LSV: Efficient Reversible GPU Multiplexing for LLM Serving

Anonymous authors
Paper under double-blind review

Abstract

Intra-GPU multiplexing lets Prefill and Decode share one LLM replica while using different GPU resources. The useful split changes with the request mix. Existing controllers can select a new split quickly. But work already submitted under the old split can delay the first useful work under the new split by milliseconds. We call this delay *reconfiguration debt*. We present Logical Stream Virtualization (LSV), an execution layer with stable logical lanes for Prefill and Decode. LSV tracks issued and completed work with resource epochs. It limits future old-epoch commitments and changes both lanes at one safe boundary. We implement LSV in SGLang and evaluate it on an eight-H100 NVLink server across three model scales and two model families. A controlled test shows that host handoff stays near 9.5 μs, while physical activation reaches 16.3 ms after sixteen old-epoch replays. Across the three models, LSV preserves more latency headroom, compliant goodput, and joint SLO attainment as load rises. The results show that adaptive multiplexing needs a runtime contract for when a resource decision becomes physical, not only a policy for which split to choose.

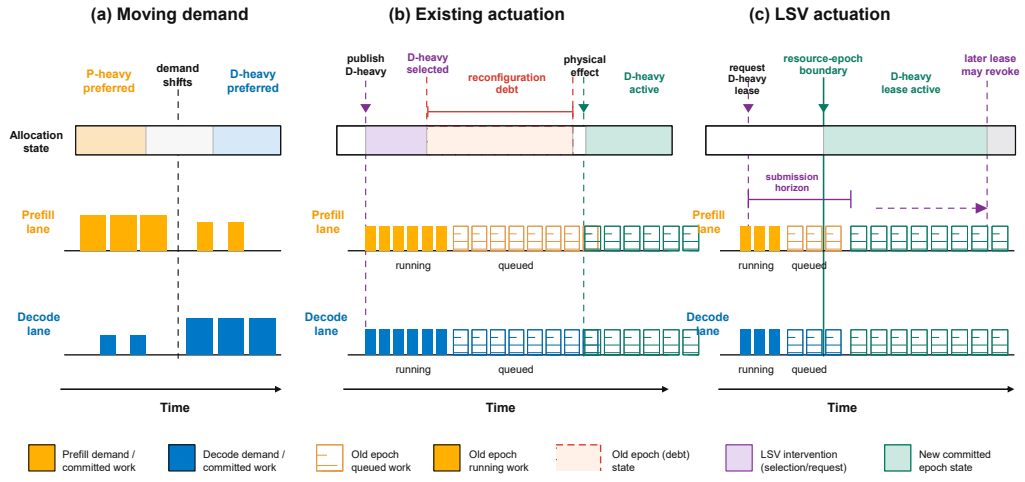


Figure 1: Why live GPU multiplexing needs an execution contract. **(a)** The preferred allocation changes with Prefill and Decode demand. **(b)** A controller can select a new profile while old-epoch work is still running or queued. The red interval is *reconfiguration debt*: software has requested the new profile, but useful work still runs under the old one. **(c)** LSV limits future old-epoch work with a submission horizon. It changes both logical lanes at one resource-epoch boundary. Work after that boundary belongs to the new committed epoch. A later lease can be revoked again.

1 Introduction

LLM serving alternates between two kinds of GPU work. Prefill processes a prompt and benefits from parallel compute. Decode advances active requests one token at a time and is sensitive to per-iteration delay. Recent runtimes separate, chunk, or spatially share these phases to improve

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

latency and throughput (Yu et al., 2022; Kwon et al., 2023; Agrawal et al., 2024; Zhong et al., 2024; Patel et al., 2024; Cui et al., 2025).

The useful Prefill/Decode split changes at runtime. Prompt length, Decode batch size, KV length, and bursty arrivals change the operating point. Figure 1(a) shows one case. Demand shifts from Prefill-heavy to Decode-heavy while requests remain live. A multiplexer must change its physical split online.

Selecting a new split is not the same as making it physical. CUDA submission is asynchronous. The host can change a profile pointer or record an event in microseconds. Earlier kernels and CUDA Graph replays can still remain ahead in the device queues. Those operations still use the old allocation. Figure 1(b) separates the two events. The policy selects Decode-heavy resources first. The physical effect arrives only after old Prefill and Decode work retires.

We call the outstanding old work *reconfiguration debt*. It matters most when response time is tight. During a burst, a controller can correctly decide that Decode needs more SMs. The GPU may still spend several more milliseconds on work already committed to the previous profile. Host-side switching latency then understates the actual control delay.

We present Logical Stream Virtualization (LSV), an execution layer that makes this delay explicit and controllable. Prefill and Decode submit through stable logical lanes. Each lane resolves through a *resource epoch*. The epoch binds the physical stream, SM profile, Decode CUDA Graph row, attention backend, and progress state. LSV separates issued work from completed work. A submission horizon limits future old-epoch commitments. Both lanes then move to the next epoch in one ordered operation. Figure 1(c) shows the contract. A policy requests a lease. The old epoch reaches a safe boundary before the new lease becomes active.

Resource selection stays outside this mechanism. Our prototype uses a small online lease controller. It observes Decode latency and batch size, Prefill pressure, prompt age, and completed progress. It revokes Prefill resources quickly when Decode loses headroom. It expands toward Prefill more slowly when service is stable. The controller needs no offline latency surface. The main contribution is the execution boundary that makes a selected split safely and promptly reversible.

We implement LSV in SGLang with more than 3,000 lines of runtime code (Zheng et al., 2024). We evaluate three models with 35B, 70B, and 235B parameters from the Qwen and Llama families. All experiments run on eight NVLink-connected H100 GPUs. A controlled transition test isolates the mechanism. Host handoff stays near 9.5 μs, while physical activation reaches 16.3 ms after sixteen old-epoch replays. The serving results show the same pattern across model scale and family. In the burst experiment, the new allocation takes effect before the request queue grows as far as in the comparison systems.

