BRIDGE: Block-Wise Speculative Coordination for Cloud-Edge Retrieval-Augmented Generation

Yuting Li*
Tianjin University
Tianjin, China
1023244051@tju.edu.cn

Shaoyuan Huang*
Tianjin University
Tianjin, China
hsy_23@tju.edu.cn

Xiangqi Liu
Tianjin University
Tianjin, China
lxq8294@tju.edu.cn

Yunfeng Zhao†
Tianjin University
Tianjin, China
yfzhao97@tju.edu.cn

Xiaofei Wang
Tianjin University
Tianjin, China
xiaofeiwang@tju.edu.cn

Abstract

Retrieval-augmented generation (RAG) improves factuality by conditioning LLMs on retrieved evidence, yet real-world knowledge is often split across tiers: cloud-based RAG can exploit large public corpora, whereas edge-based RAG is the natural place to access private, user-specific stores. This raises a key question: how can one jointly leverage cloud and edge knowledge without transferring sensitive edge data or incurring prohibitive key-value (KV) recomputation. A direct output-level aggregation requires per-token synchronization, causing severe stalls under heterogeneous decoding speeds, and token-wise speculation still suffers from frequent communication and rollback overhead, limiting efficiency and usability. To address these challenges, we present BRIDGE, a Block-wise speculative RAG framework for Inter-database Distributed GEneration. BRIDGE enables parallel cloud–edge retrieval and drafting, without exposing private data or recomputing KV states. It introduces block-wise transmission and verification to amortize communication cost and improve acceptance efficiency, thereby reducing rollback waste. BRIDGE further employs an adaptive block-length strategy to balance the benefits of larger blocks against their verification overhead. Across public-private RAG benchmarks, model families, and network regimes, BRIDGE significantly improves question-answer relevance and personalization, while reducing end-to-end latency by up to 79.1%.

CCS Concepts

• **Computing methodologies** → **Natural language processing**;
• **Security and privacy** → **Privacy-preserving protocols**.

Keywords

Retrieval-Augmented Generation; Large Language Models; Speculative Decoding; Cloud-Edge Collaboration

*Both authors contributed equally to this research.
†Corresponding author.

KDD '26, Jeju Island, Republic of Korea

ACM ISBN 979-8-4007-2259-2/2026/08
https://doi.org/10.1145/3770855.3818064

ACM Reference Format:
Yuting Li, Shaoyuan Huang, Xiangqi Liu, Yunfeng Zhao, and Xiaofei Wang. 2026. BRIDGE: Block-Wise Speculative Coordination for Cloud-Edge Retrieval-Augmented Generation. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '26), August 09–13, 2026, Jeju Island, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3770855.3818064

Resource Availability:
The source code of this paper has been made publicly available at https://doi.org/10.5281/zenodo.20474967.

1 Introduction

Retrieval-augmented generation (RAG) improves large language models (LLMs) by conditioning generation on relevant content retrieved from external corpora, enhancing factuality, domain coverage, and robustness without costly fine-tuning [10, 17, 23].

In practice, however, the effectiveness of RAG critically depends on knowledge accessibility under deployment constraints [26]. As illustrated in Fig. 1, **cloud-based LLMs with RAG** are particularly effective at leveraging large-scale public or enterprise corpora (e.g., POI, technical, or clinical resources) to strengthen domain capabilities [5, 18, 21], yet it typically lacks access to user-private data due to privacy requirements [32], limiting personalization. Conversely, **edge-based small language models (SLMs) with RAG** can retrieve from private on-device stores (e.g., personal records, confidential enterprise files, or patient data) to enable personalized responses [12, 19, 24, 25, 29], but are constrained by limited model capacity and local knowledge coverage, making them less effective for complex queries requiring broad background knowledge.

These two paradigms reveal a fundamental gap in real-world deployments: modern applications increasingly require *both* broad, general knowledge from cloud-scale corpora and private, personalized knowledge from local stores, while simultaneously requiring strong usability guarantees such as low end-to-end generation latency under strict service-level objectives (SLOs). This motivates the following key question: *Can we design a new RAG paradigm that jointly integrates cloud-scale and on-edge knowledge to improve generation quality, while remaining usable under stringent latency SLO constraints?* We show that the answer is affirmative, but achieving this goal requires addressing several intertwined challenges. **Challenge 1: Jointly utilizing cloud and edge knowledge.** An

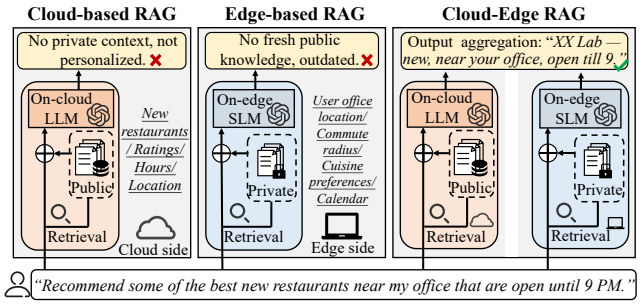


Figure 1: Comparison between different RAG architectures.

intuitive solution is to directly aggregate cloud and edge data into a unified knowledge space for retrieval and generation. However, such a design is impractical in real-world applications. Uploading edge-side private data to the cloud violates privacy and compliance constraints. Conversely, pushing cloud documents to the edge incurs high transmission latency and expensive KV recomputation from raw text. A more viable approach is output-level aggregation, where cloud and edge retrieve and generate locally and then aggregate their outputs. However, due to the auto-regressive nature of language models, such token-wise aggregation enforces tight synchronization: at each decoding step, the faster side must stall and wait for the slower side to finish generating and get aggregation feedback, substantially amplifying end-to-end latency.

While distributed speculative decoding offers a promising alternative for overlapping continuous generation with communication via a *draft-then-verify* paradigm [8, 28, 31], the vanilla speculative generation still proceeds on a token-wise basis, which introduces two additional bottlenecks that limit system effectiveness.

Challenge 2: Transmission bottleneck under frequent aggregation. The output aggregation requires frequent exchange of data packets between the cloud and device at every token generation step. Such fine-grained interactions result in many small data transfers whose latency is highly sensitive to network condition such as round-trip time (RTT). Under variable networks, RTT fluctuations repeatedly delay the completion of aggregation, prolonging per-output token processing time and ultimately inflating end-to-end latency, which leads to frequent SLO violations.

Challenge 3: Computation bottleneck from speculative rollback. The draft-then-verify paradigm incurs a fundamental recomputation cost. When verification rejects a drafted token, the system must rollback decoding states and discard the corresponding KV cache, forcing recomputation and wasting substantial compute. This overhead becomes severe when the cloud and edge exhibit low agreement in their token distributions, which increases rejection frequency and degrades throughput.

To address these challenges, we propose **BRIDGE**, a Block-wise speculative RAG framework for Inter-database Distributed GEneration. BRIDGE enables the cloud and edge to retrieve from their respective databases and generate candidate drafts in parallel, whose outputs are subsequently aggregated to jointly leverage public and private knowledge without transferring sensitive edge data or recomputing KV states. To avoid synchronization stalls, BRIDGE decouples output aggregation from step-by-step decoding via dual-side speculative coordination, enabling continuous generation and overlapping computation with communication.

To overcome efficiency bottlenecks, BRIDGE introduces a novel *block-wise* perspective for cloud-edge speculation. Instead of token-by-token aggregation, BRIDGE operates on token blocks drafted by both sides and performs *block-wise transmission and verification*. This design reduces synchronization frequency over multiple tokens, amortizing transmission latency and mitigating RTT amplification. More importantly, block-wise verification enables BRIDGE to evaluate drafts from a global perspective and commits the longest prefix consistent with the target distribution, improving acceptance efficiency and reducing rollback-induced recomputation.

Furthermore, BRIDGE employs an *adaptive block-length strategy* to balance the benefits and overhead of block-wise speculation. While longer blocks better amortize communication cost, verification cost grows linearly with block length and the expected accepted prefix cannot increase indefinitely. Accordingly, BRIDGE adapts the block length to network conditions and observed acceptance behavior, trading off communication amortization against verification cost to sustain low latency and robust throughput.

Our contributions can be summarized as follows:

- We propose BRIDGE, a novel cloud-edge collaborative RAG framework that jointly leverages public cloud and private edge knowledge to improve generation quality with high efficiency.
- We design a block-wise speculative coordination mechanism that decouples synchronous aggregation from sequential decoding. By employing block-wise transmission and verification, it significantly reduces communication cost while providing theoretical completeness for achieving higher acceptance rates.
- We develop an adaptive block-length strategy that optimizes the trade-off between communication amortization and verification cost, enabling robust under varying network conditions.
- Extensive experiments show that BRIDGE significantly improves generation quality over baselines while reducing end-to-end latency across diverse benchmarks and network conditions.

2 Preliminaries

2.1 Multi-Document Aggregation

RAG conditions LLM generation on retrieved documents to incorporate up-to-date, long-tail knowledge beyond parametric memory. Given a prefix $x_{<M} = \{x_0, \ldots, x_{M-1}\}$, an autoregressive model samples $x_t \sim \mathcal{P}(x_t \mid x_{<t})$ for $t \geq M$.

Context aggregation (concatenate-then-generate). A standard approach concatenates the top-k retrieved documents into a single context $\mathrm{C}_t = \textsc{Concat}(d_1, \ldots, d_k)$ and prepends (or inserts) it into the prompt, yielding $\mathcal{P}(x_t \mid x_{<t}, \mathrm{C}_t)$.

Output aggregation (generate-then-aggregate). Alternatively, output aggregation conditions on each document separately and interpolates the resulting next-token distributions [1, 9]. For step t, output aggregation follows the *Law of Total Probability* as:

$$\mathcal{P}(x_t \mid x_{<t}) = \sum_{d \in D_t} \omega_t(d) \cdot \mathcal{P}(x_t \mid d, x_{<t}), \tag{1}$$

where $D_t = \{d_1, \ldots, d_k\}$ is the top-k set retrieved from corpus D by a relevance score $R(d, x_{<t})$, and $\omega_t(d)$ denotes interpolation weights for document d. Compared with context aggregation, Eq. (1) enables parallel per-document generation and aggregates distributions at each decoding step.

