NeurIPS 2026

SPIRIT

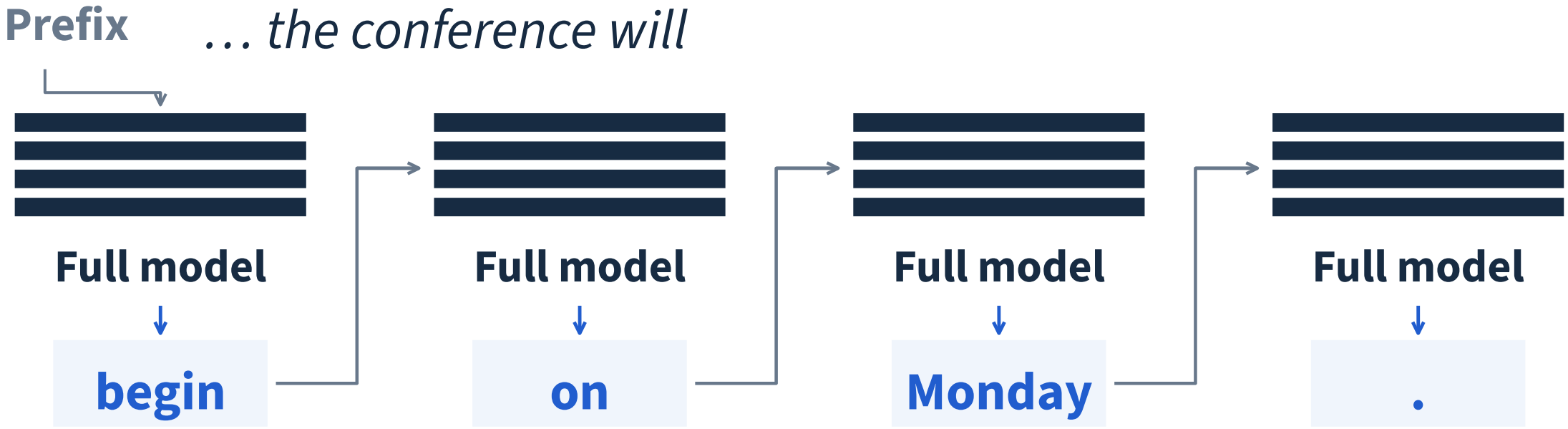
Speed-Driven Online Adaptation for Self-Speculative Decoding

Chenghao Liu, Chengbo Zhan, Hui Lu, and Jia Rao

The University of Texas at Arlington

One new token requires another full-model pass.

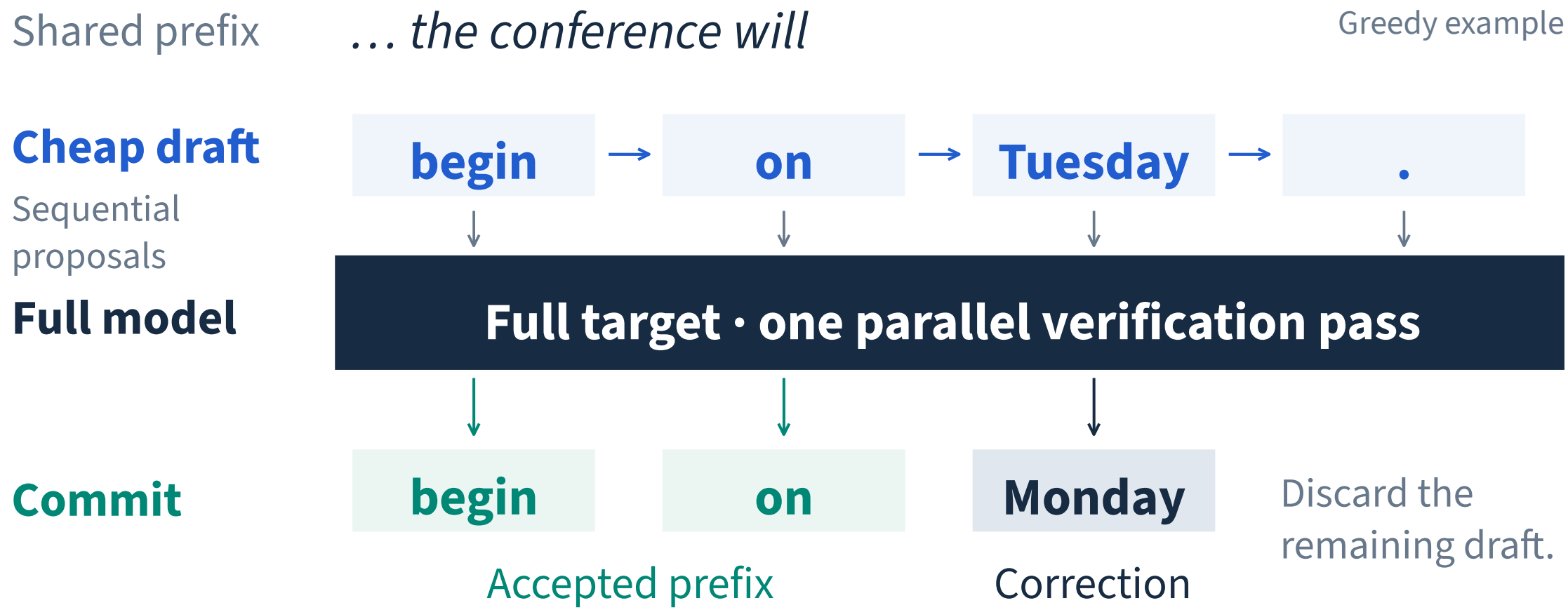
Autoregressive decoding



4 serial full-model passes → 4 new tokens

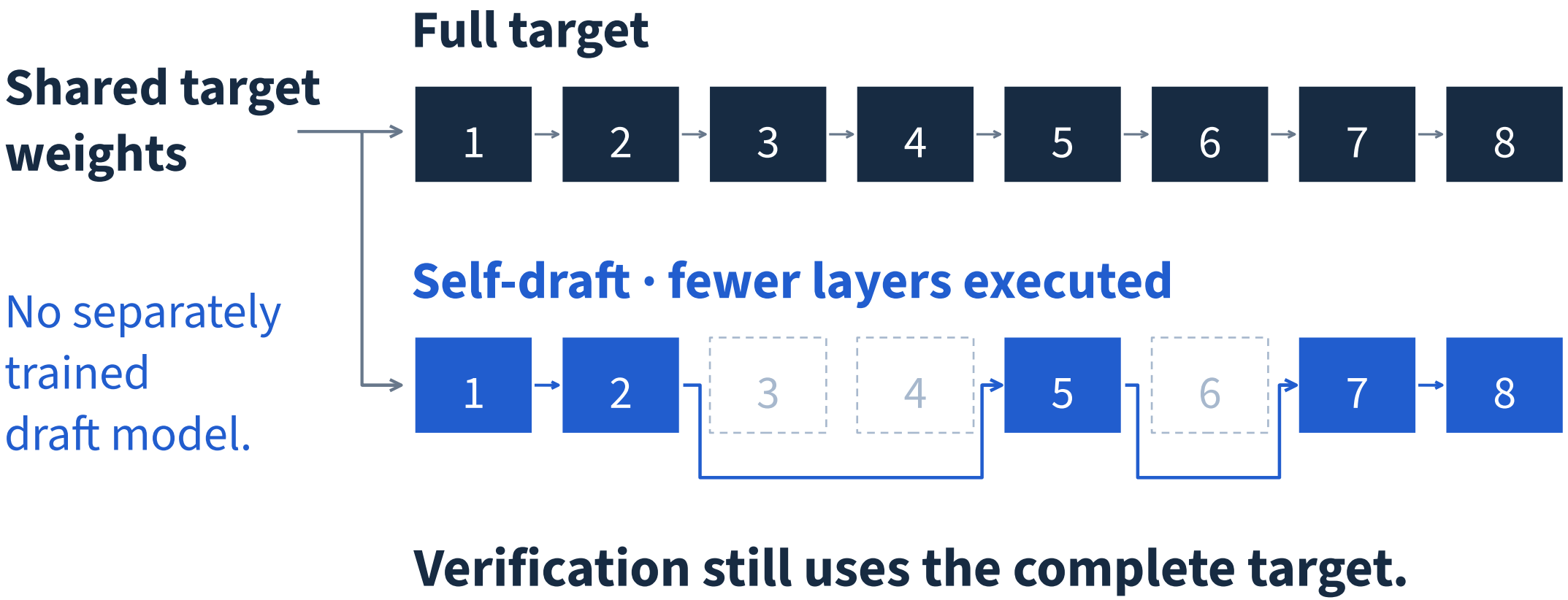
Speculate first. Verify in parallel.

A cheap drafter proposes a sequence; one target pass checks its tokens.



The target can draft for itself.

Reuse the target's weights; skip selected layers only during drafting.



Verification still uses the complete target.

Agreement alone does not determine speed.

Representative learning and ranking signals across three routes.

Route	Examples	Learning / ranking signal
Distill draft weights	**DistillSpec · Online SD** ICLR / ICML 2024	Target-distribution alignment
Train draft heads	**EAGLE · Medusa** ICML 2024	Feature / future-token prediction
Select layers to skip	**SWIFT · CLaSp** ICLR / ACL 2025	Token / hidden-state matching

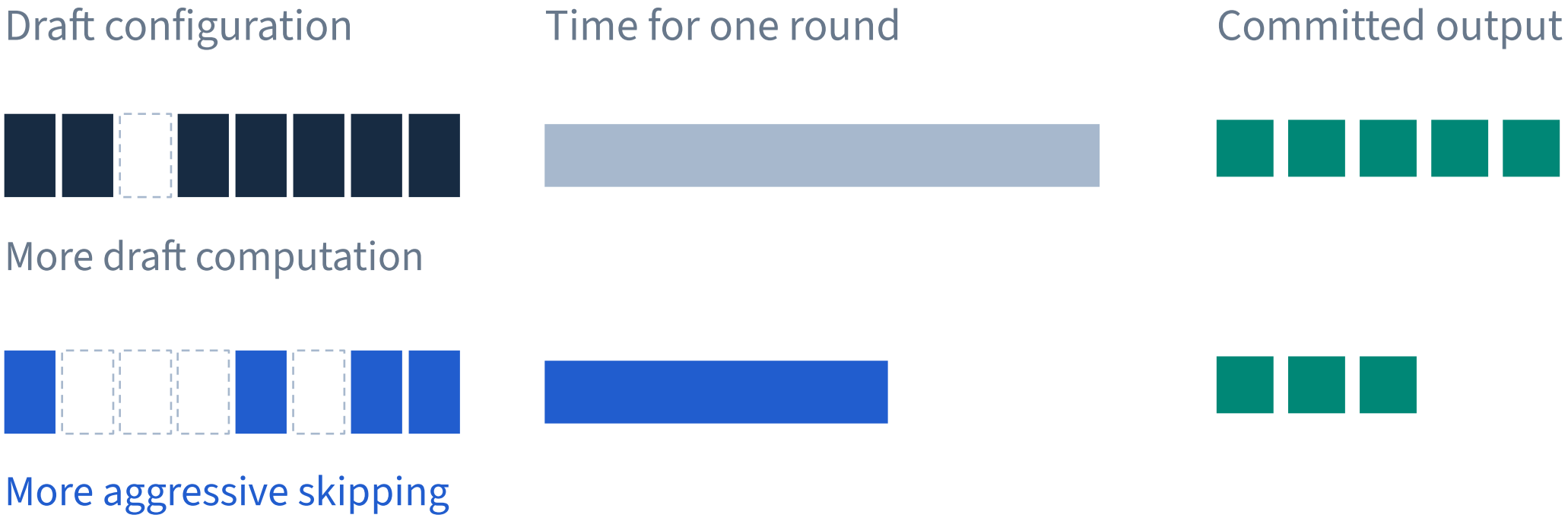
SPIRIT **Optimize from productive runtime windows.**
Length-calibrated throughput guides online selection.

Draft & Verify: offline development-set timing → fixed skip configuration.

DistillSpec / Online SD / EAGLE / Medusa (2024) · SWIFT / CLaSp (2025) · Primary sources in notes

Skipping saves work—but can reduce agreement.