Our contributions are:

- We identify *reconfiguration debt*: old issued GPU work between a resource decision and its first physical effect.
- We design LSV with state-bearing logical lanes, debt-budgeted resource epochs, group-consistent transitions, and revocable leases.
- We evaluate steady load, burst response, and model generality across three model scales and two model families on an 8×H100 NVLink server.

2 Why Existing Selection Is Not Enough

2.1 The best split moves with the workload

A spatial profile assigns part of each GPU to Prefill and the rest to Decode. We write P_x/D_y for a profile with x Prefill SMs and y Decode SMs. Our H100 profiles include Prefill-heavy, balanced, Decode-heavy, and single-phase endpoints. The controller uses the same execution path for concurrent and temporal sharing.

The best point depends on current work. Figure 2(a) sweeps the split for different prompt lengths and Decode batch sizes. With 4K-token Prefill and Decode batch 96, the normalized compliant rate peaks near a Decode-heavy P_{36}/D_{96} split. With 16K-token Prefill and Decode batch 32, the useful region

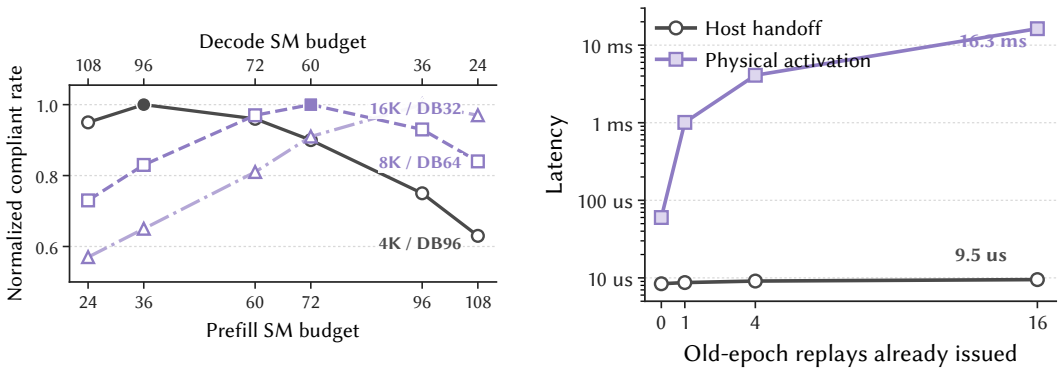


Figure 2: Two parts of the motivation. **Left:** the best static SM split changes with prompt length and Decode batch size. The y-axis is normalized compliant rate. **Right:** a controlled handoff separates software cost from physical effect. Host handoff remains about 9.5 μs. First target-profile activation reaches 16.3 ms when sixteen old-epoch replays are already queued.

moves toward more Prefill SMs. The 8K/DB64 case peaks between them. One fixed split cannot be best for all three states.

2.2 Issued Work Is a Future Commitment

Choosing another profile does not revoke submitted GPU work. Suppose the Prefill lane has issued model-layer quanta through layer $k + 16$. Completion events confirm progress only through layer k. The sixteen issued quanta are future commitments to the current resource epoch. A host-side profile change cannot remove them from the stream.

We make this state explicit. For logical lane ℓ, let $I_\ell(t)$ be issued quanta and $C_\ell(t)$ be quanta whose completion has been observed. If $\widehat{R}_q(t)$ estimates the remaining time of quantum q, we use

$$D_\ell(t) = \sum_{q \in I_\ell(t) \setminus C_\ell(t)} \widehat{R}_q(t) \tag{1}$$

as an estimate of reconfiguration debt. This is an online scheduling estimate, not a hard time bound. A running kernel is non-preemptive. Duration estimates can be wrong, and memory interference can change with the co-runner. LSV therefore also limits the number and estimated duration of outstanding quanta.

Figure 2(b) isolates the effect. With no old replay ahead of a handoff, the target profile activates in tens of microseconds. With one replay, activation is about 1 ms. With four, it is several milliseconds. With sixteen, it is 16.3 ms. The host-side handoff remains near 9.5 μs. The gap comes from work already submitted to the GPU. CUDA events can remove host synchronization, but an event still waits behind submitted work. CUDA Graphs reduce repeated launch overhead, but a queued graph replay is still queued work (NVIDIA Corporation, 2026a).

2.3 Requirements

A live resource change needs four properties. It must preserve all profile-bound state. New work must start after both old phase tails. The runtime must limit work issued before that boundary. It must also retain optimized paths such as CUDA Graph Decode. These properties are coupled. A fast transition cannot overcome a long old queue. A quick mapping change is also unsafe if it exposes stale graph or attention state.

3 Logical Stream Virtualization

Figure 1 gives the design overview. Panel (a) shows why the allocation changes. Orange Prefill work dominates first. After the demand-shift line, Decode work becomes denser and a Decode-heavy

profile is preferred. LSV does not require a particular predictor. It accepts a target resource request from the policy.

3.1 From moving demand to a resource lease

Panel (b) shows the state hidden by a normal profile change. Purple markers are software actions. Solid orange and blue blocks are running old-epoch work. Outlined blocks are queued old-epoch work. Selecting or publishing D-heavy resources does not remove either queue. The red interval ends when the first useful target-profile work can start. Device progress sets its length, not the CPU call.

Panel (c) shows LSV actuation. The policy requests a Decode-heavy *lease*. A submission horizon stops the old Prefill lane from building an unbounded tail. Existing work finishes at legal non-preemptive boundaries. LSV then orders both old lane tails. It crosses one resource-epoch boundary and publishes the new state as a unit. Green blocks belong to the new committed epoch. A later request can revoke the lease and repeat the transition. The contract is simple: *the policy requests an allocation; the execution layer defines when it can safely take effect*.

