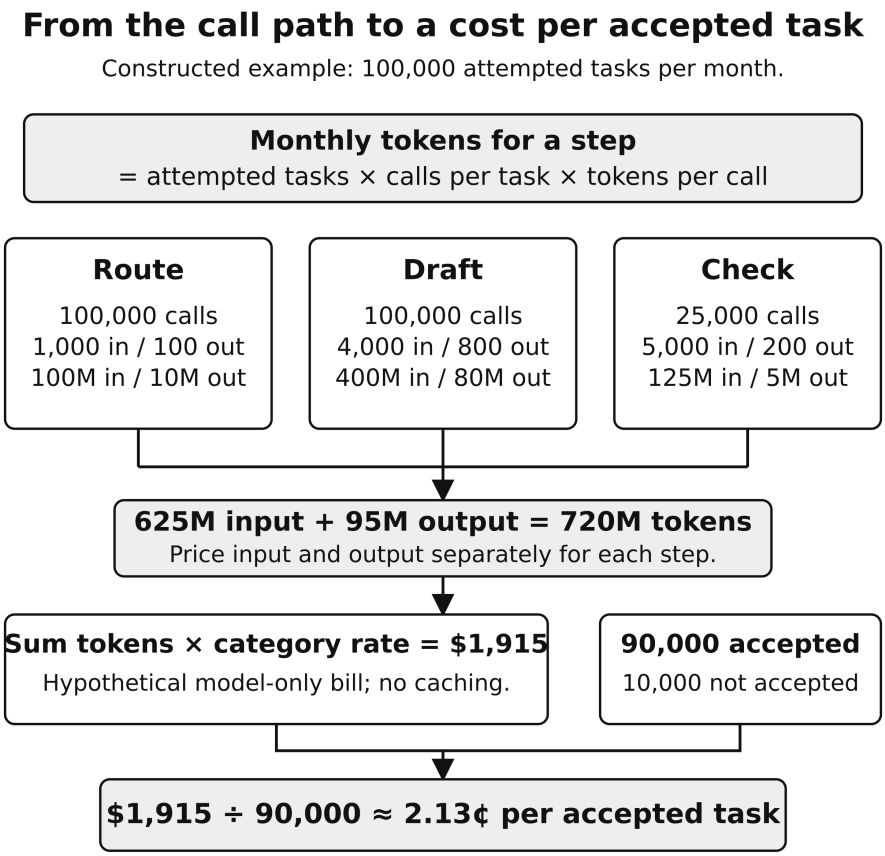


Diagram 8.1. The constructed call path expands attempted tasks into route, draft and check calls, then into token totals and the model-only bill. Dividing by accepted tasks retains the cost of rejected attempts.

Explore this section in the full chapter →

Describing the model mix

Start with a single benchmark version run under comparable conditions. For illustration, suppose two models score 0.85 and 0.45, where the score is a fraction of test cases passed. These are hypothetical results on one test, not universal capability ratings.

For each model, multiply its token count by that score, then add the products. If model k consumes T_k tokens and scores b_k, the index is:

$$B = \sum_{k} T_k \times b_k.$$

One million tokens at 0.85 contribute 850,000 benchmark-weighted tokens; one million at 0.45 contribute 450,000. This is an accounting convention proposed here, not an established measure of effective capability purchased.

An increase from ten million to twelve million weighted tokens is $(12 - 10) / 10 = \mathbf{20}$ **percent** growth in the index, not evidence of 20 percent more useful work. Repeating a prompt unnecessarily would raise the index too. A change in tokenization, benchmark version or task mix can also break the comparison. Publish those conditions alongside it.

Suppose routing moves work to a model with a lower benchmark score. The index falls for the same token count, making that model-mix change visible. It does not establish lost capability on this workload: the cheaper model may pass every task that matters here. Compare acceptance rates, correction effort and total cost before calling the routing change a saving.