2.2 Target Distribution for Cloud-Edge RAG

In a cloud-edge collaboration, the corpus is partitioned into a private on-device store $\mathcal{D}^{\text{edge}}$ and a public cloud store $\mathcal{D}^{\text{cloud}}$. At step t, each side $s \in \{\text{edge}, \text{cloud}\}$ retrieves $D_t^s \subseteq \mathcal{D}^s$ and computes per-document next-token distributions in parallel, yielding $\mathcal{P}^s(x_t \mid d, x_{<t})_{d \in D_t^s}$. Let $D_t = D_t^{\text{edge}} \cup D_t^{\text{cloud}}$ and stack all per-document distributions as $\mathcal{P} = \left[\mathcal{P}^{\text{edge}}, \mathcal{P}^{\text{cloud}}\right]^\top$. The target next-token distribution induced by output aggregation is:

$$x_t \sim \omega_t^\top \mathcal{P} = \sum_{d \in D_t} \omega_t(d) \cdot \mathcal{P}(x_t \mid d, x_{<t}), \tag{2}$$

where $\omega_t(d)$ is typically a softmax over relevance scores:

$$\omega_t(d) = \frac{\exp\big(R(d, x_{<t})\big)}{\sum_{d' \in D_t} \exp\big(R(d', x_{<t})\big)}. \tag{3}$$

For $s \in \{\text{edge}, \text{cloud}\}$, define $h_t^s \triangleq \sum_{d \in D_t^s} \exp(R(d, x_{<t}))$ and mixing coefficients $\eta_t^s = \frac{h_t^s}{h_t^{\text{edge}} + h_t^{\text{cloud}}}$. Eq. (2) can be written as a two-source interpolation:

$$\mathcal{P}(x_t \mid x_{<t}) = \eta_t^{\text{edge}} \mathcal{P}^{\text{edge}} + \eta_t^{\text{cloud}} \mathcal{P}^{\text{cloud}}. \tag{4}$$

Eq. (4) highlights that the next-token distribution is jointly determined by private edge and public cloud retrieval. In practical implementations, the intra-side aggregation on either the cloud or the edge can be simplified using context aggregation to avoid repeated per-document generation, while the cross-side collaboration still follows the output-aggregation form above.

Despite its effectiveness, this paradigm incurs substantial latency from frequent cross-side synchronization, which leads to idle waiting under heterogeneous decoding speeds and makes latency highly sensitive to network RTT. These per-token costs accumulate over long generations, significantly increasing end-to-end latency.

3 BRIDGE

To jointly leverage cloud-scale general knowledge and on-edge private knowledge while maintaining service usability, we propose BRIDGE, a block-wise speculative coordination framework for cloud-edge RAG. BRIDGE enables both sides to draft tokens in parallel and performs output aggregation, thereby reducing the inherent synchronization stalls (§3.1.1). Specifically, it exchanges draft tokens and auxiliary statistics in blocks to amortize RTT-dominated communication cost (§3.1.2), verifies each block to commit the longest acceptable prefix and reduce rollback waste (§3.2), and adaptively adjusts the block length to balance communication amortization against verification overhead (§3.3).

As shown in Fig. 2, BRIDGE follows a draft-then-verify workflow with four steps: Upon receiving a request, the edge and cloud **Decoders** independently draft tokens using their locally retrieved documents (❶). Draft tokens and auxiliary statistics are buffered into **Draft Queues** and sent to the edge-side[1] **Coordinator** in blocks, reducing per-token synchronization (❷). The **Coordinator** jointly verifies the received draft blocks and determines the next-iteration block length using **Profiler** signals (❸). The aggregation output and updated block length are broadcast to both sides (❹).

[1]Since the cloud typically decodes faster and can stream drafts without waiting, while downlink bandwidth is often better than uplink, reducing coordination delay.

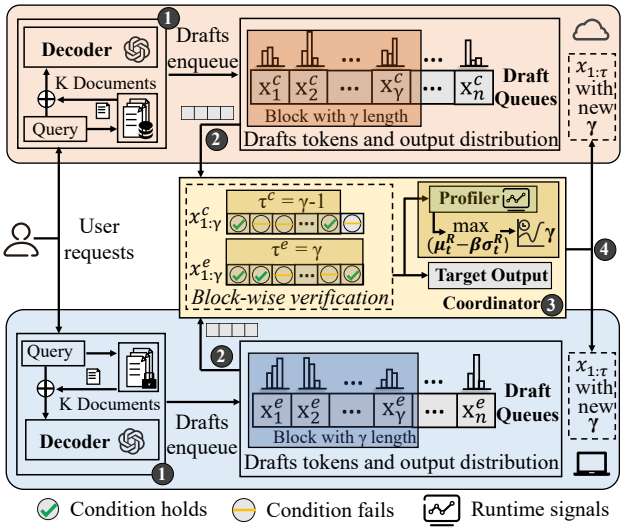


Figure 2: Overview of the BRIDGE framework.

3.1 Speculative Coordination Framework

3.1.1 Event-driven Coordinator. The core of BRIDGE is an event-driven coordinator, which collects draft blocks from both sides, constructs the aggregated target distribution (Eq. (4)), and produces the committed outputs. To ensure that the speculative process without introducing additional sampling bias, the coordinator must be distribution-preserving. Formally, we define:

Definition 1 (Distribution-preserving.) Given the coordinator Coord takes the draft blocks x^s, distributions along the sample path $\mathcal{P}^s$ and weights h^s as input, and outputs a target x. Coord is said to be distribution-preserving if for any context c, x^s and $\mathcal{P}^s$, the outputs of Coord satisfy:

$$\textsc{Coord}\big(c, x^s, \mathcal{P}^s, h^s\big) \sim \mathcal{P}(\cdot \mid c), \tag{5}$$

where $\mathcal{P}(\cdot \mid c)$ denotes the target distribution in Eq. (4).

In particular, the coordinator operates as an event-driven protocol that alternates between collecting drafts and committing outputs. Specifically, it reacts to two events: *the arrival of draft blocks from both sides* and *the completion of a global commit finalization*:

- For an arrival event, for $s \in \{\text{edge}, \text{cloud}\}$, the coordinator dequeues the incoming draft block $x_{1:\gamma}^s$ from the local draft queue Q_{draft}^s and retrieves the cached along-path auxiliary statistics, including the conditional distributions $\mathcal{P}^s$ and the associated weights h^s. Subsequently, it computes the interpolation weights η^s, performs the block-wise verification BlockVerify to obtain the committed prefix, and samples one verified prefix $\tilde{x}_{1:\tau}^s$ with uniform probability as the committed output $x_{1:\tau}$.
- For a completion event, the coordinator checks both sides' verification results. If a rejection occurs, it sends the committed output so that they can update queues and rollback consistently. In addition, the block length γ is updated to adapt runtime changes.

Algorithm 1 summarizes the event-driven workflow. In this procedure, BRIDGE achieves efficient coordination through two mechanisms: **Block-wise Transmission** and **Block-wise Verification**.

3.1.2 Block-wise Transmission. BRIDGE adopts a block-wise transmission scheme: each side buffers draft tokens and lightweight

Algorithm 1: Event-Driven Speculative Coordinator

Input: Draft tokens x_t^s, distributions $\mathcal{P}^s$, and local weights h_t^s, for $s \in \{\text{edge}, \text{cloud}\}$. Draft length γ.

Event: Both draft queues Q^s_{draft} are ready:
for $s \in \{edge, cloud\}$ **do**
  Obtain the head block $x^s_{1:\gamma}$ from Q^s_{draft}, retrieve cached distributions $\{\mathcal{P}_s(\cdot \mid c, x^s_{1:i})\}_{i=0}^{\gamma-1}$ and weights $h^s_{1:\gamma}$;
  for $i = 1, \ldots, \gamma$ **do**
    $\eta^s_i \leftarrow h^s_{1:i} / (h^{edge}_{1:i} + h^{cloud}_{1:i})$;
  $\tilde{x}^s_{1:\tau} \leftarrow \textsc{BlockVerify}(x^s_{1:\gamma}, \mathcal{P}^s_{1:\gamma}, \eta^s_{1:\gamma})$;
Sampling $x_{1:\tau} \leftarrow \mathbf{1}_{\sigma \le 0.5} \cdot \tilde{x}^{\text{edge}}_{1:\tau} + \mathbf{1}_{\sigma > 0.5} \cdot \tilde{x}^{\text{cloud}}_{1:\tau}$, $\sigma \sim U(0,1)$;
Event: On completion of a block verification:
for $s \in \{edge, cloud\}$ **do**
  if $x^s_{1:\gamma} \neq x_{1:\tau}$ **then**
    broadcast the committed prefix $x_{1:\tau}$;
Update γ by profiling signals.

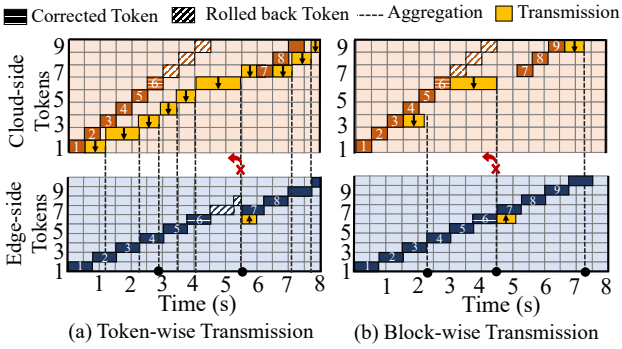
auxiliary statistics into blocks before sending to the coordinator. Fig. 3 illustrates the superiority of this scheme with a timeline comparison (block length $\gamma = 3$).

In the typical case where the first three tokens are fully accepted, block-wise transmission commits the target prefix earlier (i.e., 2.2s versus 2.9s) by reducing cross-side waiting compared to token-wise transmission, which exhibits pronounced delays under RTT-dominated links. This benefit persists during subsequent aggregation when a rejection occurs (e.g., at token 6): with fewer synchronization points, block-wise transmission reduces communication-induced blocking, mitigating redundant edge-side computation and unnecessary rollbacks. Consequently, under the same wall-clock budget, it achieves a longer committed output (e.g., 9 tokens within 7.3 seconds). Notably, as both sides draft tokens sufficiently fast compared to RTT, block-wise transmission does not introduce noticeable idle time for waiting to fill a block in practice. Moreover, although block-wise coordination commits outputs per block, it does not meaningfully increase instantaneous per-token latency. With a short block window and higher acceptance efficiency, the amortized coordination overhead becomes negligible and can even be lower than token-wise speculation.

The key next step is to decide which part of a draft block can be committed without altering the target distribution. Next, we detail our block-wise verification mechanism, which is distribution-preserving by construction (Definition 1) and improves acceptance efficiency by jointly verifying a block.

3.2 Block-wise Verification

Instead of the standard token-wise verification that stops at the first rejection, we introduce a block-wise verification to verify an entire draft block jointly and commit the *longest acceptable prefix* in one shot. Crucially, this aggressive reuse must not bias decoding, therefore we design BlockVerify and prove its optimality under the constraint of distribution-preserving.



(a) Token-wise Transmission (b) Block-wise Transmission

Figure 3: Speculative coordination pipeline: token-wise vs block-wise transmission (reject at Token 6). The edge and cloud decoding time are 0.75s and 0.5s, respectively.

