Beyond Token Time

Persistent Multi-Stream Latent World States for Modality-Agnostic Intelligence

Position Paper and Research Proposal

Nako Sung
NAVER Cloud, Hyperscale AI
August 2026

Abstract. Modern language models maintain rich continuous hidden states, yet their prevailing operating regime repeatedly projects those states through a discrete token interface and advances on a single token clock. Continuous latent reasoning, multi-stream language modeling, and multimodal architectures each relax part of this constraint: intermediate states need not always be verbalized, several token streams can coexist, and heterogeneous observations can share latent computation. Building on these results, this paper advances a broader architectural hypothesis: intelligent computation can be organized around persistent structured state, selective event-triggered updates, and modality-specific readout rather than a single global token sequence. A Latent Event Bus coordinates a Persistent Multi-Stream Latent World State. Its event order respects causal precedence and imposes an explicitly declared observer-relative chronology only where no causal relation exists. The hypothesis is situated against prior work and paired with compute-matched experiments designed to falsify it.

1 Introduction

Autoregressive language models follow a simple rhythm. Given a token prefix $x_{\leq t}$, a transformer computes a continuous hidden representation h_t, maps it to a vocabulary distribution, selects a discrete symbol, and repeats:

$$h_t \longrightarrow x_{t+1} \longrightarrow h_{t+1}. \tag{1}$$

This rhythm is well suited to language generation. Whether internal computation must follow the same rhythm is less clear. Coconut weakens that coupling by feeding a hidden state back as a subsequent input embedding rather than decoding every intermediate state into a token [1]. Removing intermediate tokens, however, addresses only one constraint. Computation is still commonly framed as a single sequence advancing under a global step index.

Interactive environments are not synchronized this way. A conversational agent may listen, update beliefs, plan, speak, and detect interruptions concurrently. A robot may update control at high frequency while invoking planning only when needed. A scene contains many entities whose states evolve at different rates.

Central question. What if the fundamental unit of intelligent computation were not the next token, but the next relevant event?

The contribution lies not in latent reasoning, parallel streams, multimodality, or reinforcement learning alone, but in their architectural combination: a general agent maintains persistent, structured, asynchronous latent processes and updates them in response to events rather than token production.

2 Modalities as Lenses

Let s denote an underlying world state. Observing it through modality m yields

$$o^{(m)} = g_m(s) + \epsilon^{(m)}, \tag{2}$$

where g_m is a modality-specific measurement or projection and $\epsilon^{(m)}$ captures noise and information loss. An RGB image, speech waveform, text description, depth map, or motor signal is not the world itself; each is a partial lens on it.

Shared latent spaces are not new. Perceiver IO demonstrated a general latent-processing architecture for heterogeneous structured inputs and outputs [2]. The proposal here is more operational: the shared representation persists across interactions, evolves recurrently, and contains multiple processes with distinct update schedules.

Design principle. A modality is an interface to internal state, not necessarily the format of thought.

2.1 Video generation is multi-stream

Video generation makes the multi-stream requirement concrete. A model rendering two actors in conversation must preserve actor identity, per-actor pose and motion, turn-taking, voice-to-character binding, lip synchronization, camera motion, shared scene state, and ambient audio along one evolving timeline. Multi-person conversational video research exposes the binding problem directly: several audio streams must remain attached to the correct visible people over time [8]. Native audio-video generators with synchronized dialogue provide a broader system-level example [9].

At the behavioral level, this capability demonstrates that

one generator can coordinate several concurrently evolving processes. It does not establish that the implementation contains explicit or independently addressable latent stream slots; the internal representation may remain entangled. The architectural question is whether making those processes persistent, selective, and schedulable would improve control, efficiency, interruption handling, and consistency.

3 From Token Time to Event Time

At generation time, a standard autoregressive model couples each increment to a token position. Call this cadence *token time*. A general agent may instead maintain audio, vision, planning, memory, and motor processes with different update schedules:

$$t_{\text{token}} \neq t_{\text{compute}} \neq t_{\text{world}}. \tag{3}$$

Computation may instead be scheduled by events $e_1, e_2, \ldots$ that activate only the relevant state components. A new acoustic observation may update speaker and semantic state; a sudden obstacle may activate vision, planning, and control. If no verbal response is needed, no language token need be produced.

3.1 Observer-relative event order

Deterministic event processing still requires a reproducible order, but that order must not be confused with causation. Let $e_i \prec_c e_j$ mean that e_i can causally influence e_j. The relation is a partial order: some event pairs remain unrelated.