Explore this section in the full chapter →

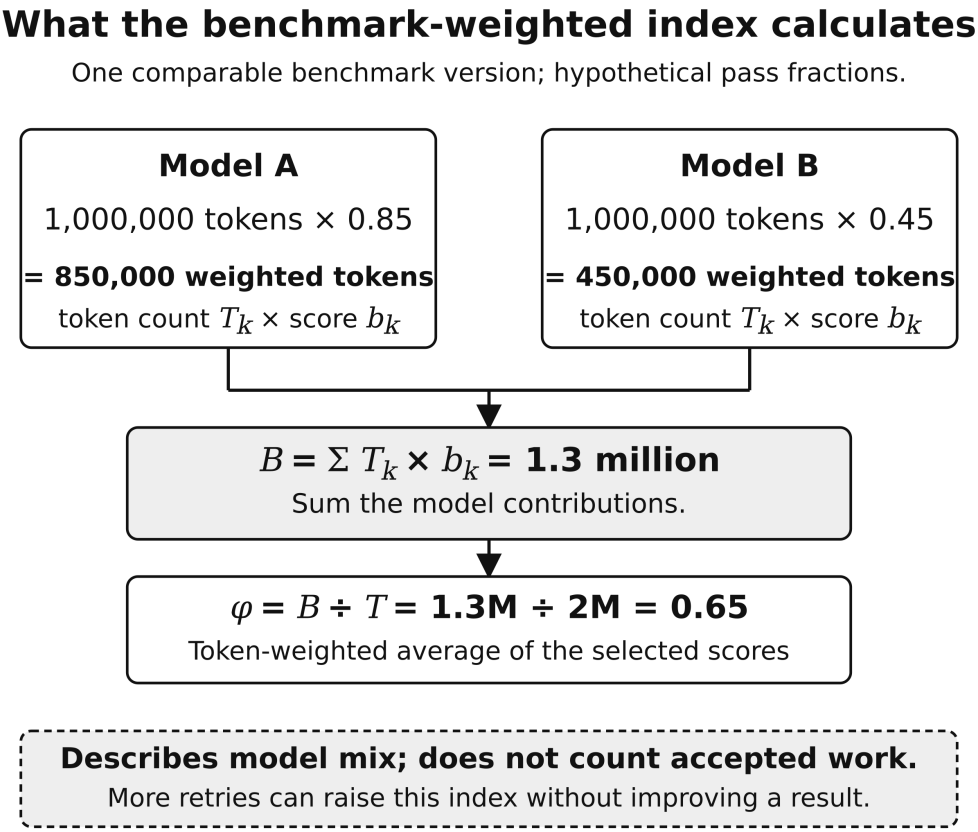


Diagram 8.2. The proposed benchmark-weighted index multiplies each model's tokens by a comparable benchmark score, sums the contributions and divides by total tokens to describe the model mix. All scores here are hypothetical; the calculation does not measure accepted work.