3.2.1 Design of Block-wise Verification. For $s \in \{\text{edge}, \text{cloud}\}$, BlockVerify takes as input two draft blocks $x^s_{1:\gamma}$ proposed by both sides, together with the cached distributions $\{\mathcal{P}^s(\cdot \mid c, x^s_{1:i})\}_{i=0}^{\gamma-1}$ and the corresponding target distributions $\{\mathcal{P}(\cdot \mid c, x_{1:i})\}_{i=0}^{\gamma-1}$. Its output is the committed prefix length τ and the committed tokens $x_{1:\tau}$. To make block-level decisions without introducing bias, the design revolves around a *prefix survival probability* $p(x_{1:i})$ that characterizes how likely a draft prefix can be safely committed.

Prefix survival probability. Given a realized draft prefix $x_{1:i}$, define the cumulative survival probability recursively as

$$p(x_{1:i}) \;=\; \min\!\left(p(x_{1:i-1}) \cdot \frac{\mathcal{P}(x_{1:i} \mid c, x_{1:i-1})}{\mathcal{P}^s(x_{1:i} \mid c, x_{1:i-1})}, 1\right), \tag{6}$$

with $p(x_0) = 1$. Intuitively, $p(x_{1:i})$ accumulates the acceptance budget along the draft block and explicitly captures the dependency across prefix decisions, which is the key difference from token-by-token verification.

Lemma 1. *Let $X_{1:\gamma} \sim \mathcal{P}^s_{1:\gamma}(\cdot \mid c)$ be a draft block, and let $(X_{1:\tau}, \tau) = \textsc{BlockVerify}(X_{1:\gamma}, \{\mathcal{P}^s(\cdot \mid c, X_{1:i})\}_{i=0}^{\gamma-1}, \{\mathcal{P}(\cdot \mid c, X_{1:i})\}_{i=0}^{\gamma-1})$. Then for any $i \le \gamma$ and any realized prefix $x_{1:i}$,*

$$\Pr(\tau \ge i \mid X_{1:i} = x_{1:i}) = p(x_{1:i}). \tag{7}$$

The proof is deferred to Appendix B.1. Lemma 1 indicates that achieving the desired survival probabilities $p(x_{1:i})$ is sufficient for realizing the correct block-wise acceptance. Based on this, BlockVerify uses two coupled rules to determine the committed prefix while ensuring optimality-guarantee and distribution-preserving.

Rule 1: block-wise acceptance. BlockVerify commits a prefix rather than individual tokens. Concretely, it constructs a sequence of acceptance probabilities $\{h(x_{1:i})\}_{i=1}^{\gamma}$ that will be used to sample the accepted length. At runtime, BlockVerify draws i.i.d. random variables $\lambda_i \sim \text{Unif}(0,1)$ and obtains the longest *accepted* prefix:

$$\tau' \;=\; \max\{\, i \in [\gamma] : \lambda_j \le h(x_{1:j}) \text{ for all } j \le i \,\}. \tag{8}$$

The remaining question is how to set $h(x_{1:i})$ so that the resulting τ' satisfies Lemma 1 while preserving the target distribution.

Deriving $h(x_{1:i})$ via shortfall mass. If the verification stops at prefix $x_{1:i}$, the next token should follow the target conditional distribution $\mathcal{P}(\cdot \mid c, x_{1:i})$. However, the draft side has already spent

Algorithm 2: BlockVerify: Block-wise Verification

Input: For $s \in \{\text{edge}, \text{cloud}\}$, draft block $x^s_{1:\gamma}$; per-step distributions $\{\mathcal{P}^s(\cdot \mid c, x_i)\}_{i=0}^{\gamma-1}$; mixture weights $\eta^s_{1:\gamma}$.

for $i \leftarrow 0$ **to** $\gamma - 1$ **do**
  Calculate target distribution $\mathcal{P}(\cdot \mid c, x_i)$ as Eq. (4)
Set $\tau \leftarrow 0$, $p(x_0) \leftarrow 1$; draw i.i.d. $\{\lambda_i\}_{i=1}^{\gamma} \sim \text{Unif}(0,1)$;
for $s \in \{edge, cloud\}$ **do**
  for $i \leftarrow 1$ **to** γ **do**
    Set cumulative probability $p(x_{1:i})$ as Eq. (6);
    Calculate acceptance probability $h(x_{1:i})$ as Eq. (10);
    if $\lambda_i \le h(x_{1:i})$ **then**
      Set $\tau'_s \leftarrow i$;
$\tau \leftarrow \min\{\tau'_{\text{edge}}, \tau'_{\text{cloud}}\}$;
if $\tau < \gamma$ **then**
  for $s \in \{edge, cloud\}$ **do**
    Sample $Y \sim p_{\text{res}}(\cdot \mid c, x_{1:\tau})$ as in Eq. (11)
    $x^s_{\tau+1} \leftarrow Y$, $\tau \leftarrow \tau + 1$;
return $(x^s_{1:\tau})$ *for both sides.*

probability mass according to $\mathcal{P}^s(\cdot \mid c, x_{1:i})$. We quantify the remaining mass needed for each candidate token $x' \in \mathcal{X}$ using the pointwise shortfall

$$d_i(x') \;=\; \Big[p(x_{1:i})\,\mathcal{P}(x' \mid c, x_{1:i}) - \mathcal{P}^s(x' \mid c, x_{1:i})\Big]_+, \qquad (9)$$

and aggregate it as $D_i = \sum_{x' \in \mathcal{X}} d_i(x')$. We then define the acceptance probability by normalizing the shortfall with the total available mass at step i (shortfall plus the remaining slack $1 - p(x_{1:i})$):

$$h(x_{1:i}) \;=\; \frac{D_i}{D_i + 1 - p(x_{1:i})}\,. \qquad (10)$$

This choice couples the acceptance rule with the survival probability $p(x_{1:i})$, enabling block-level commitment while keeping the overall sampling budget consistent.

Rule 2: residual correction at the boundary. We set the committed length $\tau = \tau' + \mathbb{1}[\tau' < \gamma]$. If $\tau' = \gamma$, the whole block is accepted and no correction is needed. Otherwise ($\tau' < \gamma$), the remaining suffix is generated in a distribution-preserving manner. To avoid sampling bias at the boundary, BlockVerify draws a correction token Y from a residual distribution that reallocates the unaccounted probability mass. Given the accepted prefix $x_{1:\tau'}$, the draft contributes mass $\mathcal{P}^s(\cdot \mid c, x_{1:\tau'})$, while the target requires $p(x_{1:\tau'})\,\mathcal{P}(\cdot \mid c, x_{1:\tau'})$. Using the shortfall in Eq. (9), the residual distribution is

$$p_{\text{res}}(x' \mid c, x_{1:\tau'}) = \frac{d_{\tau'}(x') + (1 - p(x_{1:\tau'}))\,\mathcal{P}(x' \mid c, x_{1:\tau'})}{D_{\tau'} + 1 - p(x_{1:\tau'})}, \quad (11)$$

where $D_{\tau'} = \sum_{x' \in \mathcal{X}} d_{\tau'}(x')$. The denominator normalizes the distribution since $\sum_{x'} d_{\tau'}(x') = D_{\tau'}$ and $\sum_{x'} \mathcal{P}(x' \mid c, x_{1:\tau'}) = 1$.

3.2.2 Block-wise Optimality under Distribution Preservation. We provide two theoretical guarantees for BlockVerify. First, BlockVerify is *distribution-preserving*:

Theorem 1. *$\forall c$, the committed outputs of BlockVerify are distributed identically to the aggregated target distribution $\mathcal{P}(\cdot \mid c)$.*

Moreover, under this distribution-preserving constraint, BlockVerify is *optimal* in expected efficiency:

Theorem 2. *For $i > 0$, let $N(i)$ denote the total number of* committed *tokens after i iterations of Algorithm 1. For any distribution-preserving verification (i.e., any algorithms satisfying Theorem 1), we have $\forall c, \mathcal{P}, \mathcal{P}^s, \gamma$ and i, the block-wise verification BlockVerify maximizes $\mathbb{E}[N(i)]$. Specifically, it is optimal for token-wise verification:*

$$\mathbb{E}_{\text{BlockVerify}}[N(i)] \;\ge\; \mathbb{E}_{\text{TokenVerify}}[N(i)]\,. \qquad (12)$$

Therefore, BlockVerify maximizes the expected number of committed tokens under the distribution-preserving constraint. The full proofs of Theorem 1 and Theorem 2 are provided in Appendices B.2 and B.3, respectively.

3.3 Adaptive Block-length Strategy

Larger blocks amortize communication by reducing synchronization frequency, but they also increase verification overhead, user-perceived latency, and rollback waste once acceptance gains saturate. BRIDGE therefore adapts the block length γ to maximize effective throughput under latency and acceptance constraints.

3.3.1 Profiling. The adaptive block-length strategy is driven by a lightweight profiler that exposes both cost and benefit signals to the coordinator: (i) drafting cost on each side, (ii) communication and verification cost for aggregation progress, and (iii) committed progress from recent block verification outcomes. Specifically, we use per-iteration latency to estimate (i) and (ii) , and use acceptance statistics to estimate (iii). In practice, the profiler is low-overhead and updated per iteration. It is initialized with offline estimates and refined online using timestamps already available in the protocol, together with simple smoothing to reduce noise. Implementation details and calibration procedures are deferred to Appendix A.

3.3.2 Adaptive Block-length Strategy. BRIDGE selects the block length γ_t to maximize aggregation throughput, defined as the committed progress per unit wall-clock time. Let ΔT_t denote the wall-clock duration of iteration t, BlockVerify commits a prefix of length τ_t. We define the throughput as $R_t = \tau_t / \Delta T_t$.

Consequently, we formulate block-length selection as a constrained contextual bandit [20]. At iteration t, given the profiled runtime context z_t and candidate set $\Gamma == \{1, \ldots, \gamma_{\max}\}$, the Coordinator selects γ_t to maximize cumulative throughput while bounding per-iteration latency and maintaining sufficient acceptance:

$$\max_{\{\gamma_t\}_{t=1}^{T}} \sum_{t=1}^{T} R_t(z_t, \gamma_t) \qquad \text{s.t.} \quad L_t(z_t, \gamma_t) \le L_{\max}, \quad F_t(z_t, \gamma_t) \ge \alpha_{\min}, \forall t. \qquad (13)$$

where L_t denotes the per-iteration latency and $F_t = \tau_t / \gamma_t$ denotes the acceptance ratio.

Safe Online Bayesian Optimization. We instantiate the bandit with a Gaussian Processes (GP)-based safe Bayesian optimization policy [27]. At iteration t, the GP posteriors yield (μ^R_t, σ^R_t), (μ^L_t, σ^L_t), and (μ^F_t, σ^F_t), corresponding to the GP-predicted mean and uncertainty for throughput, latency, and acceptance ratio, respectively.