3.2 State-bearing logical lanes

The serving scheduler submits through two stable identities: a logical Prefill lane L_P and a logical Decode lane L_D. A lane does not map to a bare CUDA stream. It resolves to an immutable executor capsule. The capsule contains the physical stream, SM profile, epoch identifier, and profile-dependent serving state. Decode capsules also bind the matching attention backend and CUDA Graph row.

This packaging prevents partial publication. For example, Decode may move to a stream with a larger SM budget. Replaying a graph captured for another profile at the same time can mix state from two allocations. LSV constructs the target capsule first. It exposes the mapping only when the transition commits. The scheduler-visible lane identity never changes. Request queues, KV ownership, and normal lifecycle state therefore stay live across a resource change.

The same mapping handles spatial and temporal execution. In a multiplexed epoch, both logical lanes resolve to non-empty SM partitions. In a Decode-only epoch, L_D resolves to the full active allocation and L_P is parked. Parking stops GPU submission from that lane. It does not create another scheduler or move the request queue.

3.3 Debt-budgeted resource epochs

A resource epoch is an interval in which both lanes use one consistent allocation. Each split-Prefill launch creates a ticket. The ticket stores its epoch, layer interval, token count, estimated duration, result handle, and completion event. *Issued* progress advances at submission. *Completed* progress advances only after every tensor-parallel rank finishes the head ticket.

Two controls limit how much old work a transition can face. *Quantum size* limits one non-preemptive Prefill unit. The *submission horizon* limits the estimated remaining time of all outstanding tickets plus a new candidate. Submission pauses when the candidate would exceed the debt budget. Both controls are needed. Many small quanta can still form a long queue. One large quantum can set an irreducible delay. LSV adapts quantum size from recent completion time and budgets the remaining queue separately. The estimate can be wrong, so the mechanism is debt-budgeted rather than debt-bounded.

3.4 One transition for both lanes

A profile change must order both phases. Starting new Decode work before old Prefill retires can overlap incompatible SM sets. Changing the lanes independently can also expose a new graph row with an old stream mapping. LSV therefore uses one group transaction. It stops new old-epoch Prefill issue and records Prefill and Decode tail events. Both target streams wait on both tails. LSV then publishes the target profile-bound state, installs both mappings, and increments the epoch. CUDA events carry this dependency without device-wide synchronization. Running work is not preempted (Han et al., 2022). It finishes in the old epoch. Later work resolves through the new capsule.

3.5 Revocable leasing without an offline profile table

The lease controller uses signals already present in the serving loop. These include Decode iteration latency and batch size, queued Prefill tokens, oldest-prompt age, completed Prefill progress, profile tenure, and recent transition outcomes. It needs no offline latency surface. Lease changes are asymmetric. Decode pressure can revoke Prefill resources at the next legal boundary. Expansion waits for headroom and minimum tenure, then moves through adjacent profiles. A prompt-age and progress guard prevents low TPOT from being obtained by starving Prefill. Section 5.6 isolates the debt budget, fast revoke path, Prefill guard, and the boundary between execution mechanism and lease policy.

4 Implementation

Our prototype adds more than 3,000 lines of runtime code to SGLang (Zheng et al., 2024). It provisions Green-Context stream configurations (NVIDIA Corporation, 2026b). It wraps profile-bound execution state in immutable capsules, tracks Prefill tickets and resource epochs, and runs the online lease controller. Admission, continuous batching, KV ownership, sampling, and the request lifecycle remain in SGLang. Decode keeps profile-specific CUDA Graph replay (NVIDIA Corporation, 2026a). Split Prefill exposes bounded layer quanta and records issue, completion, epoch, and first-use events.

All three model experiments use tensor parallelism across eight NVLink-connected H100 GPUs (Shoeybi et al., 2019). Rank 0 selects the profile. Every rank performs the same epoch transition. Prefill progress commits only after the group completes the head ticket. The slowest completion signal updates the horizon. This prevents ranks from entering different resource epochs.

5 Evaluation

We evaluate four questions. Does LSV preserve the serving region as load rises? Does it react before a burst grows the queue? Does the effect hold across model scale and family? Does it depend on the protections described above? Appendices A–C add direct transition measurements, controller sensitivity, and formal execution properties.

5.1 Setup

Hardware and models. All end-to-end experiments use one 8×H100 NVLink server. Tensor parallelism spans all eight GPUs. We evaluate Qwen3.6-35B-A3B, Llama 3.1-70B, and Qwen3-235B-A22B. The set covers three model scales, dense and MoE architectures, and two model families (Qwen Team, 2026; Grattafiori et al., 2024; Qwen Team, 2025b;a).

Workloads, baselines, and metrics. We use Azure Conversation, Kimi, and LooGLE Summary. Azure-style traces contain burst and length variation (Wang et al., 2024). LooGLE supplies long-context summarization workloads (Li et al., 2024). At each point, all systems receive the same request sequence. We compare with PDMux, Bullet, SGLang, and vLLM using the supplied experiment configurations. We report mean TPOT, mean TTFT, request throughput, token goodput, and joint SLO attainment. Goodput counts tokens that meet the per-token target. Joint attainment requires both prompt and token-latency objectives. Model, TP width, request sequence, and memory configuration stay fixed within each comparison.

5.2 Qwen3.6-35B-A3B: steady load

Claim. If slow physical actuation is the main problem, LSV should keep more latency headroom as load rises.

Figure 3 reports the steady-rate sweep for Qwen3.6-35B-A3B. The rows are Azure Conversation, Kimi, and LooGLE Summary. The columns are mean TPOT, compliant token goodput, and joint SLO attainment. The same pattern appears in all three workloads. At high request rates, LSV keeps mean TPOT below the comparison curves. It also has higher compliant goodput. Its joint attainment stays between about 0.95 and 1.0 across the plotted range.