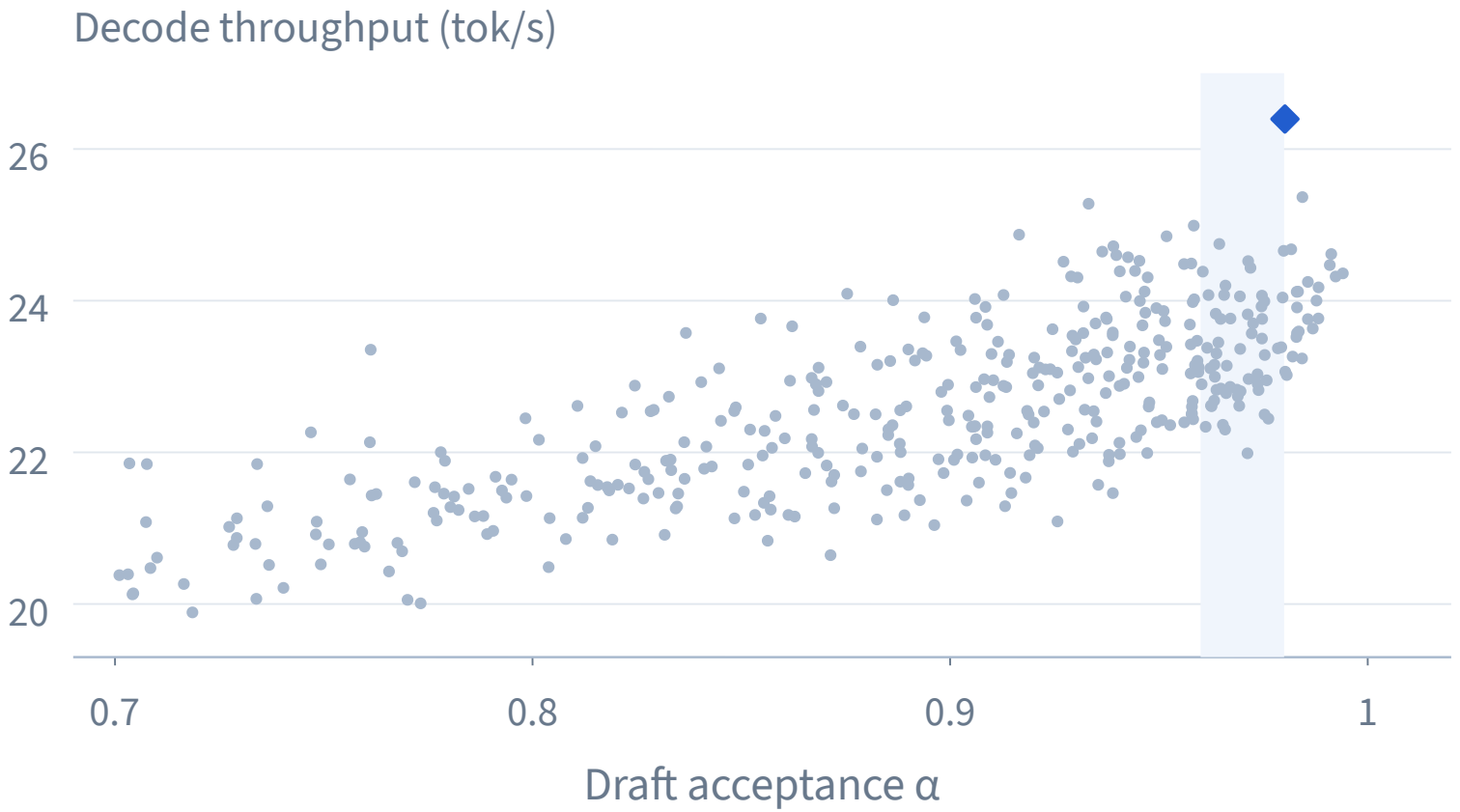
Illustrative configurations: execution cost and token yield change together.



$$\frac{\text{Committed output}}{\text{Elapsed decode time}} \longrightarrow \textbf{Optimize output per unit of time.}$$

High agreement does not imply high throughput.

LLaMA-2-13B · CNN/DM · FP16 · same 24 prompts per configuration



At ≈96–98% acceptance

21.99–26.39

tokens per second

Same agreement band.
Different execution cost.

Draft configurations are workload dependent.

Prior evidence from SWIFT · LLaMA-2-13B · speedup over each task's AR

Evaluation task	Mask optimized for Storytelling	Mask optimized for Reasoning
Storytelling TinyStories	**1.47×**	1.28×
Reasoning GSM8K	1.01×	**1.12×**

The ranking reverses across workloads.

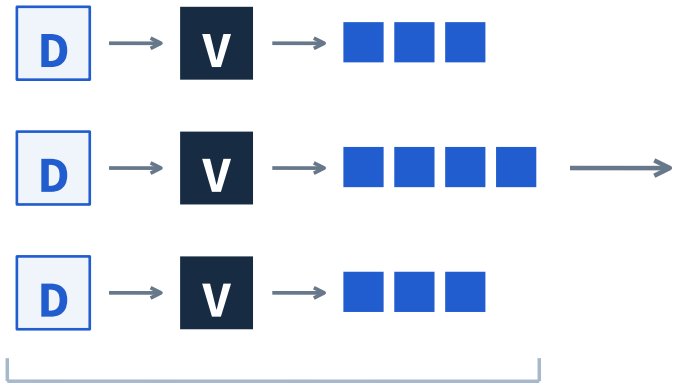
Best per workload

Window feedback drives online adaptation.

Capture agreement and intrinsic speed advantage while committing output.

Productive window

Output keeps advancing.



Window feedback

Realized throughput

$$S_w = G_w / T_w$$

Committed tokens per second

Draft acceptance

$$\alpha_w = A_{d,w} / D_w$$

Accepted / proposed tokens

WITHIN A REQUEST

Length calibration + EMA

Score Q_w

Optimize ratio + pattern

AT REQUEST HEADS

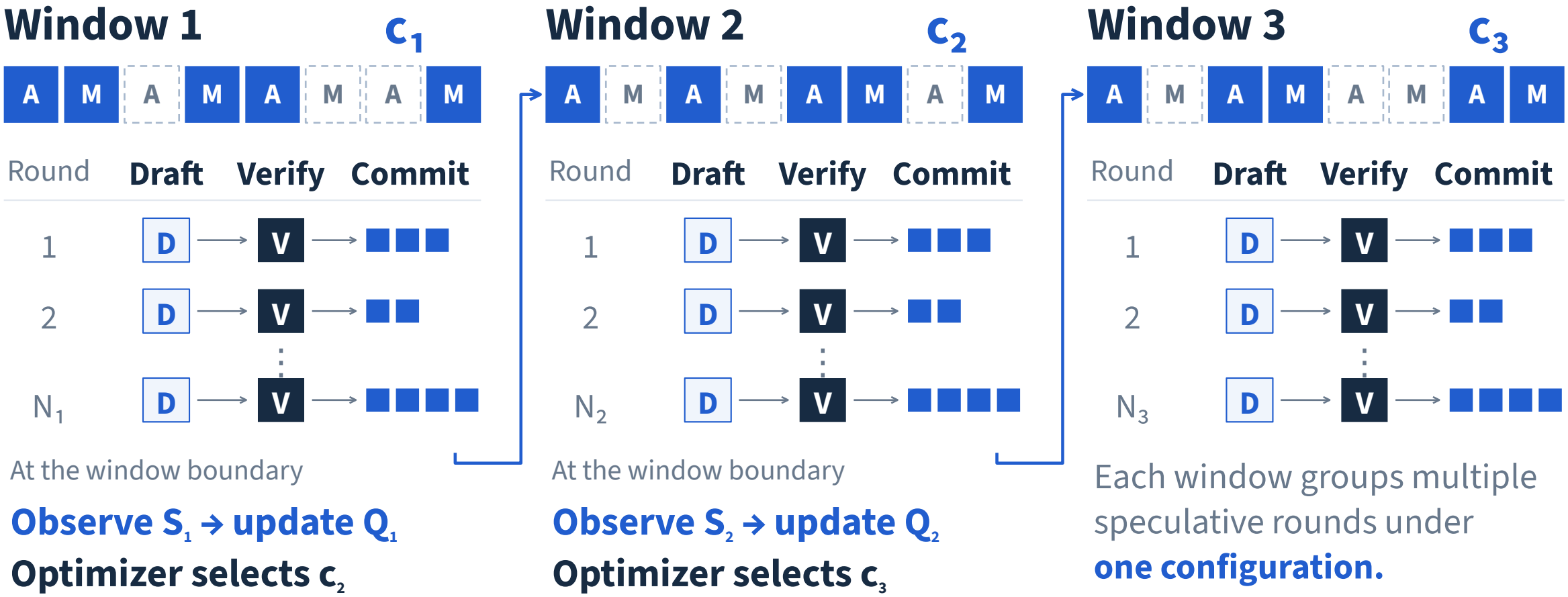
Raw S, α vs. profile history

Detect shifts; adapt

Choose the next configuration

Measure a window. Improve the next.

Each window commits output while evaluating its configuration.



Count every token that reaches the response.

A schematic window with three speculative rounds

Committed tokens



Teal: accepted draft token Navy: verifier token

$$G_w = A_{d,w} + N_w$$

9 = 6 + 3

Elapsed decode time: 0.60 s

S = 9 / 0.60 = 15 tok/s

Each window captures the joint signal.

All quantities come from the same productive runtime window.

Realized throughput = **Useful token yield** × **Execution rate**

Agreement

Intrinsic speed advantage

$$\frac{G_w}{T_w} = \frac{A_{d,w} + N_w}{D_w} \times \frac{D_w}{T_w}$$

Committed tokens per decode second

Accepted draft + verifier tokens per drafted token

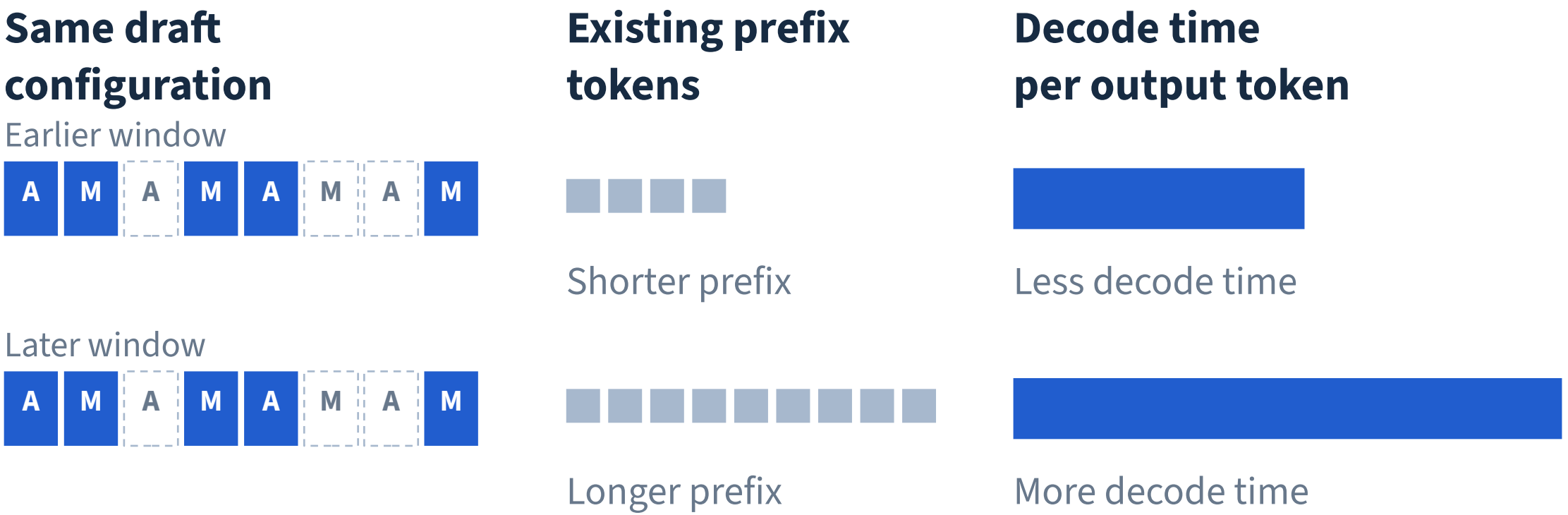
Drafted tokens per decode second

$$\alpha_w = A_{d,w} / D_w \qquad \beta_w = N_w / D_w \qquad \nu_w = D_w / T_w$$

$$S_w = (\alpha_w + \beta_w)\, \nu_w$$

Longer prefixes change the cost of decoding.