Algorithm 3: Collaborative Gating SafeOBO Algorithm

Input: Control space Γ; safe seed set S_0; kernel k; $L_{\max}$ (maximum delay); $\alpha_{\min}$ (minimum accuracy); exploration parameter β

Initialization: $Z_0 = \emptyset$; $y_0^R = \emptyset$; $y_0^L = \emptyset$; $y_0^F = \emptyset$

Function *Execute* z_t, γ_t**:**
- Observe $F_t(z_t, \gamma_t)$ (accuracy), $L_t(z_t, \gamma_t)$ (response time), E_t (the net committed tokens), ΔT_t (the wall-clock);
- Compute $R_t(z_t, \gamma_t) \triangleq \frac{\tau_t + \mathbb{1}[\tau_t < \gamma_t]}{\Delta T_t}$
- Update GP posteriors:
  - $y_t^R \leftarrow y_{t-1}^R \cup R_t(z_t, \gamma_t)$ (update throughput)
  - $y_t^L \leftarrow y_{t-1}^L \cup L_t(z_t, \gamma_t)$ (update response time)
  - $y_t^F \leftarrow y_F \cup F_t(z_t, \gamma_t)$ (update accuracy)

for $t = 1, \ldots, T_0$ **do**
- Update the online profiler and observe context z_t;
- Randomly select $\gamma_t \in \Gamma$;

for $t = T_0 + 1, \ldots, T$ **do**
- Update the online profiler and observe context z_t;
- Compute $\mu_{t-1}^{(i)}(z_t, \gamma)$ and $\sigma_{t-1}^{(i)}(z_t, \gamma)$ for all $i \in \{R, L, F\}$ using GP posteriors;
- Estimate the safe set as Eq. (14)
- Select $\gamma_t = \arg\max_{\gamma \in \mathcal{S}_t} \left(\mu_t^R(z_t, \gamma) - \beta \sigma_t^R(z_t, \gamma)\right)$;

return γ_t.

In particular, we construct a *safe set* of block lengths that satisfies the constraints with high probability and restrict decisions to:

$$\mathcal{S}_t = \left\{\gamma \in \Gamma : \mu_t^L + \beta\, \sigma_t^L \le L_{\max},\ \ \mu_t^F - \beta\, \sigma_t^F \ge \alpha_{\min}\right\}, \tag{14}$$

where β controls the confidence level. We initialize with a conservative γ that is empirically safe to ensure $\mathcal{S}_t \neq \emptyset$, then select

$$\gamma_t = \arg\max_{\gamma \in \mathcal{S}_t} \mu_t^R(z_t, \gamma) - \beta\, \sigma_t^R(z_t, \gamma), \tag{15}$$

and update the GP posteriors after observing (R_t, L_t, F_t). We follow the standard GP-based safe BO practice[27] and Algorithm 3 presents the complete procedure.

4 Experiment

In this section, we conduct experiments to address the following research questions:

- **RQ1:** Does BRIDGE improve the overall generation quality while reducing end-to-end latency compared to baselines?
- **RQ2:** Can BRIDGE effectively integrate the cloud-side public and the device-side private data via output aggregation?
- **RQ3:** Does the proposed block-wise speculative coordination reduce communication cost and rollback overhead compared to traditional token-wise operation?
- **RQ4:** Does the adaptive block-length strategy achieve a better communication–cost trade-off, therefore improving throughput?

4.1 Experiment Settings

4.1.1 Dataset. We construct two public-private benchmarks in a cloud-edge RAG setting. Each benchmark combines (i) a cloud-side public corpus that provides general knowledge and (ii) an edge-side private dataset that provides user-specific information.

- **Co-Wiki:** For the public knowledge source on the cloud, we use WikiText [22], a Wikipedia-derived benchmark containing over 100M tokens from curated articles. For the edge-side private data, we use **CoGensis** [33], which contains privacy-sensitive user profiles and diverse personalized tasks. Each task involves long-text writing that requires integrating user-specific information.
- **Movie-Wiki:** The cloud-side public corpus is the same Wiki-Text setup as in Co-Wiki. For the edge-side private data, we use **Movie Explain** [34], a synthetic dataset for personalized movie recommendation explanations grounded in user profiles. Each instance provides a private user profile and requires the model to generate an explanation aligned with the user's preferences.

We use WikiText as a representative public corpus since it covers broad, high-quality factual and encyclopedic knowledge, e.g., entity and concept definitions for profile-grounded long-form tasks and public movie-related knowledge.

4.1.2 Implementation. We implement BRIDGE as a prototype system in Python, consisting of two symmetric runtimes deployed on the cloud and the edge, respectively. Core functionalities (e.g., decoding, verification, and transmission) are executed in parallel using eight threads, along with a memory-resident service process for document retrieval. We employ *Hugging Face Transformers* for LLM utilities, *LangChain* for document chunking, and *Faiss* [15] for indexing and similarity search of document embeddings.

Parameters Setting. Unless otherwise specified, we set the initial block length to γ=1 and cap the maximum block length by $\gamma_{\max}$=8. For safe online Bayesian optimization, we set β=2.5 and use an initial warm-up horizon of T_0=500 steps to collect profiling signals before fully enabling dynamic adaptation.

Testbed. We evaluate BRIDGE on a real hardware testbed with a cloud server with an NVIDIA A100 (80 GB) GPU and an edge laptop with an NVIDIA GeForce RTX 3090 GPU. The two machines communicate over a 2.4 GHz Wi-Fi LAN, whose baseline latency and jitter are measured by *sockperf*. To emulate diverse network conditions, we replay a predefined random latency trace by injecting controlled delay on the network interface controller (NIC) using Linux traffic control.

Communication. We implement data transmission over TCP/IP using Python *socket*, with fixed-length headers for message type and payload size. General objects are serialized with *pickle* and compressed by LZ4, while numeric vectors use *numpy.tobytes()* for lower overhead. For next-token distributions, we apply top-p truncation with p=0.8 and compactly encode token indices and probabilities using uint8 and float16, respectively.

Preemptible decoding. To promptly react to rejection signals during speculative generation, we make draft decoding preemptible. Concretely, we register a forward hook in each Transformer layer, and when a stop event occurs (e.g., the coordinator rejects the current draft block), the hook raises an exception to interrupt the ongoing forward pass. The generation caller catches the exception, rolls back the KV cache and attention masks to the latest committed prefix, and resumes decoding conditioned on the newest committed tokens to trigger re-computation.

Table 1: The performance comparison of BRIDGE, against baseline approaches on both datasets. The standard deviation within the test set indicates the consistency of performance, demonstrating the robustness of our method. ↑: higher is better.

Dataset	Method	Qwen Series		LLaMA Series	
		Q-A Rel.↑	Persona.↑	Q-A Rel.↑	Persona.↑
Co-Wiki	Edge-based RAG	5.640±0.028	5.931±1.122	6.813±2.866	6.198±1.093
	Cloud-based RAG	4.807±1.457	5.840±0.620	6.279±2.718	6.081±2.134
	EGDR	5.902±1.301	5.712±1.205	6.899±1.571	5.957±1.672
	DRAGON†	6.274±2.917	5.625±1.014	6.814±1.190	6.263±1.266
	BRIDGE	**6.592±0.056**	**6.179±0.475**	**7.220±0.693**	**6.421±1.108**
	CGDR‡	7.399±1.364	6.262±0.459	7.651±1.586	7.431±1.096
Movie-Wiki	Edge-based RAG	5.444±1.843	6.115±1.427	6.965±2.292	6.319±0.972
	Cloud-based RAG	6.957±0.744	5.051±1.832	6.991±1.936	6.081±1.548
	EGDR	5.941±0.508	4.963±0.894	6.373±1.426	3.756±1.918
	DRAGON†	5.748±1.602	6.156±1.197	5.987±2.425	6.362±0.166
	BRIDGE	**7.184±1.459**	**6.809±1.245**	**7.301±1.935**	**6.487±0.807**
	CGDR‡	7.867±0.931	7.456±0.687	8.715±1.323	7.935±0.592

† The results of VCR methods are omitted, as they produce identical outputs to DRAGON.
‡ CGDR represents the theoretical upper bound by directly leveraging private data on the cloud, which is infeasible in practical deployments.

4.1.3 Models and baselines. We evaluated our framework using two representative model series, **LLaMA** and **Qwen**. Specifically, for the LLaMA series , we use LLaMA-3.3-13B-Instruct as the cloud model and LLaMA-3.2-1B-Instruct as the edge model [6]. For the Qwen series, we use Qwen2.5-7B-Instruct as the cloud model and Qwen2.5-1.5B-Instruct as the edge model [7]. For all inferences, we set the temperature to 0.

We compare BRIDGE with several baseline methods: **Edge-based RAG (ER)**: the edge SLM performs retrieval-augmented generation over the edge-private corpus. **Cloud-based RAG (CR)**: the cloud LLM performs retrieval-augmented generation over the cloud-public corpus. **Edge Generation with Distributed Retrieval (EGDR)**: this baseline performs on-edge augmented generation with documents retrieved from a distributed corpus spanning both the device and the cloud. **Cloud Generation with Distributed Retrieval (CGDR)**: this baseline mirrors EGDR but generates on the cloud LLM after context aggregation over documents retrieved from the distributed corpus. **Vanilla Collaborative RAG (VCR)**: this baseline implements a straightforward cloud-edge RAG pipeline with distributed retrieval and output aggregation, which has synchronization stall as discussed in §2.2. **DRAGON [8]**: this baseline follows a state-of-the-art speculative RAG design that adopts output aggregation and reduces strict lock-step synchronization via a token-wise asynchronous pipelined protocol.

4.1.4 Evaluation Metrics. Following previous work [33, 34], we employ two LLM-as-a-judge [35] metrics, with an advanced LLM as the evaluator and provide the complete judge prompts and evaluation scripts in our open-source repository. Specifically, we quantify:

- **Q-A Relevance (Q-A Rel.)** measures how well the response addresses the user query in terms of semantic alignment and task completion.
- **Q-A Personalization (Persona.)** measures whether the response appropriately incorporates user-specific private context (e.g., preferences, profiles, history) when such context is available, producing more tailored outputs.

Furthermore, we report the average and P99 end-to-end latency to quantify serving efficiency under an SLO threshold of 20s [14].

4.2 Overall Performance (RQ1)

To evaluate the overall generation quality and efficiency of BRIDGE, we conduct end-to-end experiments on the Co-Wiki and Movie-Wiki datasets with other baselines under two LLM series.

4.2.1 Generation Quality. Table 1 reports **Q-A Rel.** and **Persona**. We highlight three key findings below.

BRIDGE consistently achieves the best generation quality. Across both datasets, BRIDGE attains the highest Q-A Rel. and Persona. scores among practical baselines. Compared with ER, it improves Q-A Rel. by up to 32.0%, confirming that cloud-side public knowledge and stronger model capacity remain critical for general task completion. Compared with CR, it improves Persona. by up to 34.8%, demonstrating the value of leveraging private on-device knowledge without exposing it to the cloud. BRIDGE also consistently outperforms DRAGON, suggesting that block-wise coordination yields more stable collaborative decoding and more reliable integration of cross-side knowledge.