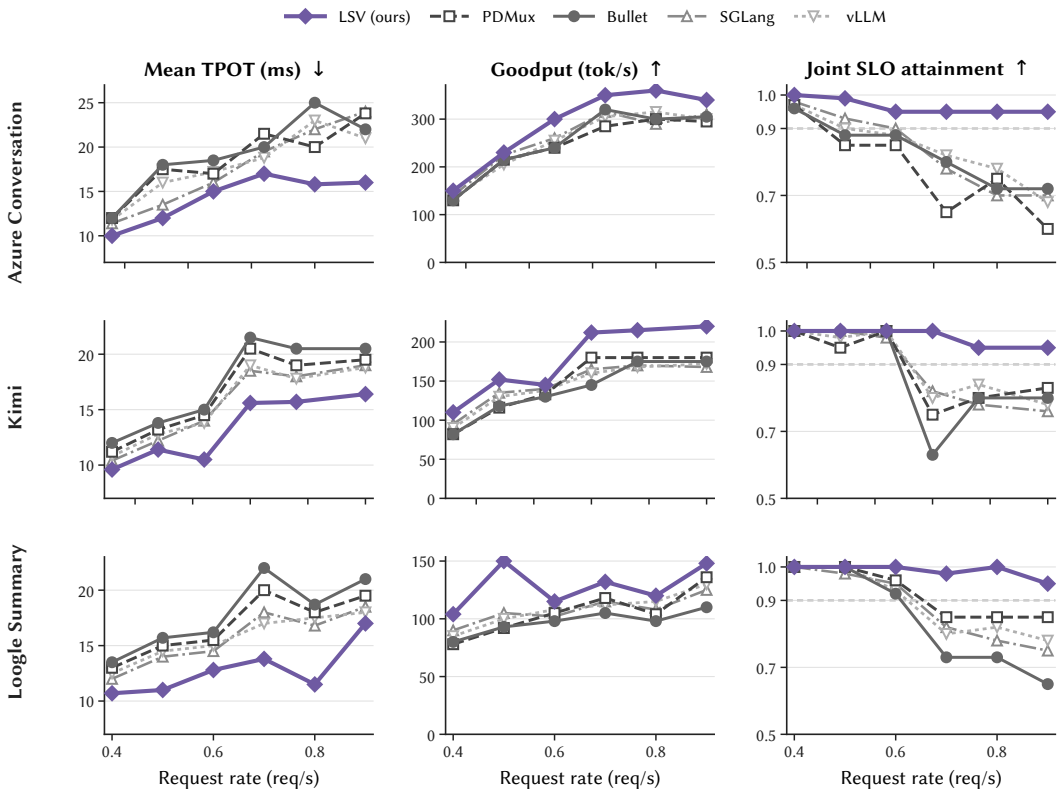


Figure 3: Steady-rate serving with Qwen3.6-35B-A3B on 8×H100 NVLink. Each row is a workload. LSV keeps mean TPOT lower at high load, sustains higher compliant goodput, and holds joint SLO attainment near 0.95–1.0 as the comparison systems lose headroom.

At the final point of each row, LSV stays near 0.95 joint attainment. The comparison systems end around 0.60–0.85. The gap is small at low load and grows near saturation, where late actuation matters. The goodput curves show the same pattern. LSV has lower TPOT and higher compliant output at high load. This is consistent with a wider feasible operating region rather than a fixed per-forward speedup.

5.3 Qwen3.6-35B-A3B: burst response

Claim. A responsive mechanism should change the physical Prefill/Decode balance during a burst. The queue should grow less than it does with a delayed transition.

Figure 4 aligns three signals on one timeline. The top panel shows the Prefill SM allocation chosen by LSV. Before the burst, the controller explores several profiles. At burst onset, it holds a Prefill-heavy allocation. When Decode pressure rises, it quickly revokes Prefill resources down to 36 SMs. It later expands through 72 and 96 SMs as the workload stabilizes. These changes happen at resource-epoch boundaries. They do not interrupt a running kernel.

The middle panel shows the conflict. Queued Prefill tokens and Decode batch size rise together. The controller must keep prompts moving while protecting active generation. Either static extreme sacrifices one side.

The bottom panel shows the outcome. LSV peaks below 9 queued requests. The largest comparison spike exceeds 20, and the other baselines lie between. The LSV queue also drains earlier. The 96-to-36 SM revoke helps only if old Prefill commitments are short enough for the new epoch to start during the burst. Debt budgeting provides that condition. Later adjacent expansions return Prefill capacity after Decode regains headroom.