A later window is not automatically a worse configuration.



Schematic: token counts and bar lengths are illustrative.

Calibrate windows for prefix length.

Separate configuration quality from the cost of a growing prefix.

OBSERVE EXECUTION

(x_i, y_i)

$y_i = T_i / G_i$

x: effective prefix tokens

y: seconds / committed token

FIT THE LENGTH MODEL

$\hat{y}(x) = a + bx$

Learn a, b from

measured (x, y) pairs.

Step-weighted least squares.
Both predictions use this fit.

ESTIMATE LENGTH EFFECT

$f_{fit} = \hat{y}(x_w) / \hat{y}(x_{ref})$

x_w: current window prefix

x_{ref}: request's reference

$f_{fit} \rightarrow f_{len}$

Ramp in fit; limit correction.

MEASURED THROUGHPUT

$S_w = G_w / T_w = 1 / y_w$

Actual output / decode time.

CALIBRATE

$\tilde{S}_w = S_w \times f_{len}$

Correct measured throughput.

SMOOTH

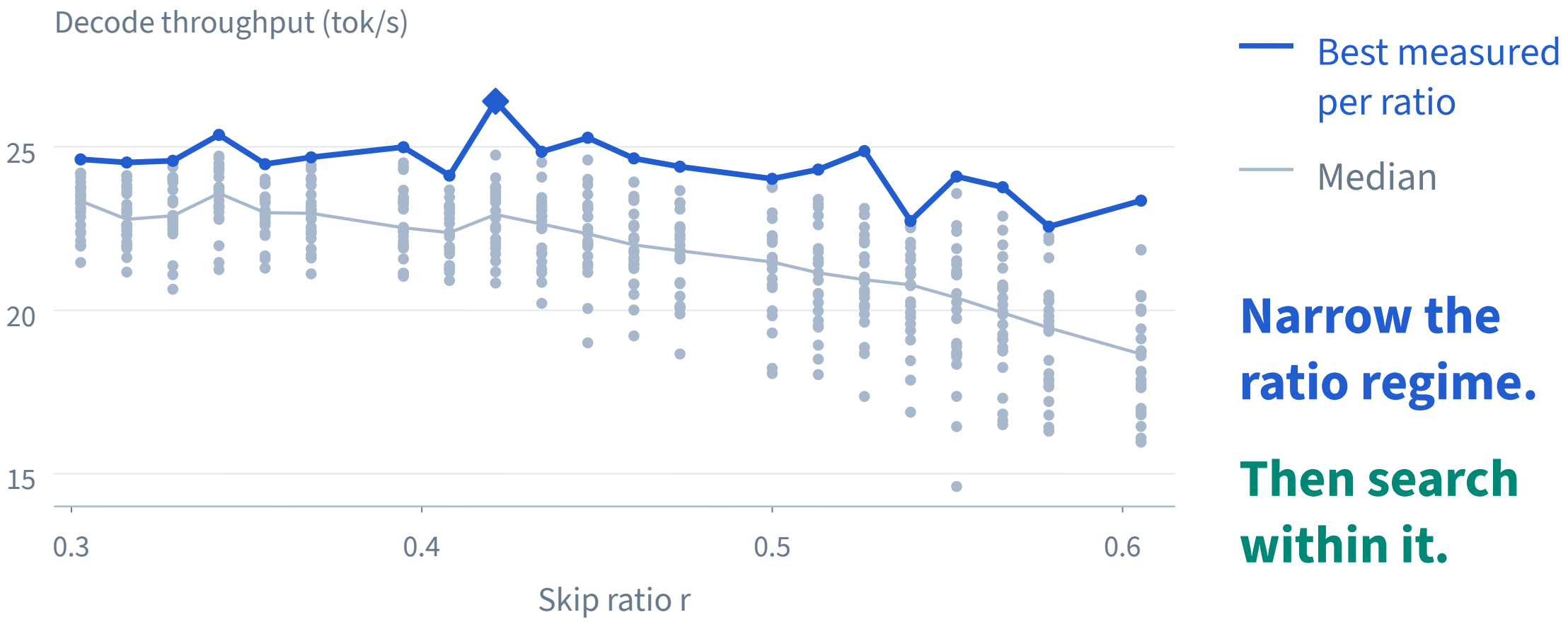
$Q_w = EMA[\tilde{S}_w]$

Score for online selection.

Runtime evidence → **comparable score** → next configuration

Ratio first. Pattern still matters.

504 measured configurations reveal both levels of the search.



Narrow the ratio regime.

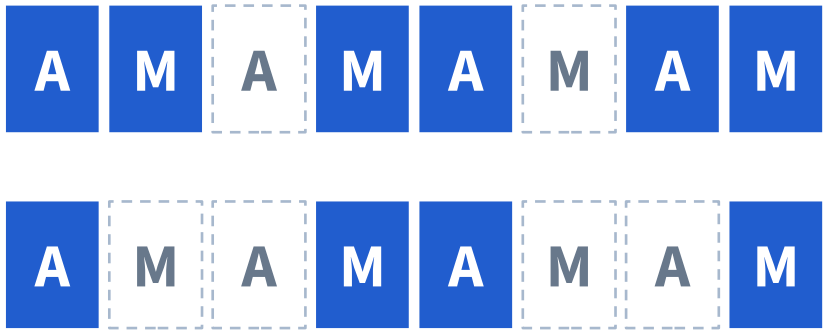
Then search within it.

SPIRIT · Figure 1b · LLaMA-2-13B · CNN/DM · FP16 · 24 prompts/configuration

Search the ratio, then refine the pattern.

Factor the draft configuration into a skip ratio and a layer pattern.

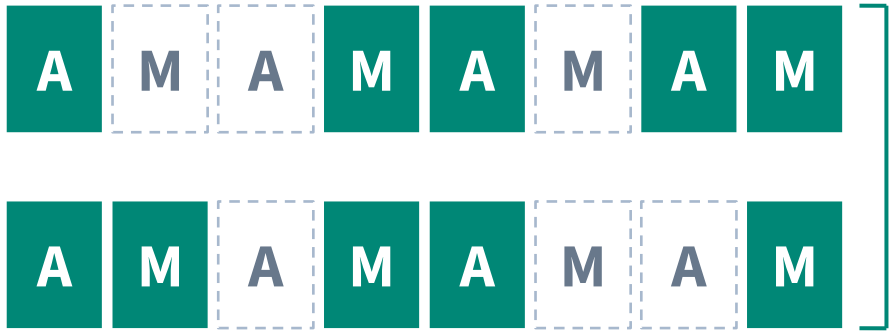
01 Skip ratio r



Fewer skips

More skips

02 Layer pattern L



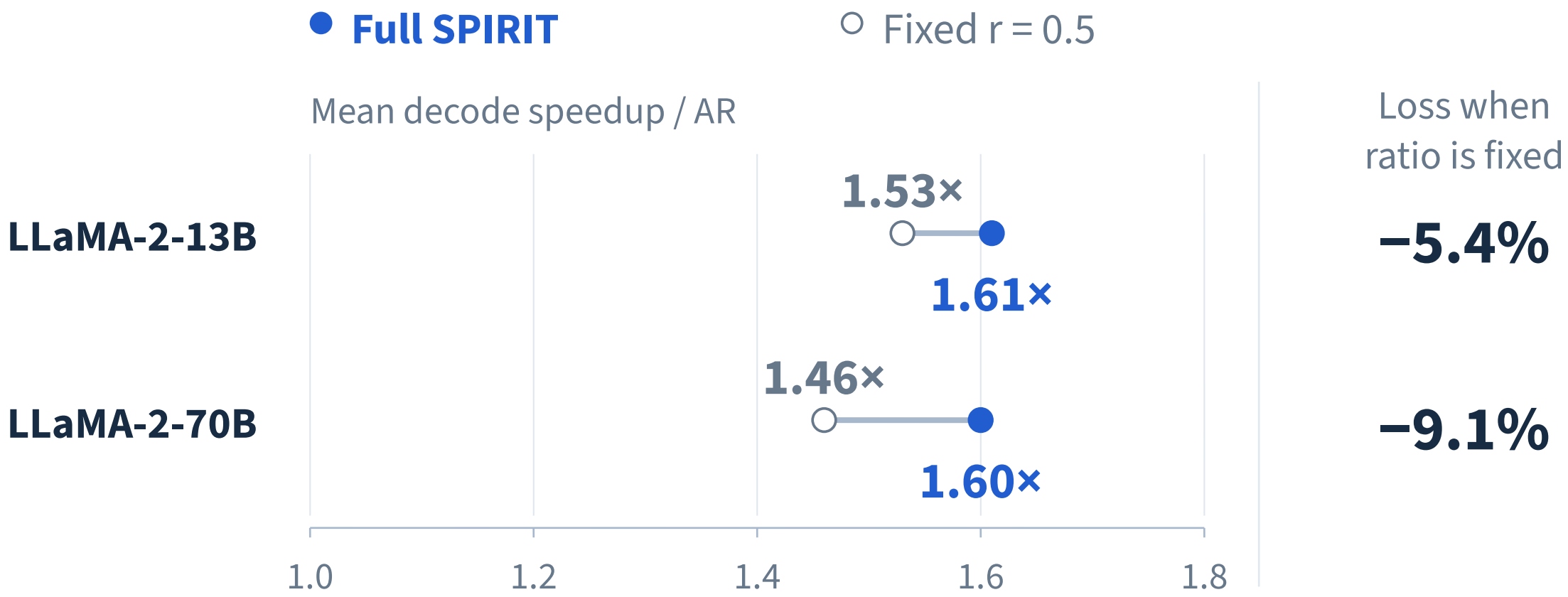
Localize a high-throughput regime on the agreement–speed frontier.

Refine layer positions within a smaller search space.

Focus the search budget on layer placement.

Ratio search adds gains beyond pattern tuning.

A2 fixes r = 0.5; pattern refinement remains enabled in both conditions.

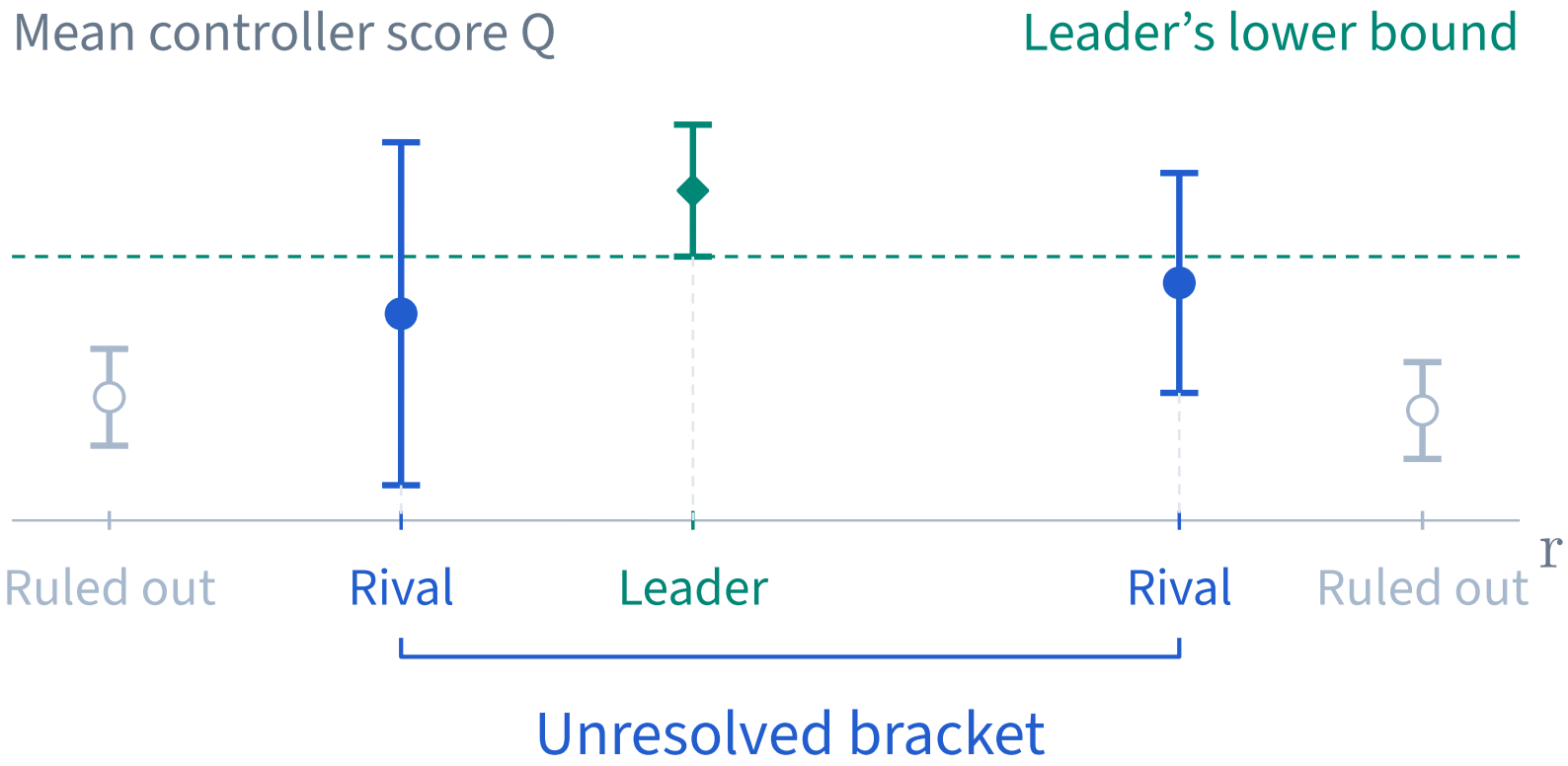


Even with pattern tuning, the ratio matters.