A declared observer O and reference frame $\mathcal{F}_O$ assign each modeled world event a coordinate time $t_{\mathcal{F}_O}(e)$. The event bus can then use the lexicographic key

$$K_O(e) = \big(t_{\mathcal{F}_O}(e),\, \tau_O(r_e),\, \iota(e)\big), \tag{4}$$

where r_e is the local receipt event, $\tau_O(r_e)$ is the observer's monotonic local clock at receipt, and $\iota(e)$ is a stable final tie-breaker. The induced order is

$$e_i \prec_O e_j \quad \Longleftrightarrow \quad K_O(e_i) <_{\text{lex}} K_O(e_j). \tag{5}$$

The frame and clock are admissible only when they preserve causal precedence:

$$e_i \prec_c e_j \implies e_i \prec_O e_j. \tag{6}$$

For causally unrelated events, $\prec_O$ records chronology in the selected frame without implying that one event caused the other. The result is a deterministic total extension of the causal partial order, conceptually related to the distinction between causal order and logical clock order in distributed systems [6].

The physics analogy requires one qualification. *Proper time* is measured by a clock along a single worldline. It can order the receipt events r_e observed locally by one agent, but it cannot by itself order remote events at different locations. That broader ordering requires coordinate time in a declared reference frame, including its convention for simultaneity [7].

Invariant precedence, declared chronology. Causal order is preserved when it exists. Otherwise the system records whose frame defines "earlier" and "later."

3.2 Causal masking is not a world clock

For a causal decoder, the autoregressive objective factorizes a sequence as

$$p_\theta(x_{1:T}) = \prod_{t=1}^{T} p_\theta(x_t \mid x_{<t}), \tag{7}$$

while the decoder applies an attention mask $M_{ij} = -\infty$ for $j > i$. Position i therefore cannot read subsequent token positions during that forward pass [10]. The mask is an externally supplied information-flow constraint over the chosen serialization, not a learned physical causal order among the events represented by the tokens.

This does not make the learned weights order-neutral. Training shapes them under the causal factorization, positional scheme, and data distribution, so they may encode order-sensitive regularities. The narrower point is that the mask itself is not a learned parameter; neither an explicit observer frame nor a world-event order follows from the weights alone.

By analogy with in-context conditioning, a frame specification $c_{\mathcal{F}}$ could select a readout without changing the weights [11, 12]:

$$y_{\mathcal{F}} = D_\theta(Z; c_{\mathcal{F}})\,. \tag{8}$$

If Z retains information compatible with several viewpoints, $c_{\mathcal{F}}$ may induce a frame-consistent serialization. This possibility is termed *frame-conditioned latent readout*. As an intuition rather than a mechanism claim, it resembles *holographic restoration*: the prompt acts like a boundary condition that reconstructs one view from distributed state. The metaphor does not imply that the weights literally implement a hologram, contain a complete frame-independent world, or preserve invariants without targeted training and evaluation.

Architectural constraint versus learned consequence. Causal masking enforces access order over token positions, and training under that constraint shapes the weights. Whether a frame prompt can recover faithful cross-frame views from the resulting state is an empirical question.

4 Persistent Multi-Stream Latent State

At event e_n in the declared order, let the model maintain

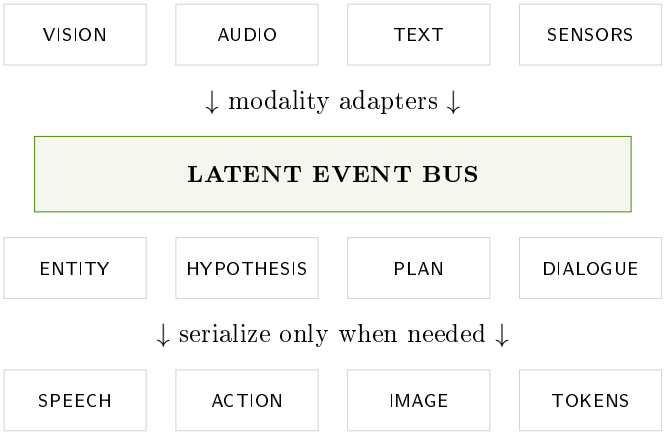
$$Z_n = \left[z_n^{(1)}, z_n^{(2)}, \ldots, z_n^{(K)} \right]. \tag{9}$$

Persistent streams may represent physical entities, goals, memory elements, candidate hypotheses, predicted futures, perceptual processes, or control processes. A routing policy produces an activity mask

$$a_n = \pi_{\text{route}}(Z_n, e_n), \qquad a_n \in \{0,1\}^K, \tag{10}$$

and updates only selected components:

$$z_{n+1}^{(k)} = \begin{cases} F_k\left(z_n^{(k)}, e_n, Z_n\right), & a_n^{(k)} = 1, \\ z_n^{(k)}, & a_n^{(k)} = 0. \end{cases} \tag{11}$$



The Latent Event Bus is a shared representational and routing layer that connects persistent latent streams to observations and outputs. Modality-specific encoders and decoders provide interfaces without prescribing the internal reasoning format.