Naive context aggregation can be ineffective. EGDR fails to improve quality as expected (e.g., on the Movie-Wiki dataset with LLaMA, it even reduces Q-A Rel. by 8.5% compared to ER), since edge SLMs struggle to reliably utilize long concatenated contexts, leading to distraction and context dilution as the prompt grows. In contrast, CGDR achieves the best absolute scores but requires uploading privacy-sensitive user data to the cloud, which is infeasible in practical deployments.

Generalizability across datasets and model series. We observe dataset-dependent gains: collaborative designs help more on Movie-Wiki, with 52.8% and 51.2% larger improvements in Q-A Rel. and Persona. than on Co-Wiki, respectively. Since Movie-Wiki explicitly requires jointly grounding generations on public movie facts and private user preferences, whereas Co-Wiki's tasks exhibit weaker cross-side dependence. BRIDGE consistently performs

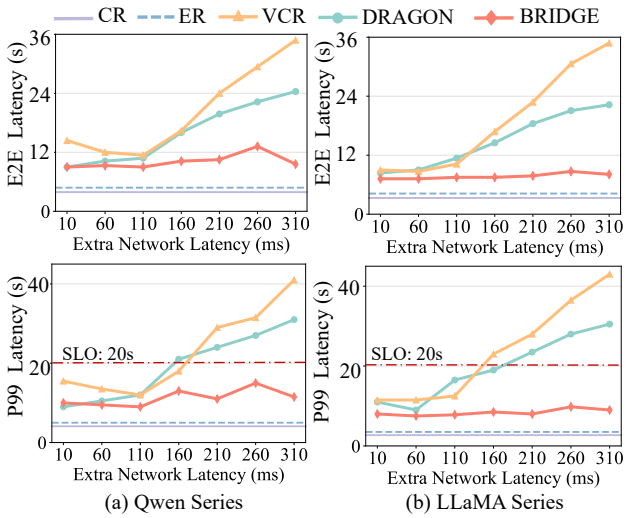


Figure 4: Latency in various network conditions. We inject additional latency to the server NIC (ranging from 10 to 310 ms) to emulate varying network conditions and set the maximum context length to 256 tokens on both sides.

best across both datasets and model series, demonstrating strong generalizability.

4.2.2 Efficiency. Fig. 4 reports the end-to-end latency and P99 latency on Movie-Wiki, reflecting the average efficiency and SLO-relevant tail sensitivity, respectively. As illustrated in Fig. 4, BRIDGE remains efficient across all network settings and consistently outperforms distributed baselines. For example, BRIDGE achieves an average latency reduction of 50.3% and 37.0% when using Qwen series compared to VCR and DRAGON, respectively. Meanwhile, BRIDGE lowers P99 latency by 46.0% on average and by up to 79.1%, indicating a more stable tail behavior under network jitter and better practical SLO compliance. We attribute these gains to block-wise speculation, which reduces communication overhead and mitigates rollback-induced inefficiency in collaborative decoding. Compared to the non-communicating ER and CR baselines, BRIDGE incurs additional latency from cross-side coordination. However, it remains within the SLO budget while supporting tasks that require both personal and general knowledge, where ER and CR may fail.

4.3 Benefit of Knowledge Integration (RQ2)

To directly assess BRIDGE's capability of integrating cloud-edge knowledge, we construct a ground-truth continuation that fuses information retrieved from both corpora. We then measure how well different frameworks align with this standard output using perplexity (PPL) [11]: a lower PPL indicates a higher likelihood of the ground-truth continuation conditioned on both-side retrieved evidence, reflecting stronger cross-source knowledge integration.

As shown in Fig. 5, BRIDGE outperforms all methods across different settings on both datasets, and the performance gap widens as more documents are integrated. In particular, centralized baselines lag behind because they can only access a single corpus. In contrast, context-aggregation EGDR improve initially but quickly saturate once the prompt reaches a context budget, after which additional

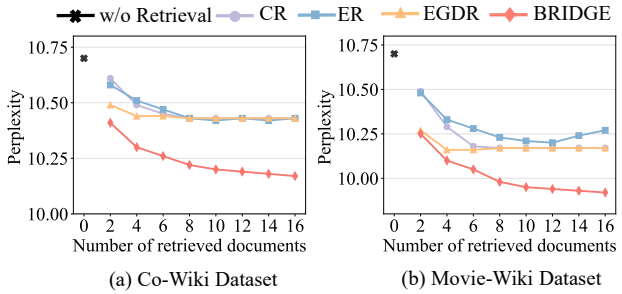


Figure 5: PPL of different frameworks. We use Qwen series and linearly increase the number of retrieved documents.

Table 2: The data reference rate of different frameworks.

Method	Co-Wiki		Movie-Wiki	
	PDRR	UDRR	PDRR	UDRR
ER	-	0.087	-	0.117
CR	0.059	-	0.131	-
EGDR	0.049	0.062	0.047	0.109
BRIDGE	0.058	0.084	0.159	0.101

documents cannot be effectively incorporated. Meanwhile, BRIDGE continues to benefit from more knowledge.

We additionally introduce Public Data Reference Rate (PDRR) and User Data Reference Rate (UDRR) to quantify knowledge coverage, which are define as PDRR $= \frac{|W_{\text{pub}} \cap W_{\text{out}}|}{|W_{\text{pub}}|}$ and UDRR $= \frac{|W_{\text{user}} \cap W_{\text{out}}|}{|W_{\text{user}}|}$, where W_{out} is the set of unique output words in the final model's generated output, W_{pub} and W_{user} are unique words in the public and user-private corpora, respectively. Table 2 shows BRIDGE attains the strongest joint coverage, indicating it integrates knowledge from both sides rather than over-relying on one. The gains are larger on **Movie-Wiki**, which requires grounding on public movie facts and private user preferences, and smaller on **Co-Wiki**, where reliance on public facts is weaker.

4.4 Block-wise Speculation for Lower Communication and Rollback (RQ3)

Fig. 6 compares BRIDGE against its token-wise ablation (w/o Block) under streaming serving with consecutive request arrivals. Across requests, BRIDGE consistently incurs lower communication overhead and fewer rollbacks. Specifically, on the Qwen series, BRIDGE reduces the average communication cost by 32.1% and decreases rollback frequency by 65.7%. On the LLaMA series, it achieves 33.3% and 67.8% reductions on the same metrics.

These gains come from two designs: (*i*) Block-wise transmission reduces the number of cross-side transmissions by batching draft tokens, thereby lowering RTT-dominated communication overhead. (*ii*) Block-wise verification commits the longest accepted prefix within each draft block, which avoids unnecessary rollbacks once a mismatch is detected. Together, block-wise speculation reduces redundant computation and improves serving efficiency.

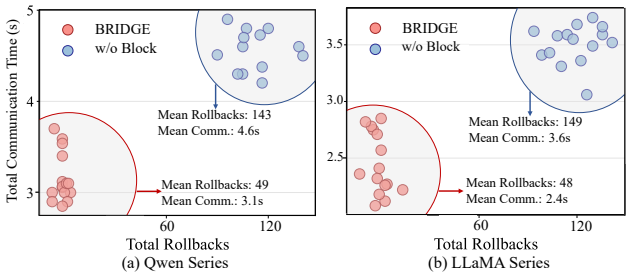


Figure 6: Comparison of BRIDGE and w/o Block on total rollbacks and communication time on Movie-Wiki under 10ms latency. Each point represents a request.

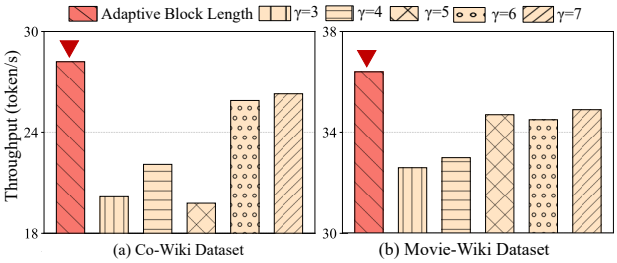


Figure 7: Throughput under adaptive and fixed block lengths. Results are averaged over network latency varying from 10ms to 310ms, using Qwen-series models.

4.5 Effectiveness of Adaptive Block-length Strategy (RQ4)

We further evaluate whether the proposed adaptive block-length yields a better communication–verification trade-off. Fig. 7 compares the average throughput of our adaptive strategy with fixed block lengths under varying network conditions. Across settings, the adaptive policy achieves the highest throughput, outperforming all fixed lengths by an average of 19.1% and 7.3% on Movie-Wiki.

This result reflects the non-uniform optimality of γ under dynamic runtime conditions. Small γ incurs frequent synchronization and RTT-dominated overhead, while overly large γ increases per-iteration cost and may waste computation when acceptance drops, leading to rollback-induced inefficiency. By leveraging runtime signals and enforcing safety constraints on per-iteration latency and acceptance, the adaptive strategy adjusts γ from a high-confidence safe set (satisfying latency and acceptance constraints) and chooses the option with the highest conservative estimated value, thereby improving throughput under fluctuating conditions.

5 Related Work

Retrieval-Augmented Generation. RAG typically retrieves multiple documents to improve LLMs. Centralized cloud-based RAG augments LLMs by retrieving from large-scale public or enterprise corpora to enhance factuality and coverage [5, 18, 21]. In contrast, edge-based RAG retrieves from private on-device stores to enable personalization with stronger privacy guarantees [12, 25, 29]. However, centralized designs cannot jointly leverage public knowledge and private context. While some work introduces enhancement via capability-based routing or collaborative guidance [4, 34], they still fail to address complex tasks that require heterogeneous corpora. In contrast, our BRIDGE integrates both knowledge sources by output aggregation for efficient cloud–edge collaboration.

Speculative Decoding. Speculative decoding accelerates autoregressive generation via a draft-then-verify paradigm, where a lightweight model proposes multiple future tokens and the target model verifies them in parallel [2, 3, 13, 16, 30]. Although recent work extends speculative decoding to cloud-edge setting for faster inference, the conventional token-wise paradigm often incurs communication bottlenecks and substantial rollback waste [28, 31]. In contrast, BRIDGE adopts block-wise transmission and verification to make cloud–edge collaboration markedly more efficient.