Paper Table 4 / §4.4 · CNN/DM, TinyStories and GSM8K · FP16 self-drafts

Ratio search: use confidence bounds to identify rivals.

The leader has the highest mean; a lower mean alone does not rule out a ratio.

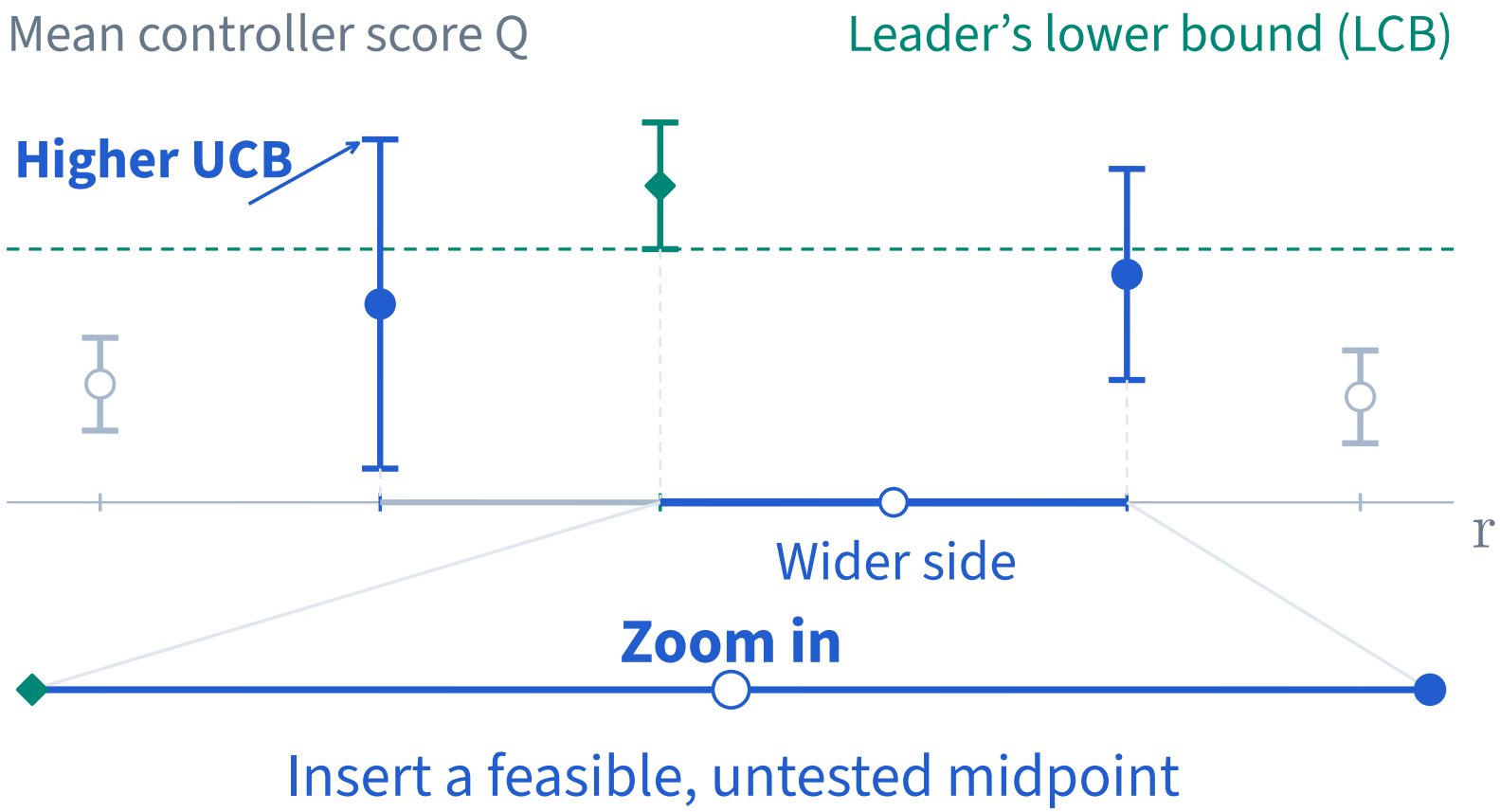


- ◆ **Leader**
  Highest mean
- ● **Rival**
  Upper ≥ leader's lower bound
- ○ **Ruled out**
  Upper < leader's lower bound

Collect minimum window evidence at each active ratio before ruling it out.

Zoom in on the unresolved ratio region.

Geometry guides refinement; uncertainty guides remeasurement.



Refine the wider side.

Bisect its largest unresolved gap.

When refinement is exhausted:

Remeasure the leader or highest-UCB rival.

Measure. Recompare. Choose the next ratio.

A midpoint can narrow the unresolved bracket without becoming the leader.

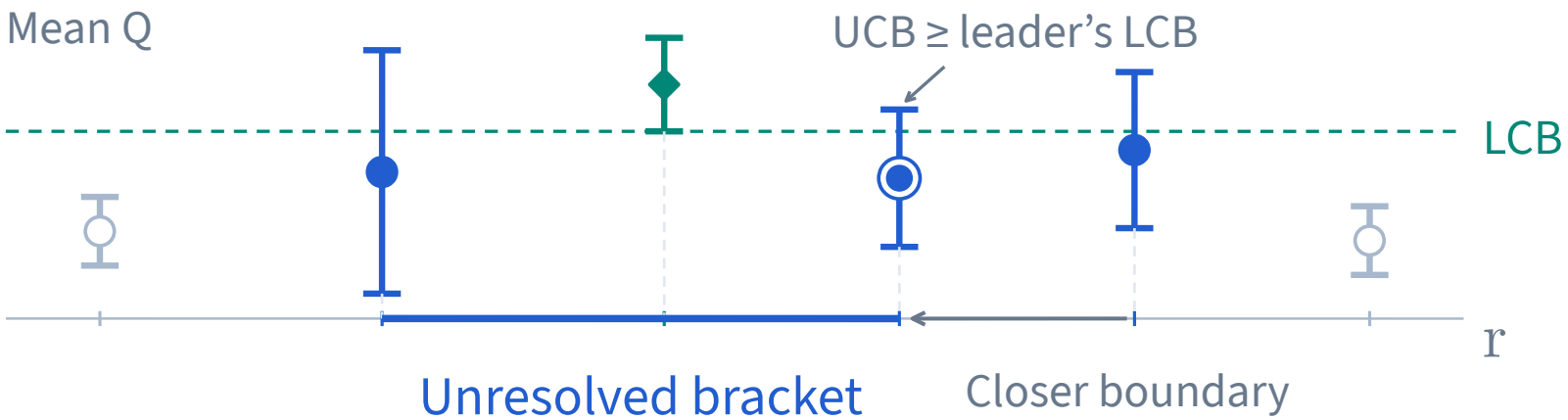
Decode + commit



Tokens + runtime

- Previous midpoint
  Now a rival
- Leader unchanged

Update mean + bounds



Rivals remain?

Yes

No

Choose next ratio

Zoom or remeasure

Confirm leader
Validate edges

Pass → Fix r⋆

Checks unresolved: remeasure.

Decoding doubles as pattern evaluation.

At the locked skip ratio, propose a layer pattern and measure it in a productive window.

EXPLORE · p = 0.96

Random proposal

Sample layer positions at the fixed skip count.



USE PRIOR SCORES · p = 0.04

Bayesian proposal

Use patterns already scored at this ratio (GP-UCB).



OR

Run one window



Tokens committed



Observed score

Q_{w+1}

If the score improves

Retain best observed pattern

$L^{\star}$



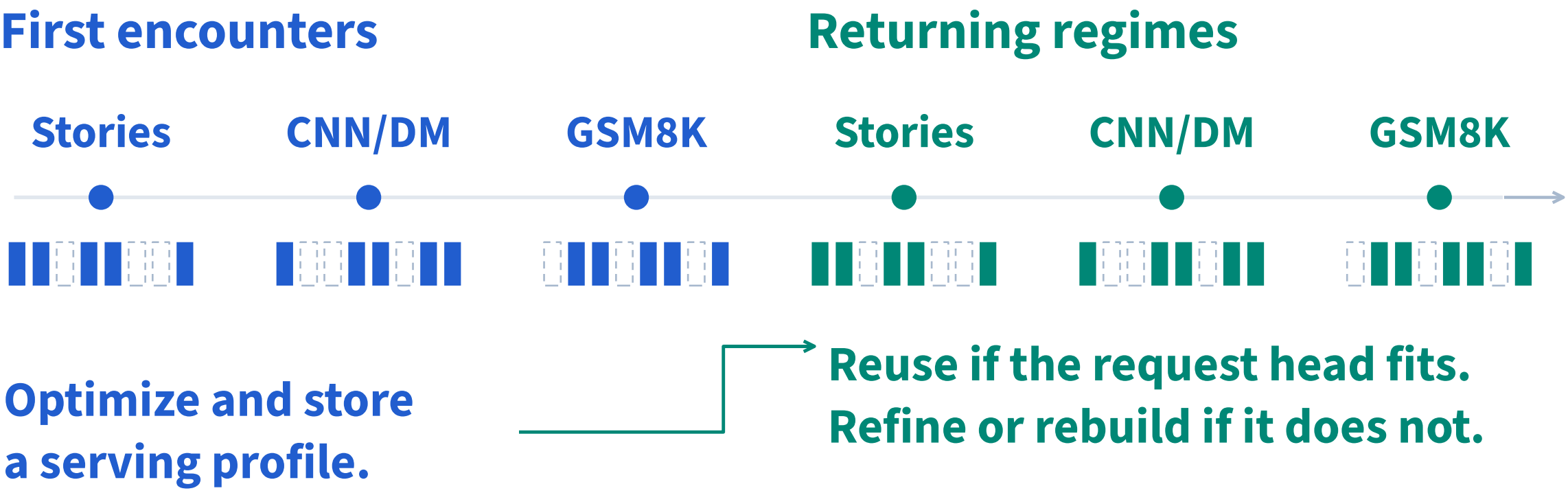
Update optimizer

Record (L, Q)

Stop when stable or the request budget is spent.

Workloads return. Search need not restart.

Remember effective configurations and the request behavior they explain.



The controller uses runtime evidence, not a task label.

A profile is a configuration with a memory.

Store what ran—and what its earlier request heads looked like.