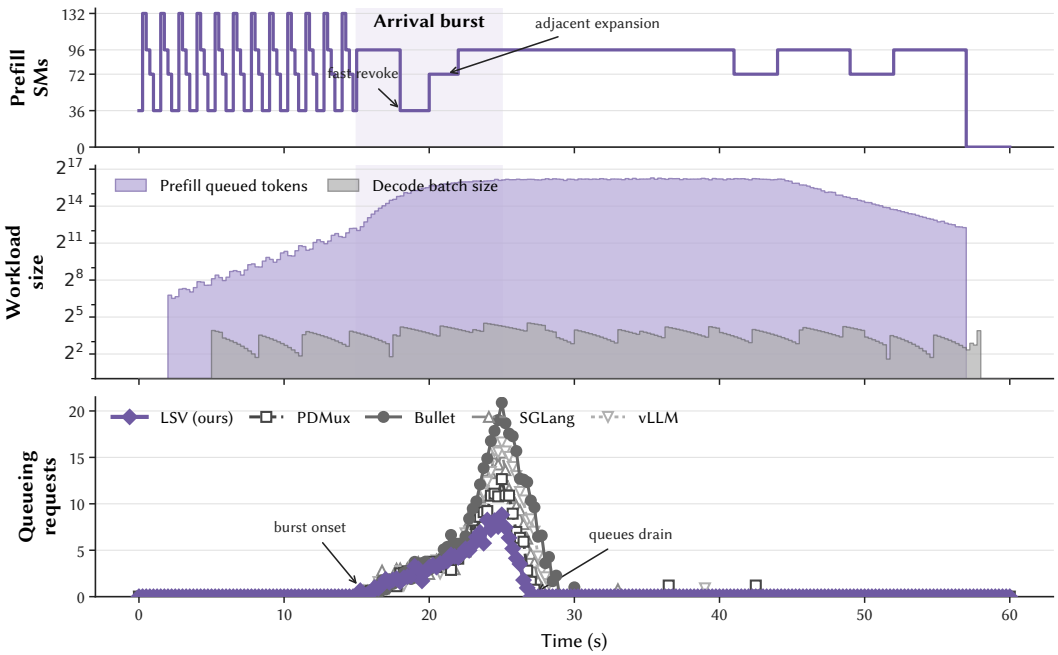


Figure 4: Burst response with Qwen3.6-35B-A3B on 8×H100 NVLink. Top: LSV revokes Prefill SMs as Decode pressure rises and expands again after the burst. Middle: queued Prefill tokens and Decode batch size. Bottom: LSV has the smallest peak request queue among the plotted systems and drains the queue earlier.

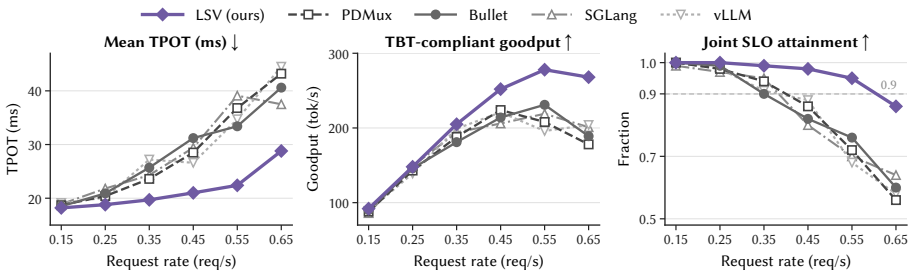


Figure 5: Kimi trace on Llama 3.1-70B with 8×H100 NVLink. As load rises, LSV keeps lower mean TPOT, higher TBT-compliant goodput, and higher joint SLO attainment than the four comparison systems.

5.4 Llama 3.1-70B: generality across model families

Claim. If the benefit comes from reversible actuation, it should also appear on a dense Llama model at an intermediate scale.

Figure 5 shows the Llama 3.1-70B sweep (Grattafiori et al., 2024). At low load, all systems are close because the GPU has spare capacity. The curves separate as the request rate rises. At the final point, LSV remains below 30 ms mean TPOT. The comparison systems are near or above 40 ms. LSV compliant goodput remains above 250 tokens/s, while the baselines are around 180–205 tokens/s. Joint SLO attainment follows the same pattern. LSV stays near 0.86 at the highest rate, while the comparison systems fall to roughly 0.55–0.63. Llama 3.1-70B is a different model family and a dense model. The same resource-epoch mechanism remains useful on this model.

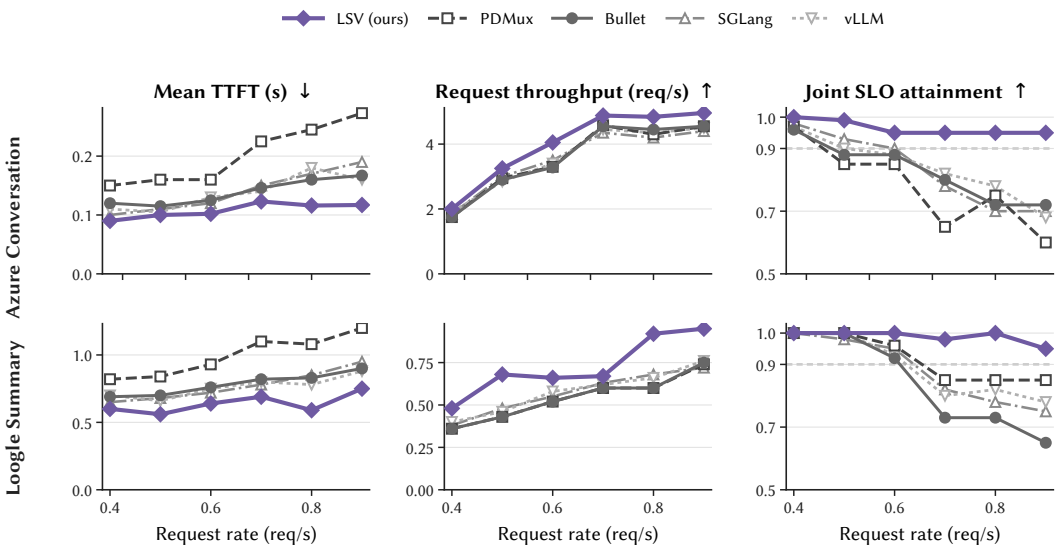


Figure 6: End-to-end serving with Qwen3-235B-A22B on 8×H100 NVLink. LSV keeps lower mean TTFT, reaches the highest request throughput at the high-rate points, and maintains about 0.95 joint SLO attainment in both workloads.

5.5 QWEN3-235B-A22B: SCALING TO THE LARGEST MODEL

Claim. The same execution contract should remain useful as model scale and tensor-parallel work increase.

Figure 6 repeats the end-to-end measurements with Qwen3-235B-A22B. Across Azure Conversation and LooGLE Summary, LSV keeps mean TTFT flatter as rate rises. It reaches the highest request throughput at the upper-rate points and keeps joint SLO attainment near 0.95. The last comparison points fall to about 0.60–0.72 on Azure and 0.65–0.85 on LooGLE. This 235B result also exercises the group boundary. All eight TP ranks must commit the same epoch before any rank activates the new profile.