6 Conclusion

In this work, we propose BRIDGE, a block-wise speculative RAG framework for inter-database distributed generation. By combining block-wise speculative coordination with an adaptive block-length strategy, BRIDGE jointly addresses communication overhead and rollback-induced computation waste, enabling cloud-edge knowledge integration while preserving service usability. Extensive experiments on public-private benchmarks demonstrate that BRIDGE significantly improves generation quality while reducing end-to-end latency across diverse workloads and network conditions.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (Youth) under Grant No. 62502341; the China Postdoctoral Science Foundation (Certificate No. 2025M771588); the Postdoctoral Fellowship Program (Grade B) of the China Postdoctoral Science Foundation under Grant No. GZB20250410 and the National Key R&D Program of China (Grant No. 2025YFB4506600); Tianjin Natural Science Foundation General Project No. 23JCYBJC00780; the Tianjin Xinchuang Haihe Lab under Grant No.22HHXCJC00002.

References

[1] Akari Asai, Zexuan Zhong, Danqi Chen, Pang Wei Koh, Luke Zettlemoyer, Hannaneh Hajishirzi, and Wen-tau Yih. 2024. Reliable, adaptable, and attributable language models with retrieval. *arXiv preprint arXiv:2403.03187* (2024).
[2] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774* (2024).
[3] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318* (2023).
[4] Lihu Chen and Gaël Varoquaux. 2025. What is the Role of Small Models in the LLM Era: A Survey. arXiv:2409.06857 [cs.CL] https://arxiv.org/abs/2409.06857
[5] Qinwen Chen, Wenbiao Tao, Zhiwei Zhu, Mingfan Xi, Liangzhong Guo, Yuan Wang, Wei Wang, and Yunshi Lan. 2025. ComRAG: Retrieval-Augmented Generation with Dynamic Vector Stores for Real-time Community Question Answering in Industry. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, Georg Rehm and Yunyao Li (Eds.). Association for Computational Linguistics, Vienna, Austria, 749–763. doi:10.18653/v1/2025.acl-industry.53
[6] Aaron Grattafiori et al. 2024. The Llama 3 Herd of Models. arXiv:2407.21783 [cs.AI] https://arxiv.org/abs/2407.21783
[7] An Yang et al. 2025. Qwen2.5 Technical Report. arXiv:2412.15115 [cs.CL] https://arxiv.org/abs/2412.15115
[8] Liu Shangyu et al. 2025. DRAGON: Enhancing On-Device Model Performance with Distributed Retrieval-Augmented Generation. In *Proceedings of the Twenty-Sixth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing* (Rice University, Houston, TX, USA) *(MobiHoc '25)*. Association for Computing Machinery, New York, NY, USA, 221–230. doi:10.1145/3704413.3764419

[9] Shi Weijia et al. 2024. Replug: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 8371–8384.
[10] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv:2312.10997 [cs.CL] https://arxiv.org/abs/2312.10997
[11] Hila Gonen, Srini Iyer, Terra Blevins, Noah A Smith, and Luke Zettlemoyer. 2023. Demystifying prompts in language models via perplexity estimation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. 10136–10148.
[12] Guihang Hong, Tao Ouyang, Kongyange Zhao, Zhi Zhou, and Xu Chen. 2025. CoEdge-RAG: Optimizing Hierarchical Scheduling for Retrieval-Augmented LLMs in Collaborative Edge Computing. In *2025 IEEE Real-Time Systems Symposium (RTSS)*. 162–174. doi:10.1109/RTSS66672.2025.00022
[13] Yunhai Hu, Zining Liu, Zhenyuan Dong, Tianfan Peng, Bradley McDanel, and Sai Qian Zhang. 2025. Speculative decoding and beyond: An in-depth survey of techniques. *arXiv preprint arXiv:2502.19732* (2025).
[14] Jinqi Huang, Yi Xiong, Xuebing Yu, Wenjie Huang, Entong Li, Li Zeng, and Xin Chen. 2025. SLO-Aware Scheduling for Large Language Model Inferences. arXiv:2504.14966 [cs.DC] https://arxiv.org/abs/2504.14966
[15] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.
[16] Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding *(ICML'23)*. Article 795, 13 pages.
[17] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA.
[18] Kunrong Li and Kwan Hui Lim. 2025. RALLM-POI: Retrieval-Augmented LLM for Zero-shot Next POI Recommendation with Geographical Reranking. arXiv:2509.17066 [cs.AI] https://arxiv.org/abs/2509.17066
[19] Ivan Lopez, Akshay Swaminathan, Karthik Vedula, Sanjana Narayanan, Fateme Nateghi Haredasht, Stephen P Ma, April S Liang, Steven Tate, Manoj Maddali, Robert Joseph Gallo, et al. 2025. Clinical entity augmented retrieval for clinical information extraction. *npj Digital Medicine* 8, 1 (2025), 45.
[20] Tyler Lu, Dávid Pál, and Martin Pál. 2010. Contextual multi-armed bandits. In *Proceedings of the Thirteenth international conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 485–492.
[21] Yuxing Lu, Xukai Zhao, and Jinzhuo Wang. 2024. ClinicalRAG: Enhancing Clinical Decision Support through Heterogeneous Knowledge Retrieval. In *Proceedings of the 1st Workshop on Towards Knowledgeable Language Models (KnowLLM 2024)*, Sha Li, Manling Li, Michael JQ Zhang, Eunsol Choi, Mor Geva, Peter Hase, and Heng Ji (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 64–68. doi:10.18653/v1/2024.knowllm-1.6
[22] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer Sentinel Mixture Models. In *International Conference on Learning Representations*.
[23] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-Context Retrieval-Augmented Language Models. *Transactions of the Association for Computational Linguistics* 11 (2023), 1316–1331. doi:10.1162/tacl_a_00605
[24] Michael J. Ryan, Danmei Xu, Chris Nivera, and Daniel Campos. 2025. EnronQA: Towards Personalized RAG over Private Documents. arXiv:2505.00263 [cs.IR] https://arxiv.org/abs/2505.00263
[25] Korakit Seemakhupt, Sihang Liu, and Samira Khan. 2024. Edgerag: Online-indexed rag for edge devices. *arXiv preprint arXiv:2412.21023* (2024).
[26] Rulin Shao, Jacqueline He, Akari Asai, Weijia Shi, Tim Dettmers, Sewon Min, Luke Zettlemoyer, and Pang Wei Koh. 2024. Scaling retrieval-based language models with a trillion-token datastore. In *Proceedings of the 38th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '24)*. Curran Associates Inc., Red Hook, NY, USA, Article 2896, 40 pages.
[27] Yanan Sui, Vincent Zhuang, Joel Burdick, and Yisong Yue. 2018. Stagewise Safe Bayesian Optimization with Gaussian Processes. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*. PMLR, 4781–4789. https://proceedings.mlr.press/v80/sui18a.html
[28] Nadav Timor, Jonathan Mamou, Daniel Korat, Moshe Berchansky, Oren Pereg, Moshe Wasserblat, Tomer Galanti, Michal Gordon-Kiwkowitz, and David Harel. 2025. Distributed Speculative Inference (DSI): Speculation Parallelism for Provably Faster Lossless Language Model Inference. In *The Thirteenth International Conference on Learning Representations (ICLR)*.
[29] Zijie J. Wang and Duen Horng Chau. 2024. MeMemo: On-device Retrieval Augmentation for Private and Personalized Text Generation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Washington DC, USA) *(SIGIR '24)*. Association for Computing Machinery, New York, NY, USA, 2765–2770. doi:10.1145/3626772.3657662
[30] Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. 2023. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. 3909–3925.
[31] Fengze Yu, Leshu Li, Brad McDanel, and Sai Qian Zhang. 2025. DSD: A Distributed Speculative Decoding Solution for Edge-Cloud Agile Large Model Serving. arXiv:2511.21669 [cs.LG] https://arxiv.org/abs/2511.21669
[32] Shenglai Zeng, Jiankun Zhang, Pengfei He, Yiding Liu, Yue Xing, Han Xu, Jie Ren, Yi Chang, Shuaiqiang Wang, Dawei Yin, and Jiliang Tang. 2024. The Good and The Bad: Exploring Privacy Issues in Retrieval-Augmented Generation (RAG). In *Findings of the Association for Computational Linguistics: ACL 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 4505–4524. doi:10.18653/v1/2024.findings-acl.267
[33] Kaiyan Zhang, Jianyu Wang, Ermo Hua, Biqing Qi, Ning Ding, and Bowen Zhou. 2024. CoGenesis: A Framework Collaborating Large and Small Language Models for Secure Context-Aware Instruction Following. In *ACL (1)*.
[34] Yingyi Zhang, Pengyue Jia, Xianneng Li, Derong Xu, Maolin Wang, Yichao Wang, Zhaocheng Du, Huifeng Guo, Yong Liu, Ruiming Tang, et al. 2025. Lsrp: A leader-subordinate retrieval framework for privacy-preserving cloud-device collaboration. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*. 3889–3900.
[35] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems* 36 (2023), 46595–46623.

A Profiling in Adaptive Block Length Strategy

We implement the profiler in two stages:

Offline stage. Before deployment, we initialize the profiler with coarse but stable estimates. For decoding, each side runs a short microbenchmark that mirrors the deployed pipeline (model, batch size, and KV-cache behavior) and fits a simple predictor $\hat{c}^{s}_{\text{dec}}(\cdot)$ as a function of context length. For communication, we adopt a standard latency–bandwidth model for message size g: $\hat{c}_{\text{trans}}(g) = \hat{L} + g/\hat{B}$, where $\hat{L}$ and $\hat{B}$ are estimated from a brief calibration procedure and cached as initial values.

Online stage. During execution, each side timestamps draft-block decoding and reports the measured latency to the Coordinator by piggybacking on existing protocol messages (no extra synchronization). The decoding predictor is updated via an exponential moving average (EMA) to smooth transient noise. For communication, we maintain an EMA of RTT using lightweight ping timestamps and refresh the bandwidth estimate at a lower frequency to limit overhead. Finally, we track acceptance statistics (e.g., EMA of τ_t/γ_t and the full-block acceptance indicator $\mathbb{1}[\tau_t = \gamma_t]$), which stabilizes γ selection under stochastic decoding and network jitter. Overall, the Profiler adds constant per-round overhead.

B FORMAL PROOFS

B.1 Proof of Lemma 1

We prove this lemma using backward induction on i, from i = γ down to i = 1.