Draft configuration

$(r^\star, L^\star)$

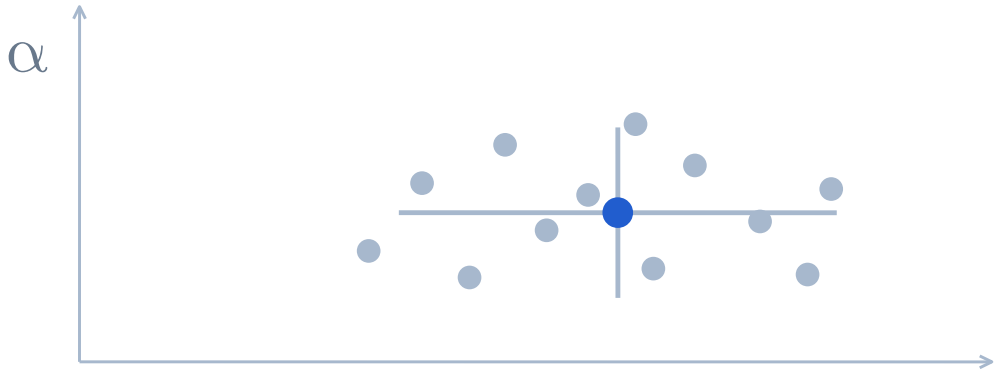


Skip ratio + sublayer pattern

No task label is required.

This profile's request history

Raw throughput S · draft acceptance α



One point per prior request head.

Historical mean ± SD

Check for drift at the start of each request.

Compare opening-window throughput and acceptance with the active profile's history.

New request

Opening windows

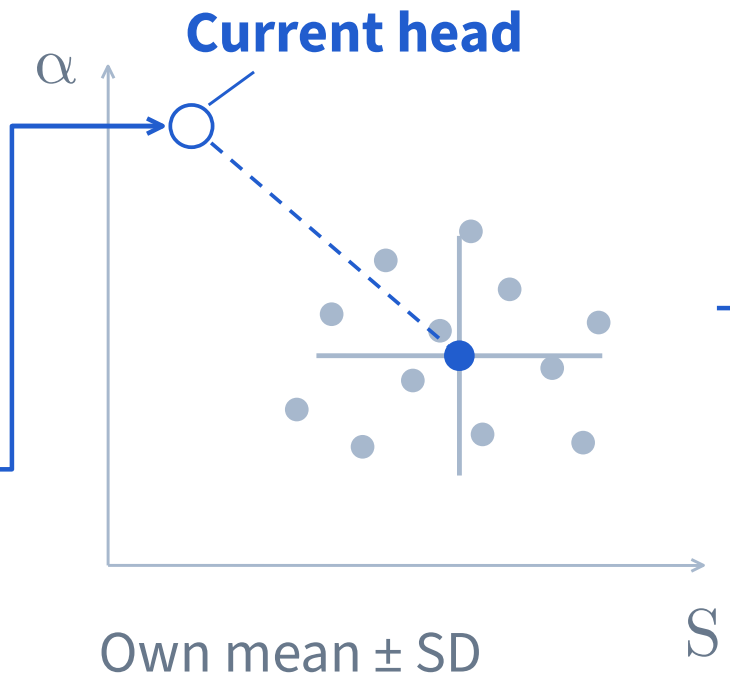
Active profile

A

Measure S, α

Raw throughput + draft acceptance

Compare with history



Shift evidence?

Compatible or not yet conclusive

Continue serving and monitoring

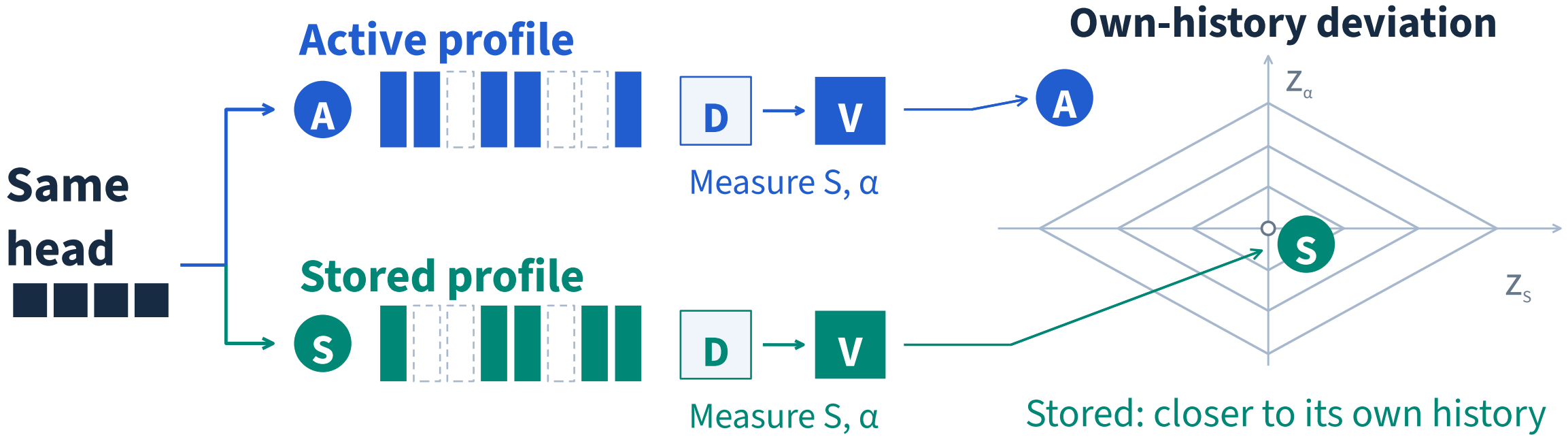
Strong or repeated deviation

Probe stored profiles

Repeated deviation is assessed across request heads.

Probe profiles on the same request head.

Replay the same head under each profile. Trial outputs are not committed.



Standardize each trial using that profile's own historical mean and standard deviation.

$$d = |z_s| + \omega\,|z_\alpha|$$

Lower d means a closer fit.

Reuse as much as the evidence supports.

Switch and Switch + refine require all three checks.

Closer fit
Lower own-history mismatch

Higher probe merit
$J(p) = \Delta S + \kappa \Delta \alpha$ (vs active)

Safe throughput
Meet the throughput floor

Stay



Keep active ratio + pattern.

Fits or evidence is inconclusive.

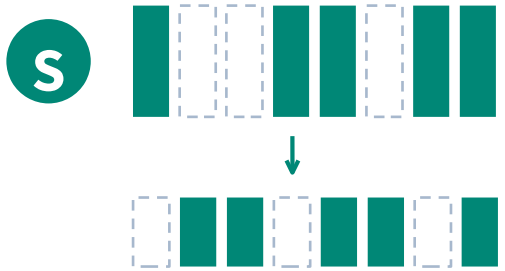
Switch



Restore stored ratio + pattern.

An eligible profile is ready to reuse.

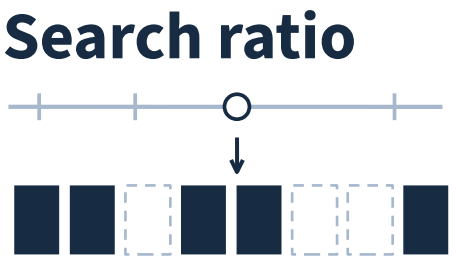
Switch + refine



Keep ratio. Refine pattern.

Ratio fits; refine locally.

Restart

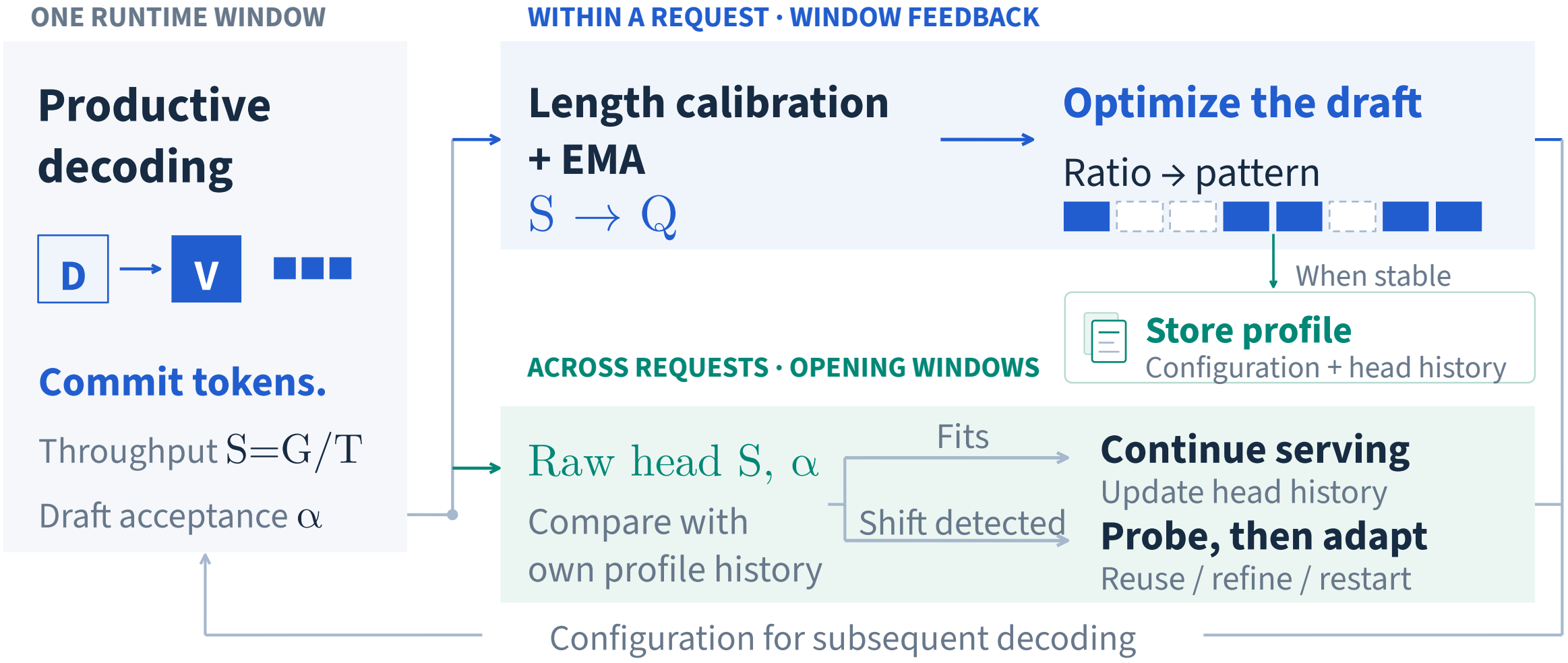


Search the ratio, then the pattern.

Persistent shift; no feasible profile.

One execution stream. Two control loops.

Window feedback guides optimization; request-head evidence guides adaptation.



Include adaptation in the speedup.