Explore this section in the full chapter →

Choosing the benchmarks

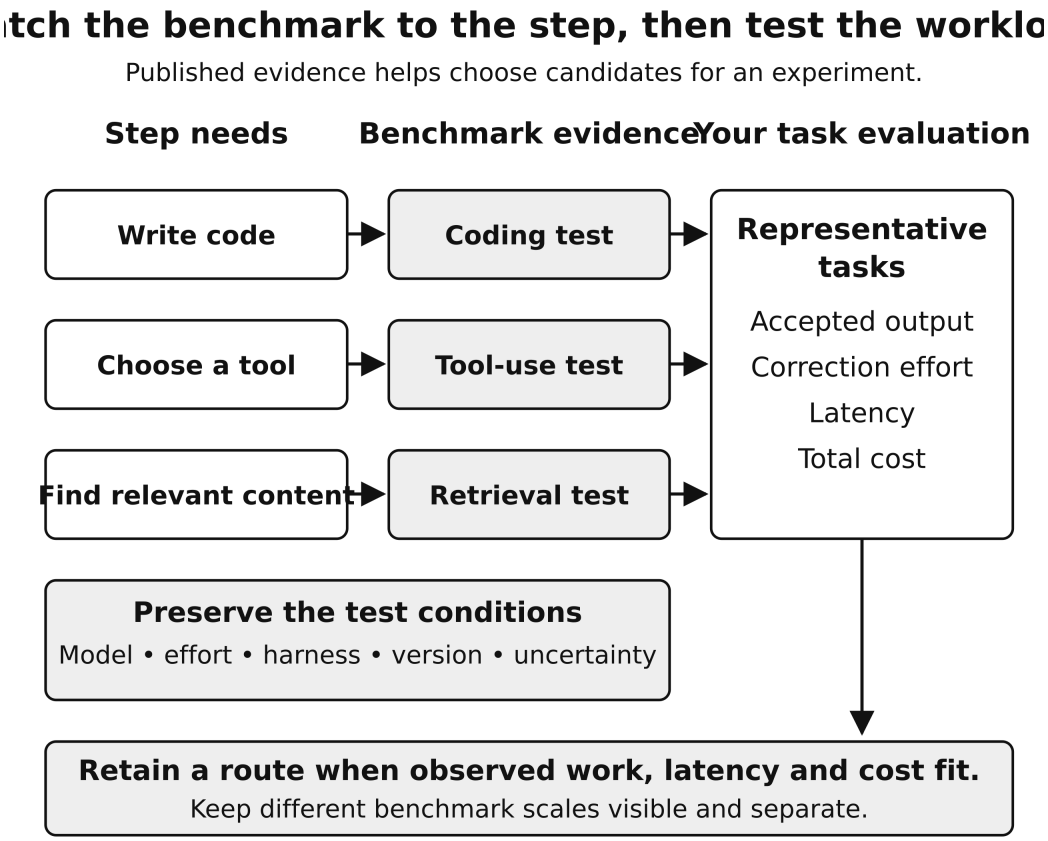


Diagram 8.3. Match the evidence to the skill a step needs, preserve its test conditions and evaluate the candidate on representative work. The routing decision depends on observed acceptance, correction, latency and total cost.

Explore this section in the full chapter →

The purpose question: what no consumption measure can say

So the measurement design needs a second mechanism: an audit question set, asked per task class, by people who can judge the answers:

- Does this step need a model at all? Formatting, validation and deterministic transformations may be better served by ordinary code, whose compute and maintenance still cost money.
- Does it need this class of model? Compare a smaller model on representative examples, including difficult cases and changes in the label set.
- Does it duplicate another workload? Two teams may have good reasons for separate systems, but the reasons should survive being asked.
- Is the architecture appropriate? Repeated retrieval, excessive context and unnecessary agent loops are testable design choices.

Explore this section in the full chapter →

Right-sizing: the cheaper ways to do the work

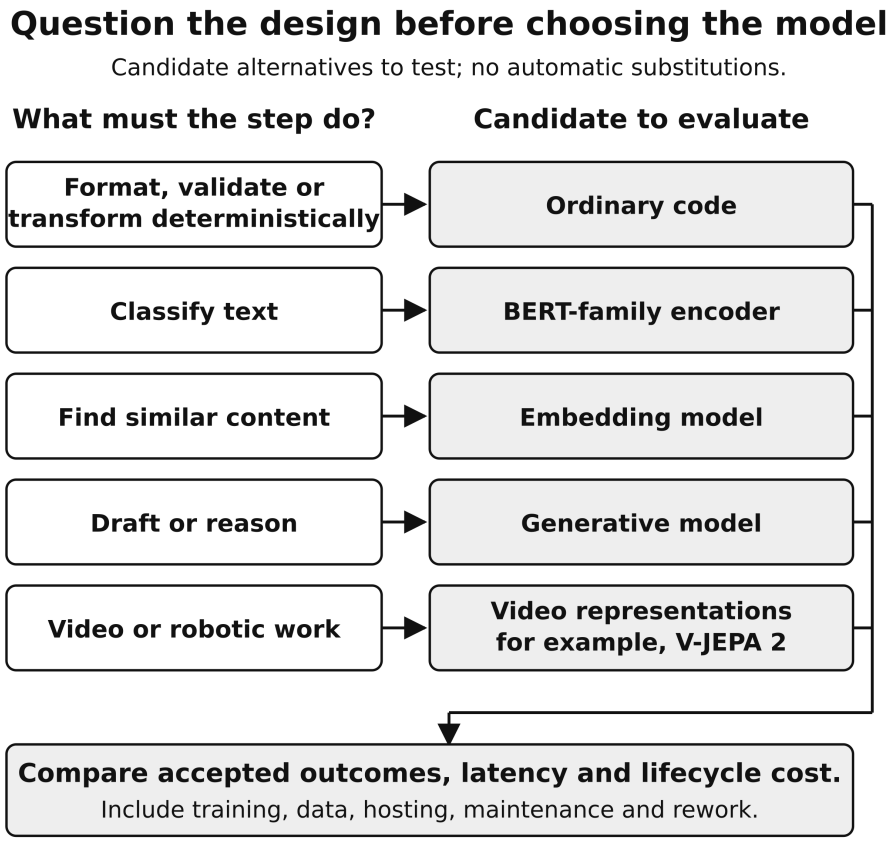


Diagram 8.4. Task requirements suggest candidates to evaluate, from ordinary code and encoders to embeddings, generative models and video representations. Compare accepted outcomes, latency and lifecycle costs before changing the design.

Explore this section in the full chapter →