(i) According to Eq. (10), when i = γ, $\forall x_{1:\gamma} \in \mathcal{X}$, we have

$$\Pr(\tau \geq \gamma | X_{1:\gamma} = x_{1:\gamma}) = h(x_{1:\gamma}) = p(x_{1:\gamma}). \tag{16}$$

(ii) Assume that the lemma holds for all i $\geq \ell$, when i = $\ell - 1$, we decompose $\Pr(\tau \geq \ell - 1 | X_{1:\ell-1} = x_{1:\ell-1})$ into two mutually exclusive events: $\tau \geq \ell$ (i.e., the accepted length is at least ℓ) and $\tau = \ell - 1$ (i.e., the accepted length is exactly $\ell - 1$), which can be

expressed as

$$\begin{aligned}&\Pr(\tau \geq \ell - 1 | X_{1:\ell-1} = x_{1:\ell-1}) \\ &= \Pr(\tau \geq \ell | X_{1:\ell-1} = x_{1:\ell-1}) + \Pr(\tau = \ell - 1 | X_{1:\ell-1} = x_{1:\ell-1}).\end{aligned} \tag{17}$$

The first term represents the probability that, given the draft prefix $x_{1:\ell-1}$, the algorithm ultimately accepts at least ℓ tokens.

$$\begin{aligned}&\Pr\big(\tau \geq \ell \mid X_{1:\ell-1} = x_{1:\ell-1}\big) \\ &= \sum_{x_{1:\ell} \in \mathcal{X}} \mathcal{P}^s\big(x_{1:\ell} \mid \boldsymbol{c}, x_{\ell-1}\big)\ \Pr\big(\tau \geq \ell \mid X_{1:\ell} = x_{1:\ell}\big) \\ &= \sum_{x_{1:\ell} \in \mathcal{X}} \mathcal{P}_\ell\big(x_{1:\ell} \mid \boldsymbol{c}, x_{\ell-1}\big)\, p(x_{1:\ell}).\end{aligned} \tag{18}$$

Substituting the definition of $p(x_{1:\ell})$ and recognizing $\mathcal{P}$, we get $\Pr(\tau \geq \ell | X_{1:\ell-1} = x_{1:\ell-1})$ as

$$\sum_{x_\ell \in \mathcal{X}} \min\left\{p(x_{1:\ell-1})\mathcal{P}(x_\ell | c, x_{\ell-1}), \mathcal{P}^s(x_\ell | c, x_{\ell-1})\right\}. \tag{19}$$

For the second term, we can get

$$\begin{aligned}&\Pr(\tau = \ell - 1 \mid X_{1:\ell-1} = x_{1:\ell-1}) \\ &= \Pr\left(\tau \geq \ell - 1 \mid X_{1:\ell-1} = x_{1:\ell-1}\right) - \\ &\sum_{x_{1:\ell} \in \mathcal{X}} \mathcal{P}^s(x \mid c, x_{\ell-1}) \cdot \Pr\left(\tau \geq \ell \mid c, x_{1:\ell}\right) \\ &= \sum_{x_{1:\ell} \in \mathcal{X}} \max\{p(x_{1:\ell-1})\mathcal{P}(x \mid c, x_{\ell-1}) - \mathcal{P}^s(x \mid c, x_{\ell-1}), 0\}.\end{aligned} \tag{20}$$

Next, substitute Eq. (19) and Eq. (20) into Eq. (17):

$$\begin{aligned}&\Pr(\tau \geq \ell - 1 \mid X_{1:\ell-1} = x_{1:\ell-1}) \\ &= \sum_{x_\ell \in \mathcal{X}} p(x_{1:\ell-1})\mathcal{P}(x_\ell \mid c, x_{\ell-1}) = p(x_{1:\ell-1}).\end{aligned} \tag{21}$$

B.2 Proof of Theorem 1

We first describe a necessary and sufficient condition for a distribution-preserving verification algorithm:

Lemma 2. *$\forall c, \mathcal{P}^s, \mathcal{P}, \gamma$, Let $X_{1:\gamma}$ be generated from $\mathcal{P}^s(\cdot|c)$ and $X_{1:\tau} = \textsc{BlockVerify}(X_{1:\gamma}, \{\mathcal{P}^s(\cdot \mid c, X_i)\}_{i=0}^{\gamma-1}, \{\mathcal{P}(\cdot \mid c, X_i)\}_{i=0}^{\gamma})$. Let $Z_{\gamma-\tau}$ be a sequence of $\gamma - \tau$ tokens generated from $\mathcal{P}(\cdot|c, X_{1:\tau})$, BlockVerify is said to be distribution-preserving if and only if, $\forall$ c, $\mathcal{P}^s$, $\mathcal{P}$, γ, the following condition holds:*

$$X_{1:\tau}, Z_{\gamma-\tau} \sim \mathcal{P}(\cdot|c). \tag{22}$$

Proof. We first prove the forward direction: (i) If BlockVerify is a distribution-preserving algorithm, then $(X_{1:\tau}, Z_{\gamma-\tau}) \sim \mathcal{P}(\cdot|c)$.

For the forward direction, assume BlockVerify is distribution-preserving. After committing $X_{1:\tau}$, the subsequent tokens generated by the verifier follow the target distribution conditioned on the updated prefix. Hence the next $\gamma-\tau$ tokens, denoted by $Z_{\gamma-\tau}$, satisfy $Z_{\gamma-\tau} \sim \mathcal{P}(\cdot \mid c, X_{1:\tau})$. Thus the completed block $(X_{1:\tau}, Z_{\gamma-\tau})$ has the same distribution as the first γ tokens sampled from $\mathcal{P}(\cdot \mid c)$, i.e., $(X_{1:\tau}, Z_{\gamma-\tau}) \sim \mathcal{P}(\cdot \mid c)$.

For the backward direction, if the completed block $(X_{1:\tau}, Z_{\gamma-\tau})$ follows $\mathcal{P}(\cdot \mid c)$ for every context c, then recursively applying the same argument after each committed block implies that the entire output sequence follows the target distribution. Therefore, BlockVerify is distribution-preserving.

We prove the claim by induction on the maximum generation length under $\mathcal{P}$. The base case is immediate: if $\mathcal{P}(\cdot \mid c)$ generates E.O.S. with probability one, then the assumption $(X_{1:\tau}, Z_{\gamma-\tau}) \sim \mathcal{P}(\cdot \mid c)$ forces BlockVerify to terminate immediately, matching $\mathcal{P}(\cdot \mid c)$. For the inductive step, assume the claim holds whenever the remaining generation length is at most T. Since the first output $X_{1:\tau}$ is nonempty, the remaining target distribution $\mathcal{P}(\cdot \mid c, X_{1:\tau})$ has length at most T, so the inductive hypothesis gives

$$Z_{\gamma-\tau} \sim \mathcal{P}(\cdot \mid c, X_{1:\tau}), \tag{23}$$

$$O^* \sim \mathcal{P}(\cdot \mid c, X_{1:\tau}, Z_{\gamma-\tau}). \tag{24}$$

Through Eq. (22), we have

$$\begin{aligned}&\textsc{BlockVerify}(c, \mathcal{P}, \mathcal{P}^s, \eta, \gamma) \\ &= X_{1:\tau}, \textsc{BlockVerify}((c, X_{1:\tau}), \mathcal{P}, \mathcal{P}^s, \eta, \gamma) \\ &= X_{1:\tau}, Z_{\gamma-\tau}, O^* \sim \mathcal{P}(\cdot \mid c).\end{aligned} \tag{25}$$

This completes the proof of Lemma 2. □

According to Lemma 2, it is sufficient to prove that BlockVerify satisfies Eq. (22). For brevity, we often refer to the sequence $(X_{1:\tau}, Z_{\gamma-\tau})$ by O_γ. Note that $O_\gamma \sim \mathcal{P}(\cdot \mid c, O_\gamma)$ always holds since when $\tau < \gamma, O_{\gamma+1} \sim \mathcal{P}(\cdot \mid \boldsymbol{c}, O_\gamma)$ by definition and when $\tau = \gamma$, $O_\gamma = Y \sim \mathcal{P}(\cdot|\boldsymbol{c}, X^\gamma) = \mathcal{P}(\cdot|\boldsymbol{c}, O_\gamma)$. Hence it is enough to prove the following

$$\forall \ell \leq \gamma, \forall x_{1:\ell} \in \mathcal{X}_{1:\ell}, \quad \Pr\left(O_\ell = x_{1:\ell}\right) = \mathcal{P}(x_{1:\ell} \mid c). \tag{26}$$

Based on Lemma 1, we further prove Eq. (26) by induction on the index ℓ.

First, when $\ell = 1$, $O_1 = x_1$ can be expressed as the sum of two mutually exclusive events: $(i)\tau \geq 1$ and x_1 is part of the accepted prefix , $(ii)\tau = 0$ and x_1 is obtained by sampling from the $P_{\text{res}}(\cdot \mid c)$, i.e., $\Pr\left(O_1 = x_1\right) = \Pr\left(O_1 = x_1, \tau \geq 1\right) + \Pr\left(O_1 = x_1, \tau = 0\right)$.

By Eq. (7) and the definition of $p(x_i)$ in Eq. (6), for the first term $\tau \geq 1$, we have

$$\begin{aligned}&\Pr\left(O_1 = x_1, \tau \geq 1\right) \Pr\left(X_1 = x_1\right) \cdot \Pr(\tau \geq 1 \mid X_1 = x_1) \\ &= \mathcal{P}^s(x_1 \mid c) \cdot p(x_1) \\ &= \min\{\eta^{edge}\mathcal{P}^{edge}(x_1|c) + \eta^{cloud}\mathcal{P}^{cloud}(x_1|c), \mathcal{P}_l(x_1|c)\} \\ &= \min\{\mathcal{P}(x_1|c), \mathcal{P}^s(x_1|c)\}.\end{aligned} \tag{27}$$

For the second term $\tau = 0$, we have

$$\begin{aligned}&\Pr\left(O_1 = x_1, \tau = 0\right) \\ &= \Pr\left(\tau = 0\right) \cdot \Pr(X' = x_1 \mid \tau = 1) \\ &= \sum_{x'} \max\{\mathcal{P}(x' \mid \boldsymbol{c}) - \mathcal{P}^s(x' \mid \boldsymbol{c}), 0\} \cdot p_{\text{res}}(x \mid \boldsymbol{c}) \\ &= \max\{\mathcal{P}(x \mid \boldsymbol{c}) - \mathcal{P}^s(x \mid \boldsymbol{c}), 0\}.\end{aligned} \tag{28}$$

Combining Eq. (27) and Eq. (28):

$$\begin{aligned}&\Pr(O_1 = x_1) \\ &= \min\left\{\mathcal{P}(x_1|c), \mathcal{P}^s(x_1|c)\right\} + \max\left\{\mathcal{P}(x_1|c) - \mathcal{P}^s(x_1|c), 0\right\} \\ &= \mathcal{P}(x_1|c).\end{aligned} \tag{29}$$

Therefore the Eq. (26) holds for $\ell = 1$.