5.6 ABLATION: WHERE DOES THE GAIN COME FROM?

Claim. Fast profile selection is not enough by itself. The runtime also needs a safe epoch transition, a limit on old commitments, and a policy that protects Decode urgency and Prefill progress.

Figure 7 tests this claim in two directions. Panel (a) starts from PDMux and adds one part at a time. Matching the profile set changes little. This makes a larger choice of SM splits an unlikely main explanation. Adding the epoch transaction gives another small step. A safe group boundary is necessary, but it does not shorten a long queue that was already issued. The larger step appears after debt budgeting. Full LSV reaches about 1.27–1.34× the normalized TBT-compliant rate across Azure, Kimi, and LooGLE. This ordering matches the microbenchmark in Figure 2(b). The costly part of a live change is work committed before the change, not the host handoff.

Panel (b) removes one protection from the complete system. Replacing the lease policy with the PDMux policy on the same LSV executor lowers normalized joint-SLA attainment to roughly 0.86. Execution support alone does not choose the right operating point. Removing fast revoke lowers attainment further. This shows the cost of waiting when Decode pressure rises. Removing the debt budget is worse. The controller may request the right profile but still leave enough old work ahead of the boundary to make the request late. Removing the Prefill age guard gives the largest drop. A controller can otherwise obtain attractive token latency by withholding prompt service. These failures are different. Fast revoke controls *when to leave* a Prefill-heavy lease. Debt budgeting controls *how soon that decision can become physical*. The age guard controls *when Prefill must receive service again*.

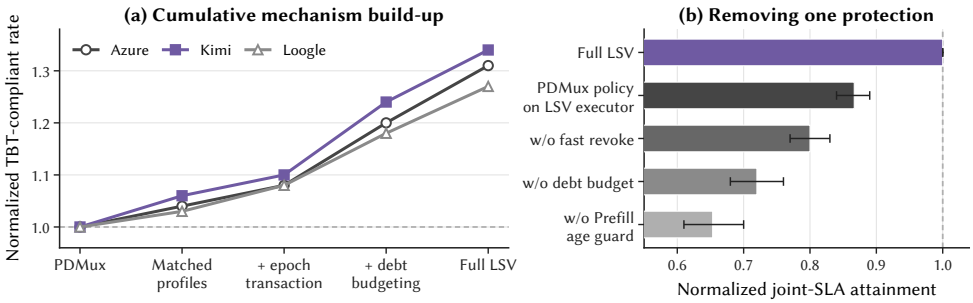


Figure 7: Ablation of LSV. **(a)** Cumulative build-up from PDMux: debt budgeting gives the largest mechanism-level step. **(b)** Removing controller protections lowers normalized joint-SLA attainment.

The panels separate mechanism from policy. Epoch transactions provide correctness. Debt budgeting provides responsiveness. Revocable leasing uses that responsiveness without starving either phase.

6 Related Work

LLM serving work covers iteration scheduling, KV-cache management, preemption, placement, and multi-tenant execution (Yu et al., 2022; Kwon et al., 2023; Zheng et al., 2024; Wu et al., 2026; Sun et al., 2024; Abhyankar et al., 2024; Zhang et al., 2026; Li et al., 2023; Sheng et al., 2023; Chen et al., 2024; Sheng et al., 2024). Other systems focus on phase execution. Some chunk Prefill or separate Prefill/Decode execution and KV movement (Agrawal et al., 2024; Zhong et al., 2024; Patel et al., 2024; Qin et al., 2024). PDMux, Bullet, Nexus, DuetServe, and RAPID-Serve adapt intra-GPU phase allocation (Cui et al., 2025; Lin et al., 2025; Shi et al., 2025; Gao et al., 2025; Masood et al., 2026). LSV targets a different interval: the time between allocation selection and physical effect. CUDA Graphs and Green Contexts provide the underlying primitives (NVIDIA Corporation, 2026a;b). Kernel, model-parallel, and GPU-sharing work is complementary (Dao et al., 2022; Dao, 2024; Shoeybi et al., 2019; Xiao et al., 2018; Yu & Chowdhury, 2020; Han et al., 2022).

7 Discussion

Reconfiguration debt is an execution property, not host API overhead. A resource call can return while old work remains ahead of the target profile. Draining at every change would stall Decode. LSV instead orders both target streams after the old tails and limits future commitments. Logical lanes bind profile-specific state. The lease controller remains replaceable. The ablation shows why both layers matter. Policy selects the operating point. Limited old work lets that decision become physical in time.

8 Limitations

LSV does not create capacity. Its debt estimate is not a formal bound. Running kernels remain non-preemptive. We evaluate one 8×H100 NVLink topology, three models from two families, SGLang, and a finite profile set. Other GPUs, engines, and topologies require validation.

9 Conclusion

LSV makes adaptive Prefill/Decode allocation promptly reversible. It exposes reconfiguration debt, limits future old-epoch commitments, and switches state-bearing lanes at one resource-epoch boundary. Across three model scales and two families on eight H100 GPUs, LSV preserves SLO headroom under load and reduces burst queue growth.

Reproducibility Statement

The artifact contains the SGLang integration, experiment launchers, configuration files, trace processing, transition instrumentation, and figure-generation scripts. The three evaluated models use eight H100 GPUs with NVLink. Appendix A reports direct transition-latency and debt-prediction evidence. Appendix B reports controller sensitivity. Appendix C states the execution invariants and the assumptions behind the conditional activation-delay result.

AI Use Statement

AI tools were used for literature search, citation checking, manuscript organization, language editing, figure presentation, feedback on experimental presentation, and the formulation and proofreading of the mathematical statements and proof sketches in Appendix C. They were not used to generate experimental measurements or replace measured values. The authors checked the AI-assisted mathematical statements against the execution contract in the paper. They also reviewed the technical claims and citations and take responsibility for the final content.