Latent continuation offers one implementation primitive: an internal action bypasses vocabulary decoding and feeds a projected hidden state back into computation. This mechanism is closely related to Coconut [1] and is not the primary novelty claimed here.

5 Delayed Commitment

When several hypotheses remain plausible, they can remain components of one persistent state rather than becoming complete, independent textual trajectories. Recent work also scales latent reasoning through parallel stochastic trajectories and latent reward models [4]. Continuous representations do not uniquely enable alternatives; the narrower proposal is to maintain unresolved alternatives as coordinated components of a persistent world model.

The argument does not depend on a continuous vector containing unlimited information. Discrete output imposes a particular quantization and semantic commitment; delaying that commitment may preserve distinctions that matter to later decisions.

6 Relation to Prior Work

Coconut. Continuous hidden-state reasoning and latent alternatives are established [1]. The present proposal extends that state across perception, reasoning, and action.

Perceiver IO. General latent processing across heterogeneous inputs and outputs is established [2]. The present proposal adds recurrent persistence and selective event-triggered updates.

Hybrid latent reasoning. Reinforcement learning can optimize latent and discrete reasoning policies [3]. Here it would train update and scheduling policies; RL itself is not the claimed novelty.

Parallel latent test-time scaling. Multiple latent trajectories can be sampled and ranked [4]. Here, by contrast, streams may represent entities, goals, modalities, or hypotheses within one state.

Multi-Stream LLMs. Several token streams can read and write in parallel [5]. Here internal streams need be neither tokenized nor synchronized to one update clock.

Intended novelty. The proposed combination is persistent structured latent multiplicity, event-driven scheduling, and explicit observer-relative event order.

7 Training

Stage I - conventional pretraining. Language or multimodal predictive training initializes useful representations and modality adapters.

Causal masking at this stage enforces left-to-right access within each serialized example; it does not supervise a unique physical chronology. Separating causal invariants from observer-dependent order therefore requires additional training signals, such as frame labels, causal graphs, or multi-frame renderings.

Stage II - latent-state bootstrapping. Recurrence, persistent slots, and routing are introduced through next-state prediction, entity tracking, masked-state reconstruction, cross-modal prediction, or distillation.

Stage III - outcome-driven optimization. Reinforcement learning is a natural candidate because many internal trajectories may solve the same task. Hybrid latent reasoning establishes RL over latent and discrete policies [3]; here the target policy would control persistent asynchronous state.

A generic objective can reward task success while penalizing unnecessary computation, unnecessary serialization, unstable recurrent dynamics, and violations of the declared event order:

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_c C + \lambda_s S + \lambda_d D + \lambda_o O. \tag{12}$$

8 Experimental Program

An initial study should be deliberately small and compare four compute-matched systems: (A) sequential discrete reasoning, (B) parallel token streams, (C) sequential latent recurrence, and (D) asynchronous multi-stream latent state.

8.1 Asynchronous multi-entity world

Construct an environment with independently evolving entities and heterogeneous update rates. As entity count increases, measure entity-state accuracy, retention, cross-stream interference, FLOPs, latency, and memory. Permute the delivery order of causally unrelated events while retaining frame timestamps; a correct implementation should reproduce the same observer-relative state trajectory.

8.2 Delayed evidence

Early observations support several hypotheses; later evidence resolves them. Measure hypothesis diversity, contradiction recovery, and revision cost.

8.3 Interruption

Begin an output or action and inject new information before completion. Measure reaction latency, reusable computation, consistency, and unnecessary recomputation.

8.4 Cross-modal action without text

Provide vision and audio and require a non-linguistic action. Compare explicit text-mediated pipelines against direct latent-state-to-action behavior.

8.5 Multi-actor dialogue generation

Generate scenes with two or more actors whose lines overlap, alternate, or are interrupted. Measure identity persistence, speaker-to-voice binding, lip synchronization, turn-order accuracy, cross-actor leakage, and recovery from an unscripted event. Under matched compute, compare an entangled baseline with explicit actor-indexed persistent streams.

8.6 Frame and receipt-order ablation

Compare three schedulers: arrival order alone, chosen-frame coordinate time alone, and the causal-consistent key K_O. Measure replay stability, stale-state writes, causal inversions, and sensitivity to transport jitter. Repeat the experiment under alternative declared frames to distinguish invariant causal predictions from frame-dependent chronology.