Speedup = committed-token throughput / AR throughput

Included in online results

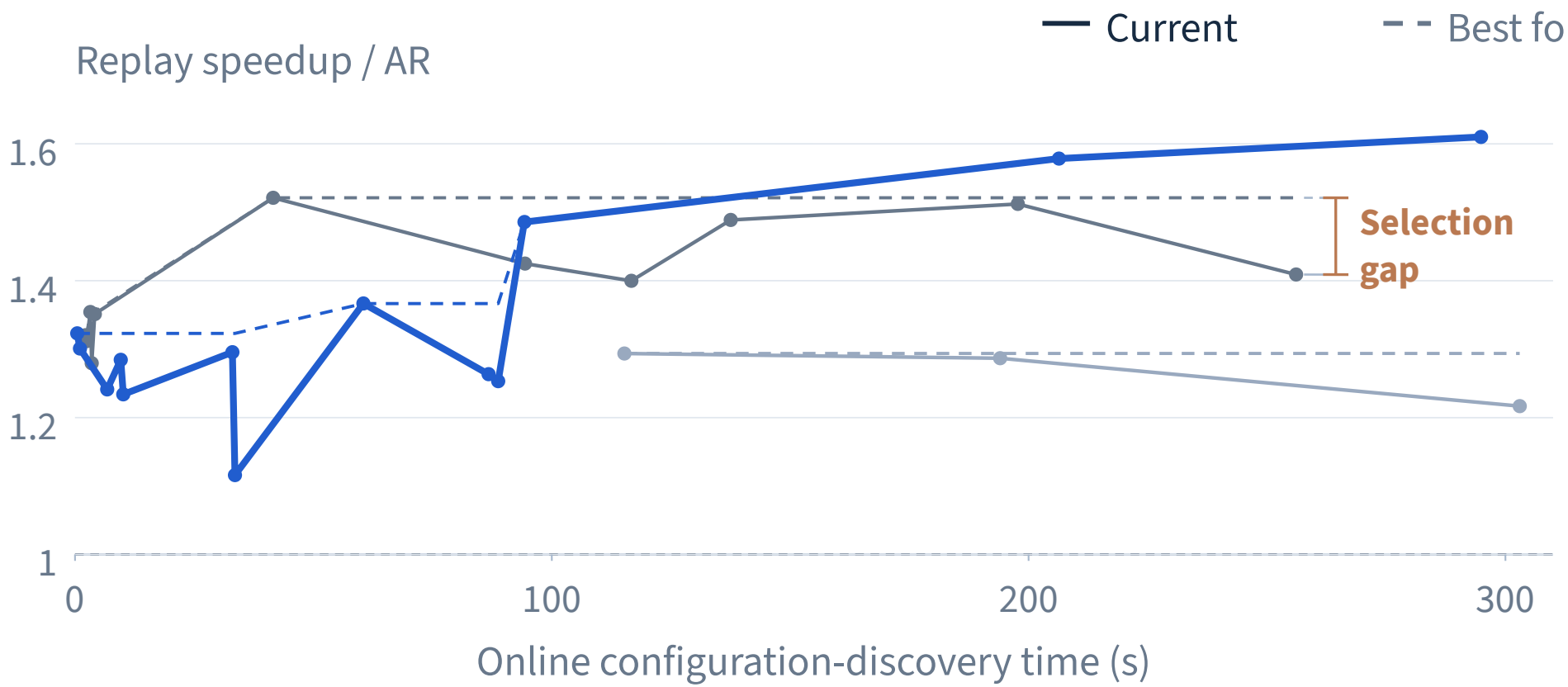
Drafting and full verification
Productive online search
Controller and profile updates

Outside decode timing

Prefill
Model loading

Reliable selection starts with the right metric.

LLaMA-2-13B · CNN/DM · FP16 · common 20-request replay probe



SPIRIT 1.61×
Final = best found

SWIFT 1.41×
Best found: 1.52×

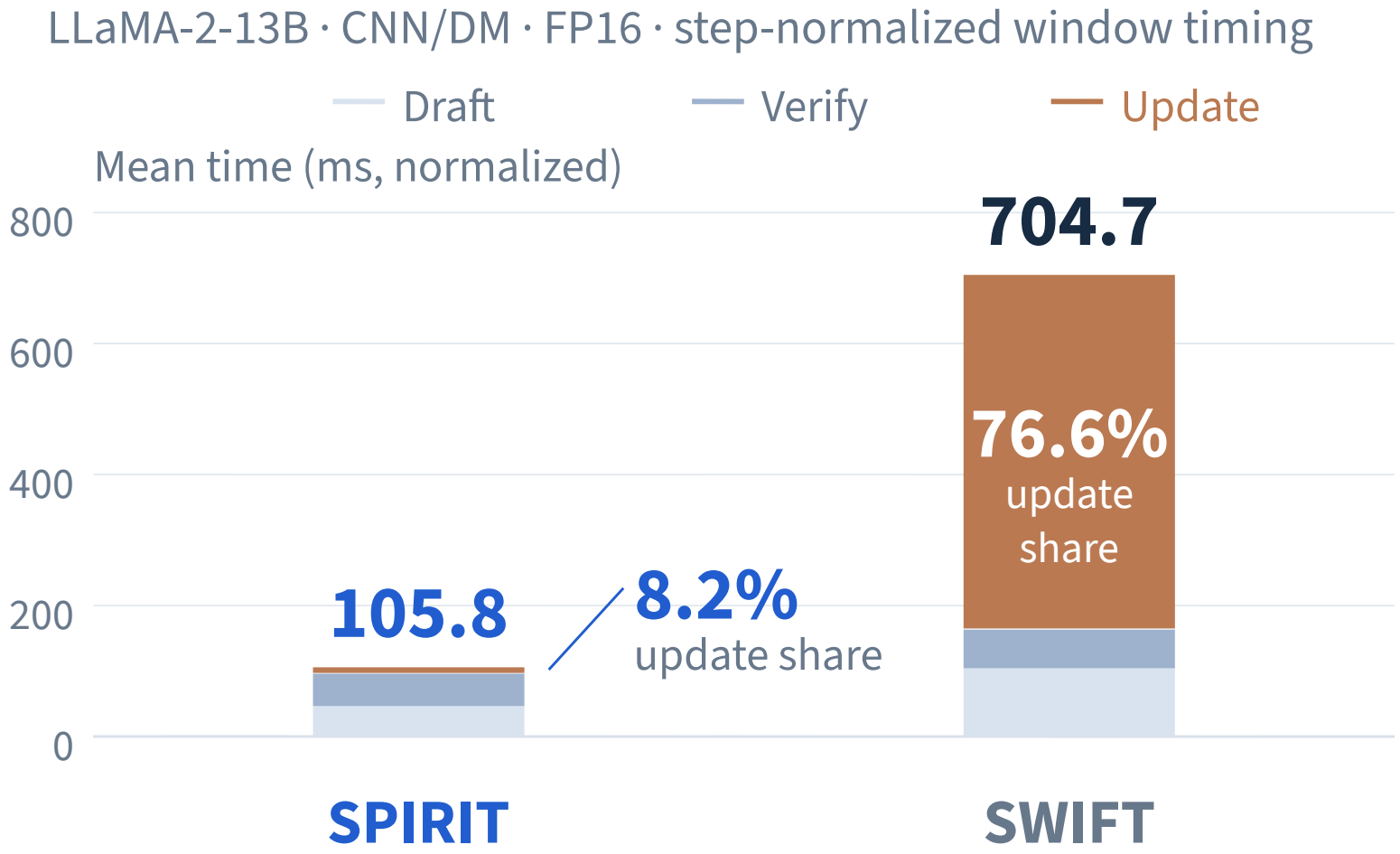
D&V 1.22×
Best found: 1.29×

Agreement-based scores can misrank configurations by decode speed.

Throughput feedback aligns discovery and final selection with actual speed.

Paper Fig. 3a / §4.3 · Published replay-curve values

Fast iteration without extra scoring passes.



SPIRIT

Draft → Verify → Commit

Window counters → score

No extra scoring forward

Update once per window.

SWIFT

Extra candidate forward

on generated context

→ token-matchness score

Speculative decoding supplies both output and optimization feedback.

Paper Fig. 3b / §4.3 · 13 requests · one seed · Other execution <1%

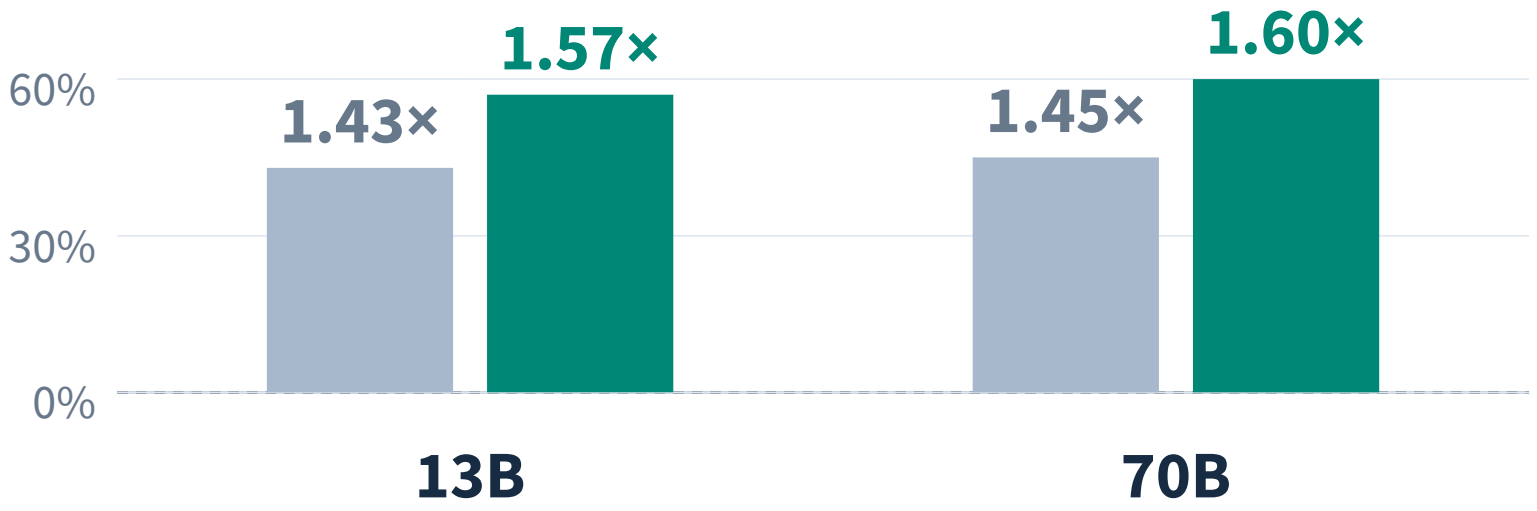
Profile reuse raises throughput by 9.8–10.3%.

Recurring six-phase stream · 600 requests · LLaMA-2 · FP16

First pass | **Workloads return**

Stories	CNN/DM	GSM8K	**Stories**	**CNN/DM**	**GSM8K**

Full-stream throughput gain over AR (%)
Bar labels: decode speedup (×)



+9.8–10.3%

Higher throughput vs. no reuse

100 requests per phase

Paper Tables 3–4 / §§4.2, 4.4 · No-reuse ablation

SPIRIT leads in all nine model–task settings.

LLaMA-2 family · FP16 self-drafts · comparison with SWIFT and Draft & Verify

Decode speedup / AR

Method	13B-Chat			13B			70B		
	CNN/DM	GSM8K	Stories	CNN/DM	GSM8K	Stories	CNN/DM	GSM8K	Stories
SWIFT	1.19	1.21	1.10	1.46	1.40	1.36	1.29	1.39	1.45
D&V	1.33	1.20	1.16	1.54	1.39	1.46	1.41	1.42	1.49
SPIRIT	**1.37**	**1.27**	**1.20**	**1.67**	**1.49**	**1.68**	**1.50**	**1.57**	**1.74**

Peak: 1.74× on 70B · TinyStories; ≈16.8% higher throughput than D&V.

Paper Table 1 · Decode speedup over autoregressive decoding

SPIRIT remains ahead on the SpecBench suite.

LLaMA-2-13B · same self-speculative backend · independent controller seeds

+17.2%

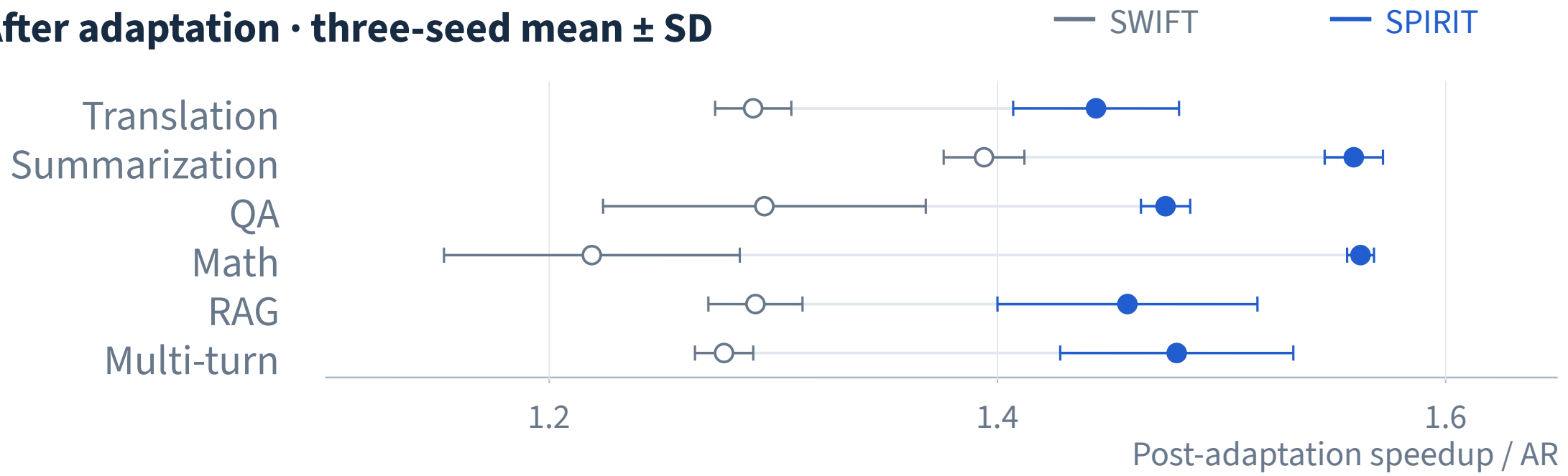
full-stream throughput

Includes online search + profile updates.