References

Reyna Abhyankar, Zijian He, Vikranth Srivatsa, Hao Zhang, and Yiying Zhang. Infercept: Efficient intercept support for augmented large language model inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 81–95, 2024.

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming throughput-latency tradeoff in llm inference with sarathi-serve. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 117–134. USENIX Association, 2024.

Lequn Chen, Zihao Ye, Yongji Wu, Danyang Zhuo, Luis Ceze, and Arvind Krishnamurthy. Punica: Multi-tenant lora serving. In *Proceedings of Machine Learning and Systems*, volume 6, 2024.

Weihao Cui, Yukang Chen, Xiaoze Fan, Han Zhao, Ziyi Xu, Xusheng Chen, Bingsheng He, and Quan Chen. Pd-multiplexing: Unlocking high-goodput llm serving with greencontext. LMSYS Org Blog, 2025. URL `https://www.lmsys.org/blog/2025-09-28-pdmux/`.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16344–16359, 2022.

Lei Gao, Chaoyi Jiang, Hossein Entezari Zarch, Daniel Wong, and Murali Annavaram. Duetserve: Harmonizing prefill and decode for llm serving via adaptive gpu multiplexing. *arXiv preprint arXiv:2511.04791*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Mingcong Han, Hanze Zhang, Rong Chen, and Haibo Chen. Microsecond-scale preemption for concurrent gpu-accelerated dnn inferences. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 539–558. USENIX Association, 2022.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165.

Jiaqi Li, Mengmeng Wang, Zilong Zheng, and Muhan Zhang. Loogle: Can long-context language models understand long contexts? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16304–16333, 2024. doi: 10.18653/v1/2024.acl-long.859.

Zhuohan Li, Lianmin Zheng, Yinmin Zhong, Vincent Liu, Ying Sheng, Xin Jin, Yanping Huang, Zhifeng Chen, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Alpaserve: Statistical multiplexing with model parallelism for deep learning serving. In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 663–679. USENIX Association, 2023.

Zejia Lin, Hongxin Xu, Guanyi Chen, Xianwei Zhang, and Yutong Lu. Bullet: Boosting gpu utilization for llm serving via dynamic spatial-temporal orchestration. *arXiv preprint arXiv:2504.19516*, 2025.

Amna Masood, Pratishtha Gaur, and Nuwan Jayasena. Rapid-serve: Resource-efficient and accelerated p/d intra-gpu disaggregation. *arXiv preprint arXiv:2601.11822*, 2026.

NVIDIA Corporation. Cuda graphs. CUDA Programming Guide, 2026a. URL https://docs.nvidia.com/cuda/cuda-programming-guide/04-special-topics/cuda-graphs.html.

NVIDIA Corporation. Green contexts. CUDA Programming Guide, 2026b. URL https://docs.nvidia.com/cuda/cuda-programming-guide/04-special-topics/green-contexts.html.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Inigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative llm inference using phase splitting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pp. 118–132, 2024. doi: 10.1109/ISCA59077.2024.00019.

Ruoyu Qin, Zheming Li, Weiran He, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. Mooncake: A kvcache-centric disaggregated architecture for llm serving. *arXiv preprint arXiv:2407.00079*, 2024.

Qwen Team. Qwen3-235b-a22b model card. Hugging Face, 2025a. URL https://huggingface.co/Qwen/Qwen3-235B-A22B.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025b.

Qwen Team. Qwen3.6-35b-a3b model card. Hugging Face, 2026. URL https://huggingface.co/Qwen/Qwen3.6-35B-A3B.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Re, Ion Stoica, and Ce Zhang. Flexgen: High-throughput generative inference of large language models with a single gpu. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31094–31116, 2023.

Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, Joseph E. Gonzalez, and Ion Stoica. Slora: Scalable serving of thousands of lora adapters. In *Proceedings of Machine Learning and Systems*, volume 6, pp. 296–311, 2024.

Xiaoxiang Shi, Colin Cai, Junjia Du, and Zhihao Jia. Nexus: Proactive intra-gpu disaggregation of prefill and decode in llm serving. *arXiv preprint arXiv:2507.06608*, 2025.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Biao Sun, Ziming Huang, Hanyu Zhao, Wencong Xiao, Xinyi Zhang, Yong Li, and Wei Lin. Llumnix: Dynamic scheduling for large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 173–191. USENIX Association, 2024.

Yuxin Wang, Yuhan Chen, Zeyu Li, Xueze Kang, Zhenheng Tang, Xin He, Rui Guo, Xin Wang, Qiang Wang, Amelie Chi Zhou, and Xiaowen Chu. Burstgpt: A real-world workload dataset to optimize llm serving systems. *arXiv preprint arXiv:2401.17644*, 2024.

Bingyang Wu, Yinmin Zhong, Zili Zhang, Shengyu Liu, Fangyue Liu, Yuanhang Sun, Gang Huang, Xuanzhe Liu, and Xin Jin. Fastserve: Iteration-level preemptive scheduling for large language model inference. In *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 57–74. USENIX Association, 2026.

Wencong Xiao, Romil Bhardwaj, Ramachandran Ramjee, Muthian Sivathanu, Nipun Kwatra, Zhenhua Han, Pratyush Patel, Xuan Peng, Hanyu Zhao, Quanlu Zhang, Fan Yang, and Lidong Zhou. Gandiva: Introspective cluster scheduling for deep learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 595–610. USENIX Association, 2018.

Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A distributed serving system for transformer-based generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 521–538. USENIX Association, 2022.

Peifeng Yu and Mosharaf Chowdhury. Salus: Fine-grained gpu sharing primitives for deep learning applications. In *Proceedings of Machine Learning and Systems*, volume 2, 2020.