8.7 Frame-conditioned reconstruction

Serialize the same causal event graph under two or more declared frames, then condition the model on $c_{\mathcal{F}}$ to reconstruct each view. Hold causal ancestry, entity identity, and outcomes fixed; vary only the coordinate order of unrelated events. Measure invariant accuracy, frame-dependent chronology, and round-trip consistency under $\mathcal{F}_1 \to \mathcal{F}_2 \to \mathcal{F}_1$. Ablate the frame prompt, explicit frame supervision, causal mask, and the order of unrelated training events. Failure to preserve invariants would indicate prompt-conditioned narration rather than reconstruction from a coherent latent event structure.

Killer ablation. If asynchronous structured state does not outperform simpler sequential or parallel baselines under matched compute on tasks designed around independent clocks, interruption, and event-order perturbation, the additional architecture is not justified.

9 Failure Modes and Falsification

- **Stream collapse:** multiple slots converge to effectively identical representations.
- **Fragmentation:** isolated streams fail to maintain global consistency.
- **Latent drift:** recurrent updates push state outside the pretrained distribution.
- **Scheduling overhead:** routing costs exceed the savings from selective updates.
- **False concurrency:** parallel slots merely disguise a serial computation.
- **Frame leakage:** changing only the declared frame alters predictions that should remain invariant.
- **Prompted confabulation:** the model produces a plausible frame-specific narrative without a cross-frame-consistent latent event structure.
- **Order overclaim:** chronological order is misread as evidence of causation.

10 Discussion

Because text is abundant and next-token prediction scales well, language models encourage communication, reasoning, memory, and scheduling to be organized around one sequence. Yet the architecture through which a capability is discovered need not be the architecture through which it is best organized.

The world does not arrive as a sentence. A room contains many entities at once. Perception need not pause during action, and planning need not stop when new evidence arrives. One hypothesis need not become a sentence before another can coexist with it.

Event time should not become another hidden global counter. The event bus instead requires a declared perspective: preserve causal precedence, identify the observer and reference frame that determine chronology, and distinguish local receipt time from modeled occurrence time.

Question for architecture. Why should language simultaneously serve as interface, reasoning trace, memory representation, scheduler, and computational clock?

11 Conclusion

Continuous latent reasoning removes the need to verbalize every internal step. Parallel-stream models relax the assumption of a single message stream, while multimodal architectures allow heterogeneous observations to share latent computation. The proposed primitive combines these directions: persistent asynchronous state updated in the order of relevant events.

Compute when an event requires computation. Update the state that the event affects. Preserve causal precedence. Declare the frame used for everything else. Serialize only when interaction requires it.

Tokens are interfaces. Events move the world.

References

[1] S. Hao, S. Sukhbaatar, D. Su, X. Li, Z. Hu, J. Weston, and Y. Tian, "Training Large Language Models to Reason in a Continuous Latent Space," arXiv:2412.06769, 2024.

[2] A. Jaegle, S. Borgeaud, J.-B. Alayrac, et al., "Perceiver IO: A General Architecture for Structured Inputs and Outputs," ICLR, 2022.

[3] Z. Yue, B. Jin, H. Zeng, et al., "Hybrid Latent Reasoning via Reinforcement Learning," NeurIPS, 2025.

[4] R. You, Y. Li, M. Liu, W. Wang, L. Nie, and W. Li, "Parallel Test-Time Scaling for Latent Reasoning Models," ACL, 2026.

[5] G. Su, Y. Yang, X. Li, and J. Geiping, "Multi-Stream LLMs: Unblocking Language Models with Parallel Streams of Thoughts, Inputs and Outputs," arXiv:2605.12460, 2026.

[6] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," Communications of the ACM, vol. 21, no. 7, pp. 558-565, 1978.

[7] A. Einstein, "On the Electrodynamics of Moving Bodies," Annalen der Physik, vol. 17, pp. 891-921, 1905.

[8] Z. Kong, F. Gao, Y. Zhang, et al., "Let Them Talk: Audio-Driven Multi-Person Conversational Video Generation," arXiv:2505.22647, 2025.

[9] OpenAI, "Sora 2 System Card," September 2025.

[10] A. Vaswani, N. Shazeer, N. Parmar, et al., "Attention Is All You Need," NeurIPS, 2017.

[11] T. Brown, B. Mann, N. Ryder, et al., "Language Models are Few-Shot Learners," NeurIPS, 2020.

[12] S. M. Xie, A. Raghunathan, P. Liang, and T. Ma, "An Explanation of In-Context Learning as Implicit Bayesian Inference," ICLR, 2022.