1.247× → 1.462×

SWIFT → SPIRIT

After adaptation · three-seed mean ± SD



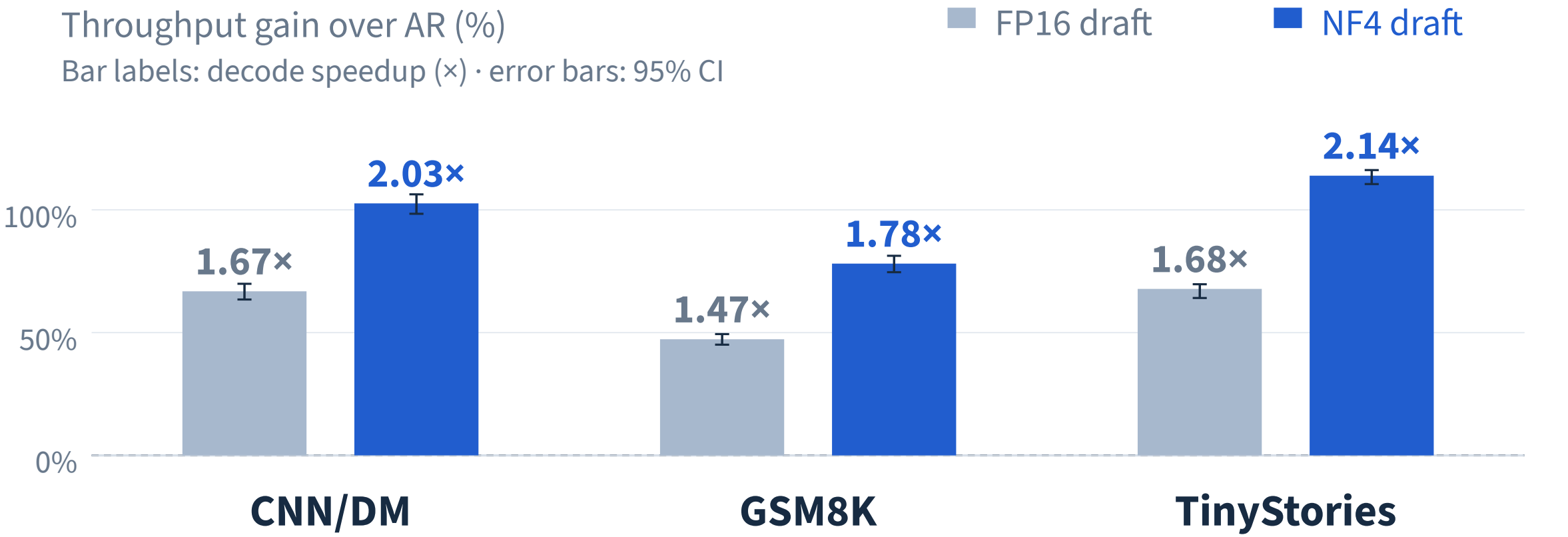
Paper §4.5 / Table 7 · Full-stream and post-adaptation scopes shown separately

NF4 drafts further accelerate decoding.

Three tasks · LLaMA-2-13B · NF4 vs FP16 self-drafts · same FP16 verifier

Throughput gain over AR (%)

Bar labels: decode speedup (×) · error bars: 95% CI



+23.6% overall throughput **1.60× → 1.98×** 95% CI: +21.4% to +26.3%

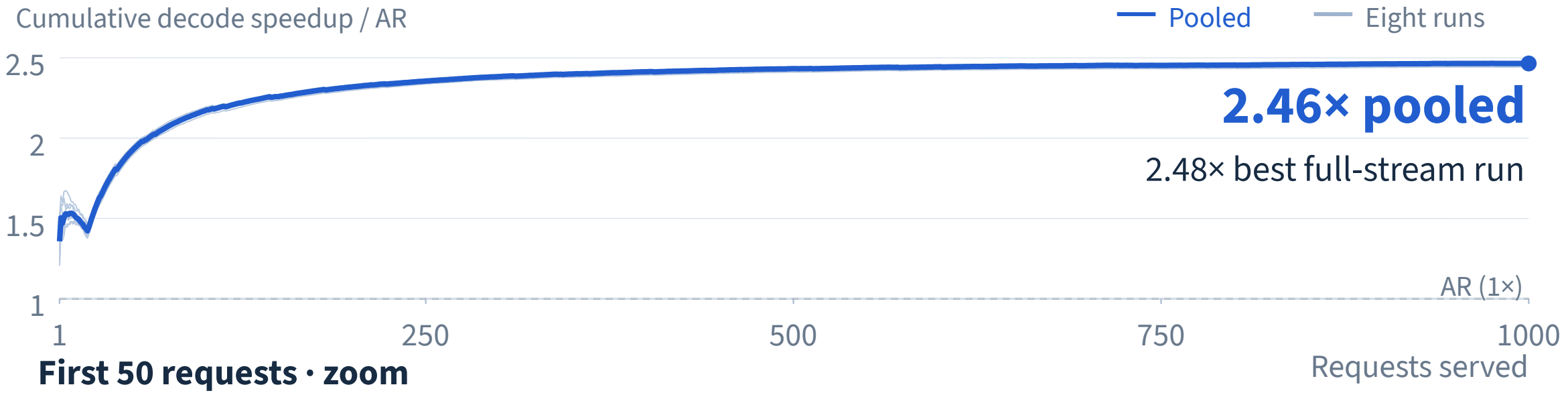
Paper Table 5 / §4.5 · Request-paired bootstrap 95% confidence intervals

EXPERIMENTS · Independent long-stream study

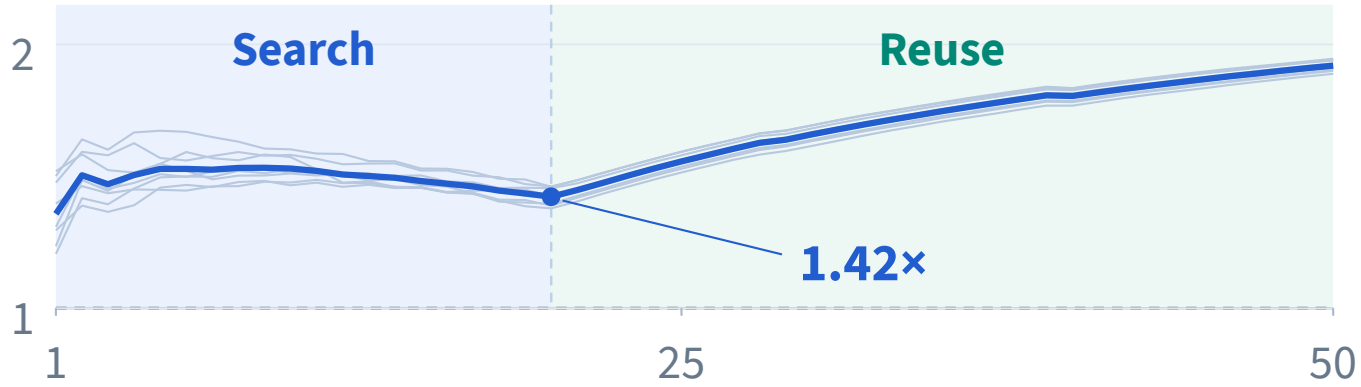
Decoding gains persist as optimization costs are amortized.

Stable TinyStories workload · 13B · fused NF4 draft / FP16 verifier

Cumulative decode speedup / AR



First 50 requests · zoom



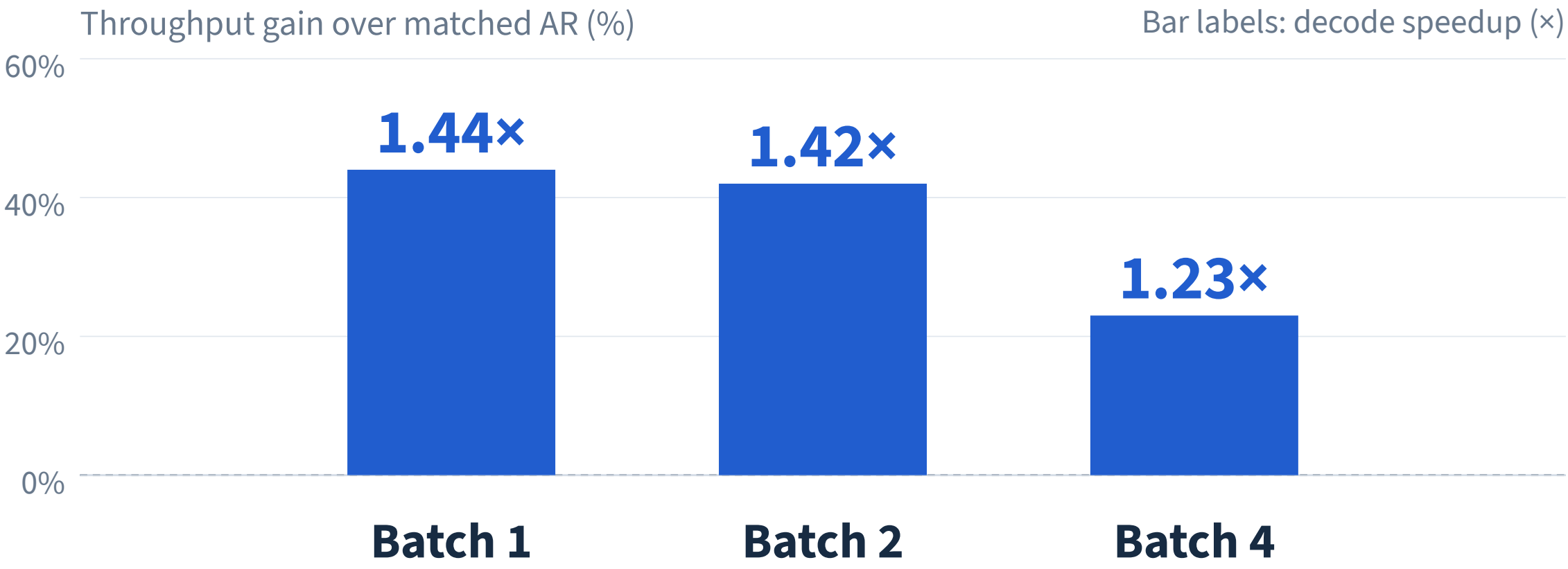
Search already accelerates decoding.

Later requests amortize the initial optimization.

Paper Fig. 4 / Appendix C.7 · Decode timing includes online search + profile updates

SPIRIT generalizes to batched decoding.