Then, we assume that Eq. (26) holds up to $\ell < \gamma$, for $\ell = \ell+1$, $O_{\ell+1}$ can be expressed as the sum of three mutually exclusive events: (i) $\tau \geq \ell + 1$ and $O_{\ell+1}$ is equal to $X_{\ell+1}$, (ii) $\tau = \ell$ and $O_{\ell+1}$ is a sample from $p_{\text{res}}(\cdot \mid c, x_{1:\ell})$, (iii) $\tau < \ell$ and $O_{\ell+1}$ is a sample from

$\mathcal{P}(\cdot \mid c, O_\ell)$, i.e., $\Pr(O_{\ell+1} = x_{\ell+1}) = \Pr(O_{\ell+1} = x_{\ell+1}, \tau \geq \ell+1) + \Pr(O_{\ell+1} = x_{\ell+1}, \tau = \ell) + \Pr(O_{\ell+1} = x_{\ell+1}, \tau < \ell)$.

For the first term $\tau \geq \ell+1$, we have

$$\begin{aligned}
&\Pr(O_{\ell+1} = x_{\ell+1}, \tau \geq \ell+1) \\
&= \Pr(x_{\ell+1} = x_{\ell+1}) \cdot \Pr(\tau \geq \ell+1 \mid x_{\ell+1} = x_{\ell+1}) \\
&= \mathcal{P}^s(x_{\ell+1} \mid c) \cdot p(x_{\ell+1}) \\
&= \mathcal{P}^s(x_{1:\ell} \mid c) \cdot \min\left\{p(x_{1:\ell})\,\mathcal{P}(x_{\ell+1} \mid c, x_{1:\ell}), \mathcal{P}^s(x_{\ell+1} \mid c, x_{1:\ell})\right\}.
\end{aligned} \tag{30}$$

For the second term ($\tau = \ell+1$), we have

$$\begin{aligned}
&\Pr(O_{\ell+1} = x_{\ell+1}, \tau = \ell) = \Pr(x_{1:\ell} = x_{1:\ell}) \\
&\cdot \Pr(\tau = \ell \mid x_{1:\ell} = x_{1:\ell}) \cdot \Pr(O_{\ell+1} = x_{\ell+1} \mid O_\ell = x_{1:\ell}, \tau = \ell).
\end{aligned} \tag{31}$$

According to Eq. (28) and Eq. (11), we have

$$\begin{aligned}
&\Pr(O_{\ell+1} = x_{\ell+1}, \tau = \ell) \\
&= \mathcal{P}^s(x_{1:\ell} \mid c) \cdot \sum_{x_{1:\ell} \in \mathcal{X}} \max\{p(x_{\ell-1})\mathcal{P}(x \mid c, x_{\ell-1}) \\
&- \mathcal{P}^s(x \mid c, x^{\ell-1}), 0\} \cdot p_{\text{res}}(x_{l+1} \mid c, x_{1:\ell}) = \mathcal{P}^s(x_{1:\ell} \mid c) \\
&\cdot \max\left\{p(x_{1:\ell})\,\mathcal{P}^s(x_{\ell+1} \mid c, x_{1:\ell}) - \mathcal{P}^s(x_{\ell+1} \mid c, x_{1:\ell}), 0\right\}.
\end{aligned} \tag{32}$$

For the third term ($\tau < \ell$), we have

$$\begin{aligned}
&\Pr(O_{\ell+1} = x_{\ell+1}, \tau < \ell) \\
&= \Pr(O_\ell = x_{1:\ell}, \tau < \ell) \cdot \Pr(O_{\ell+1} = x_{\ell+1} \mid O_\ell = x_{1:\ell}, \tau < \ell) \\
&= (\Pr(O_\ell = x_{1:\ell}) - \Pr(O_{\ell+1} = x_{\ell+1}, \tau > \ell)) \cdot \mathcal{P}(x_{\ell+1} \mid c, x_{\ell-1}) \\
&= (\mathcal{P}(x_{1:\ell}|c) - \mathcal{P}(x_{1:\ell}|c)p(x_{1:\ell}))\,\mathcal{P}(x_{\ell+1}|c, x_{1:\ell}).
\end{aligned} \tag{33}$$

Summing Eq. (30), Eq. (32) and Eq. (33), we get $\forall x_{\ell+1} \in \mathcal{X}_{\ell+1}$,

$$\Pr(O_{\ell+1} = x_{\ell+1}) = \mathcal{P}(x_{\ell+1} \mid c). \tag{34}$$

The inductive step is completed, proving Theorem 1.

B.3 Proof of Theorem 2

The following lemma, combined with Lemma 1, establishes that BlockVerify achieves the largest possible prefix-acceptance probability among all distribution-preserving verifiers in each iteration.

Lemma 3. *For any distribution-preserving verification algorithms, we have $\forall i \leq \gamma$, and $x_{1:i} \in \mathcal{X}_{1:i}$,*

$$\Pr(\tau \geq i \mid X_{1:i} = x_{1:i}) \leq p(x_{1:i}). \tag{35}$$

Proof. Fix an arbitrary committed context c and suppress c in the notation. For any $i \in [\gamma]$ and any draft prefix $x_{1:i}$, define the survival probability $a(x_{1:i}) \triangleq \Pr(\tau \geq i \mid X_{1:i} = x_{1:i})$. We prove that $a(x_{1:i}) \leq p(x_{1:i})$ for all $i \leq \gamma$ and all $x_{1:i}$.

For any $i \in [\gamma]$, any prefix $x_{1:i-1}$, and any $x_i \in \mathcal{X}$ with $P_s(x_i \mid x_{1:i-1}) > 0$, we have

$$\mathcal{P}^s(x_i \mid x_{1:i-1})\, a(x_{1:i}) \;\leq\; \mathcal{P}(x_i \mid x_{1:i-1})\, a_{i-1}(x_{1:i-1}). \tag{36}$$

Condition on $X_{1:i-1} = x_{1:i-1}$. Distribution preservation requires the mass of token x_i at position i within the surviving pool to be at most $\mathcal{P}(x_i \mid x_{1:i-1})a_{i-1}(x_{1:i-1})$. The draft-copying branch alone contributes $\mathcal{P}^s(x_i \mid x_{1:i-1})a(x_{1:i})$, while residual branches are nonnegative; hence Eq. (36) must hold. Rearranging it and using $a(x_{1:i}) \leq 1$ yields

$$a(x_{1:i}) \;\leq\; \min\left\{a_{i-1}(x_{1:i-1}) \cdot \frac{\mathcal{P}(x_i \mid x_{1:i-1})}{\mathcal{P}^s(x_i \mid x_{1:i-1})},\; 1\right\}. \tag{37}$$

Comparing Eq. (6) with Eq. (37), we see that $a(x_{1:i})$ satisfies the same one-step upper bound as p, starting from $a_0 = p_0 = 1$. Therefore, by induction on i, $a(x_{1:i}) \leq p(x_{1:i})$ for all $i \leq \gamma$ and all $x_{1:i}$. Removing the suppressed conditioning on c gives the desired statement: $\Pr(\tau \geq i \mid X_{1:i} = x_{1:i}) \leq p(x_{1:i})$. □

Let $\mathcal{V}$ be any *distribution-preserving* verification algorithm. We compare $\mathcal{V}$ with BlockVerify.

Step 1: one-step tail dominance. As the number of *newly committed* tokens in this iteration is $\Delta N = \tau$, for every $k \in \{1, 2, \ldots, \gamma\}$, we have the identity

$$\Pr(\Delta N \geq k \mid c) \;=\; \Pr(\tau \geq k-1 \mid c), \tag{38}$$

since $\Delta N \geq k$ holds iff either $\tau \geq k$ or $\tau = k-1$, which is equivalent to $\tau \geq k-1$. By Lemma 3, for any $j \in \{0, 1, \ldots, \gamma-1\}$ and any realized draft prefix $x_{1:j}$,

$$\Pr_{\mathcal{V}}(\tau \geq j \mid X_{1:j} = x_{1:j}, c) \;\leq\; p(x_{1:j}).$$

Taking expectation over $X_{1:j} \sim \mathcal{P}^s(\cdot \mid c)$ yields

$$\Pr_{\mathcal{V}}(\tau \geq j \mid c) \;\leq\; \mathbb{E}_{X_{1:j} \sim \mathcal{P}^s(\cdot|c)}\left[p(X_{1:j})\right]. \tag{39}$$

Moreover, BlockVerify is tight (Lemma 1), i.e., $\Pr_{\text{BlockVerify}}(\tau \geq j \mid X_{1:j} = x_{1:j}, c) = p(x_{1:j})$, so the right-hand side of Eq. (39) equals $\Pr_{\text{BlockVerify}}(\tau \geq j \mid c)$. Therefore, for all $j \in \{0, \ldots, \gamma-1\}$,

$$\Pr_{\mathcal{V}}(\tau \geq j \mid c) \;\leq\; \Pr_{\text{BV}}(\tau \geq j \mid c). \tag{40}$$

Combining Eq. (38) with Eq. (40) (using $j = k-1$) gives, for all $k \in \{1, \ldots, \gamma\}$,

$$\Pr_{\mathcal{V}}(\Delta N \geq k \mid c) \;\leq \Pr_{\text{BV}}(\Delta N \geq k \mid c), \tag{41}$$

i.e., conditioned on any context c, its one-step progress ΔN stochastically dominates that of any distribution-preserving verifier.

Step 2: one-step expected progress is maximized. By the tail-sum formula and Eq. (41), for any fixed context c,

$$\begin{aligned}
&\mathbb{E}_{\mathcal{V}}[\Delta N \mid c] \sum_{k=1}^{\gamma} \Pr_{\mathcal{V}}(\Delta N \geq k \mid c) \;\leq \\
&\sum_{k=1}^{\gamma} \Pr_{\text{BV}}(\Delta N \geq k \mid c) \mathbb{E}_{\text{BlockVerify}}[\Delta N \mid c].
\end{aligned} \tag{42}$$

Step 3: lift to i iterations via tower property over contexts. Let C_t denote the random committed context after t iterations, with $C_0 = c_0$. Write ΔN_{t+1} for the newly committed tokens. Then

$$\mathbb{E}_{\mathcal{V}}[N(i)] \sum_{t=0}^{i-1} \mathbb{E}_{\mathcal{V}}[\,\mathbb{E}_{\mathcal{V}}[\Delta N_{t+1} \mid C_t]\,]. \tag{43}$$

By Eq. (42), the inner expectation is pointwise bounded for every realized context c: $\mathbb{E}_{\mathcal{V}}[\Delta N_{t+1} \mid C_t = c] \;\leq\; \mathbb{E}_{\text{BlockVerify}}[\Delta N_{t+1} \mid C_t = c]$. Finally, since both $\mathcal{V}$ and BlockVerify are distribution-preserving (Theorem 1), the distribution of C_t is the same target-induced context distribution under either verifier. Therefore, taking the outer expectation over C_t preserves the inequality term-by-term in Eq. (43), yielding

$$\mathbb{E}_{\mathcal{V}}[N(i)] \;\leq\; \mathbb{E}_{\text{BlockVerify}}[N(i)]. \tag{44}$$

Thus BlockVerify maximizes $\mathbb{E}[N(i)]$ among all distribution-preserving verifiers.