LLaMA-2-13B · each batch size is compared with its own autoregressive baseline

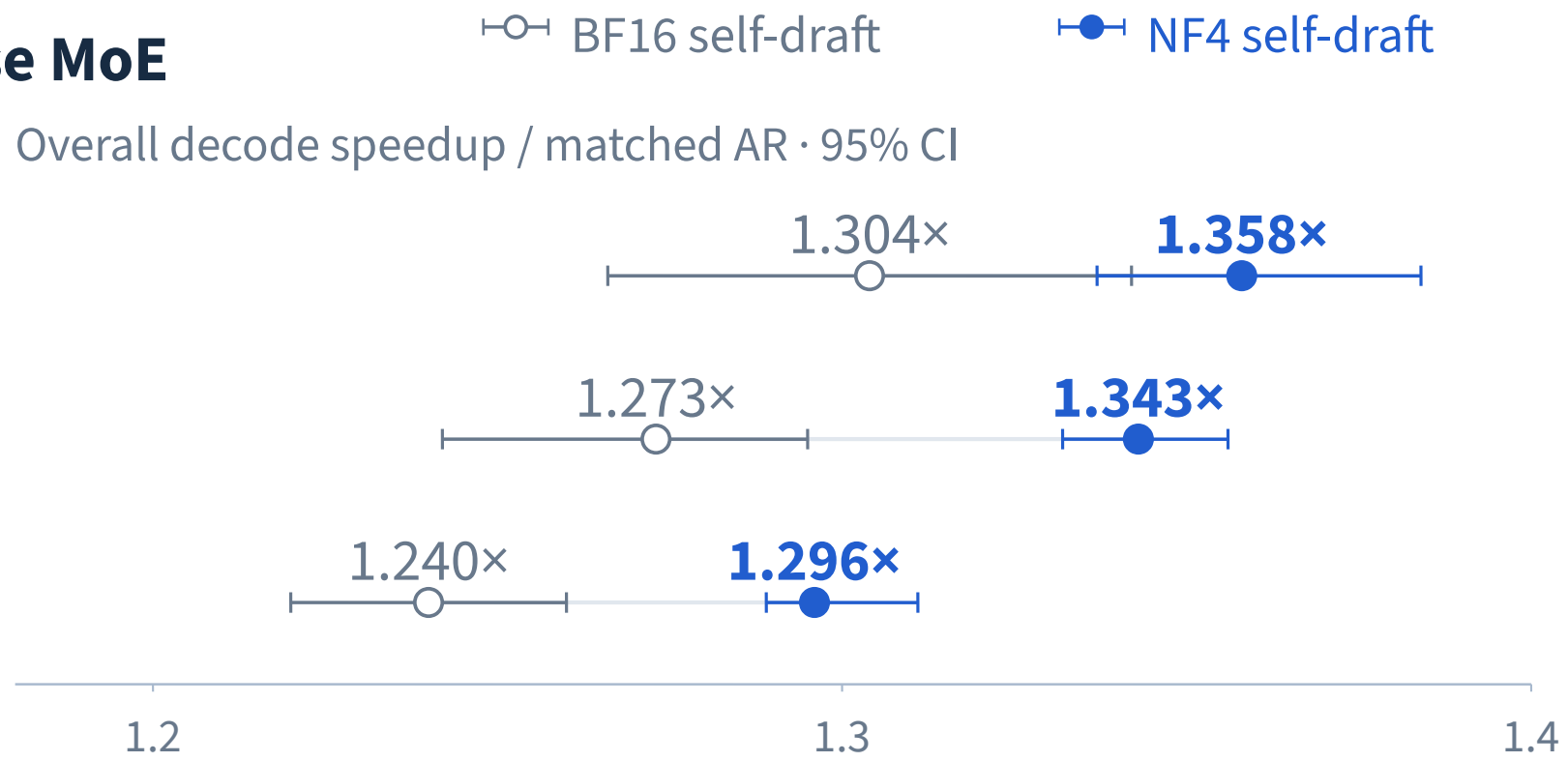


Online adaptation accelerates both single-request and batched decoding.

Speedups extend to dense GQA and sparse MoE.

CNN/DM · GSM8K · TinyStories · full BF16 verification

Dense GQA and sparse MoE



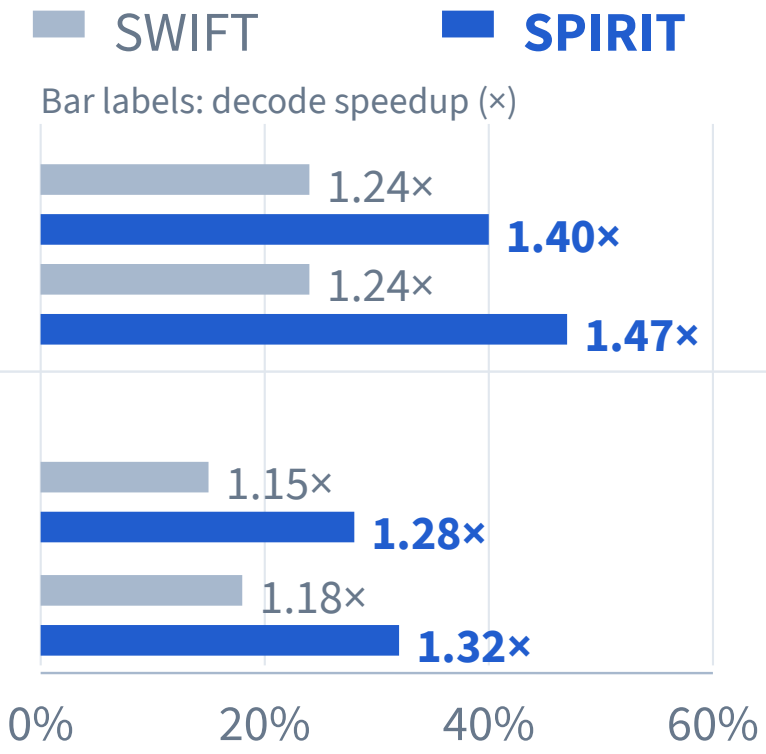
Mixtral: top-1 routing for drafts; full top-2 routing for verification.

Paper Table 6 / §4.5 · Aggregate decode speedup over each model's matched AR

SPIRIT accelerates code generation.

CodeLLaMA · HumanEval · greedy decoding and sampling at temperature 0.6

Setting	Throughput gain over AR (%)	Pass rate (%)		
		AR	SWIFT	SPIRIT
Greedy · pass@1				
13B	1.24× / **1.40×**	34.1	34.1	**34.1**
34B	1.24× / **1.47×**	48.1	48.1	**48.1**
Sampled · pass@10				
13B	1.15× / **1.28×**	70.1	62.1	**63.4**
34B	1.18× / **1.32×**	74.3	77.4	**75.6**



Greedy pass@1 matches AR; sampled pass@10 is a finite-run estimate.

Paper Table 2 / §4.1 · Overall decode speedup over matched AR

Each core mechanism contributes to speedup.

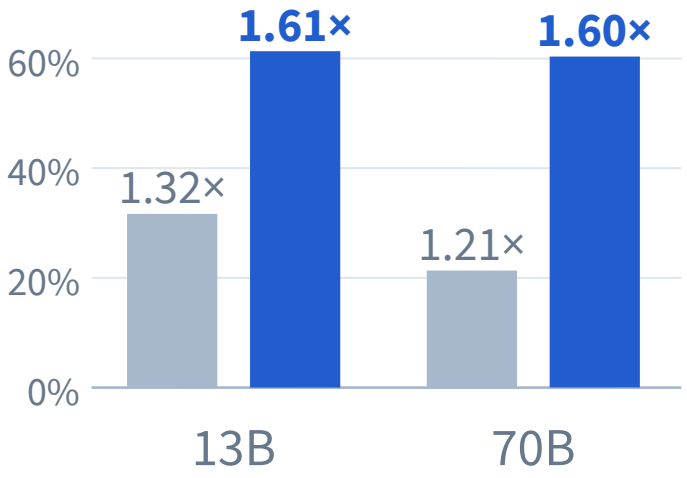
LLaMA-2 · FP16 · throughput gain over AR (%) · one intervention at a time

Full SPIRIT Ablated Bar labels: speedup (×)

Throughput objective

Acceptance-only scoring

Agreement alone is not the ideal objective.

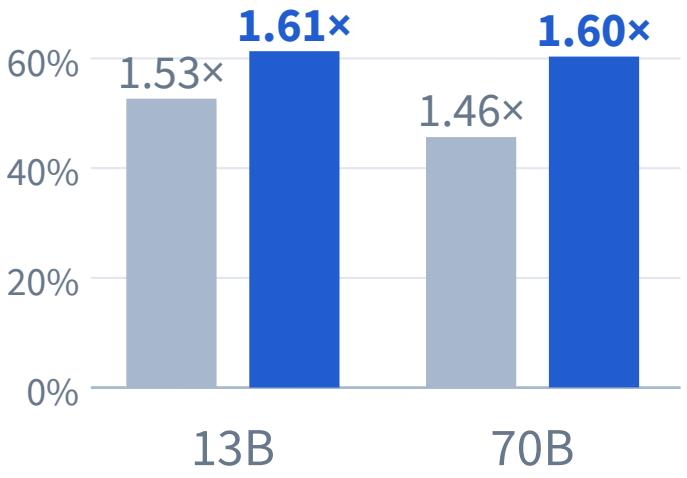


Mean over three stationary tasks

Ratio search

Fix r = 0.5; keep pattern tuning

The ratio matters even with pattern tuning.

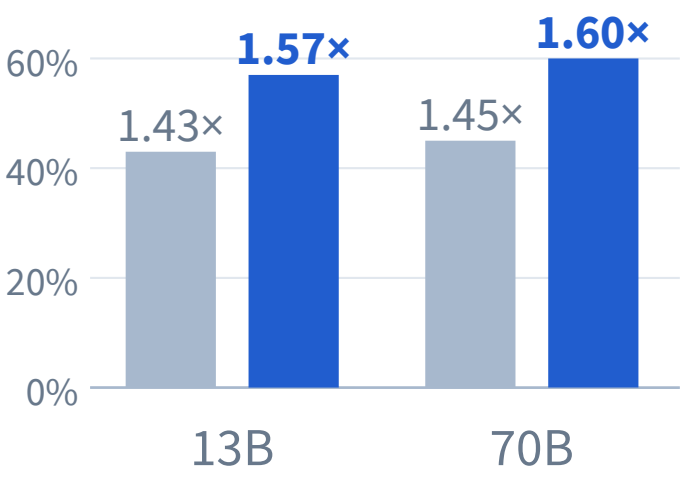


Mean over three stationary tasks

Profile reuse

Disable profile reuse

Reuse adds gains when workloads return.

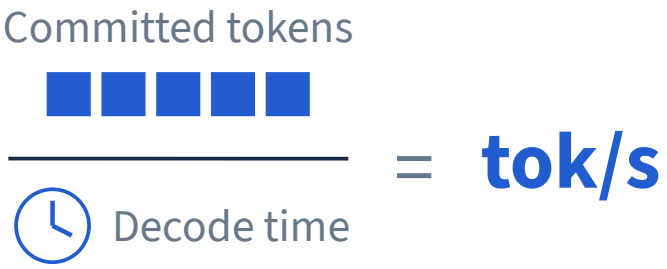


Full recurring workload stream

Productive decoding drives online adaptation.

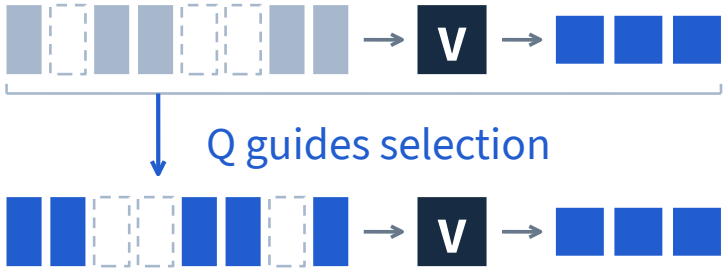
The same execution produces output and evidence for what comes next.

Realized throughput



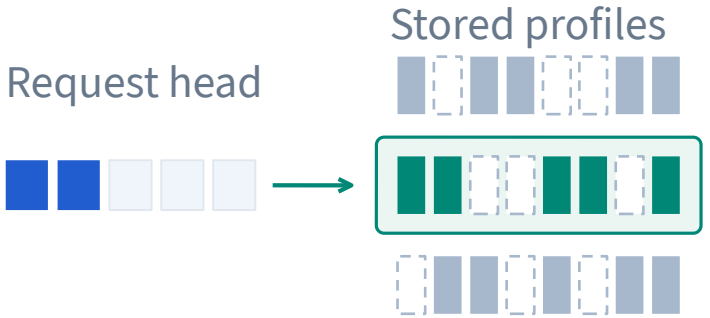
Captures agreement and intrinsic speed advantage.

Optimize while decoding



Length-calibrated feedback. Ratio, then layer pattern.

Adapt across requests



Detect shifts; reuse, refine, or restart.

UP TO **1.74×** **FP16 self-drafts**
Model–task evaluation

2.48× **NF4 self-drafts**
1,000-request stream

Decode speedup over AR · maxima from separate settings

No retraining · No offline profiling · Target distribution preserved