Dingyan Zhang, Jinbo Han, Kaixi Zhang, Xingda Wei, Sijie Shen, Chenguang Fang, Wenyuan Yu, Jingren Zhou, and Rong Chen. Simple is better: Multiplication may be all you need for llm request scheduling. In *20th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 55–73. USENIX Association, 2026.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. Sglang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 62557–62583, 2024. doi: 10.52202/079017-2000.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. Distserve: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 193–210. USENIX Association, 2024.

A Direct evidence for reconfiguration debt

Figure 8 links the controlled replay experiment in Figure 2(b) to the serving results. Panel (a) measures the time from a lease request to the first useful work under the requested allocation. The LSV distribution is left-shifted relative to PDMux and Bullet. This shows that the execution contract changes when a resource decision becomes physical, not only the host-side handoff cost. Panel (b) compares the online estimate of outstanding debt with the observed activation delay. The estimate tracks the physical delay in log space ($r = 0.96$), with a median absolute error of 0.41 ms. The debt signal is useful for online control in these measurements. It is not a certified upper bound. Running kernels remain non-preemptive, and interference can change duration estimates.

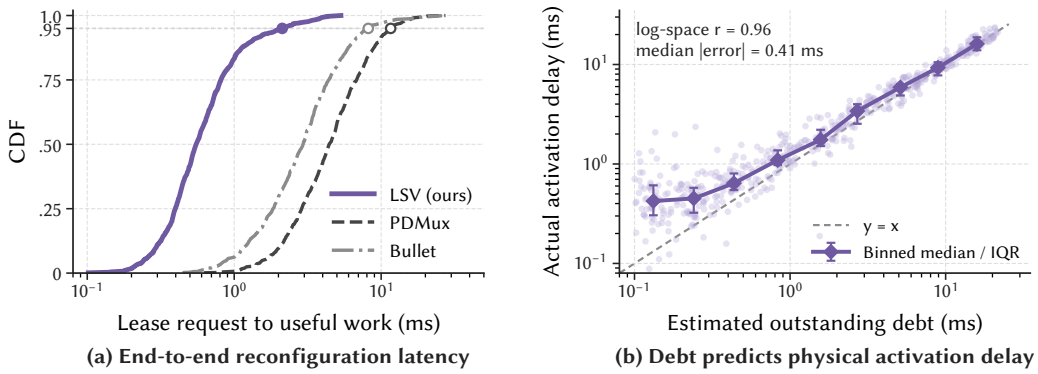


Figure 8: Direct evidence for the mechanism. **(a)** End-to-end reconfiguration latency from lease request to the first useful target-allocation work. LSV activates the requested allocation earlier than PDMux and Bullet over the measured distribution. **(b)** Estimated outstanding debt versus actual physical activation delay. The plotted measurements have log-space correlation $r = 0.96$ and median absolute error 0.41 ms. Binned medians and IQRs follow the $y = x$ trend over most of the range.

B Controller sensitivity

Figure 9 varies one controller parameter at a time around the default configuration. The default lies near the best region on Azure, Kimi, and the burst workload. Moderate changes keep normalized joint-SLA rate close to the default. Larger debt budgets or longer minimum tenure reduce responsiveness, especially under the burst. The Decode revoke guard shows the same trade-off. A modest increase stays close to the default. A larger guard delays revocation and lowers attainment. These results suggest that the gain does not depend on one narrow threshold setting. They also show the cost of too much old work or a stale lease.

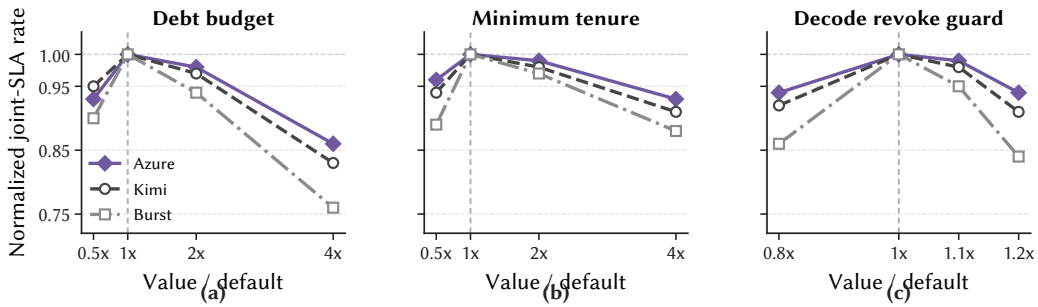


Figure 9: Controller sensitivity with one parameter varied at a time and all others held at their default values. The y-axis is normalized joint-SLA rate. **(a)** Debt budget. **(b)** Minimum profile tenure. **(c)** Decode revoke guard. The default setting is marked by the vertical dashed line.

C FORMAL EXECUTION PROPERTIES

LSV is a systems mechanism. Its correctness does not require a predictive model of workload dynamics. This section states the runtime contract directly. We separate safety properties from responsiveness. The safety properties follow from event ordering. The responsiveness result depends on the quality of the online debt estimate.

C.1 EXECUTION MODEL AND NOTATION

Let $\mathcal{L} = \{P, D\}$ denote the logical Prefill and Decode lanes. A resource epoch e assigns each lane $\ell \in \mathcal{L}$ an immutable executor capsule X_ℓ^e. The capsule contains its physical stream, SM profile, epoch identifier, and profile-dependent serving state. For a transition $e \to e+1$, let τ_ℓ^e be the tail event recorded after the final old-epoch submission on lane ℓ. The target streams for both lanes wait on both tail events. The mapping to X_P^{e+1}, X_D^{e+1} is published after these waits are installed. We call this ordered point the *group boundary*.

For Prefill, $I_P(t)$ and $C_P(t)$ are the issued and completed ticket sets from Equation 1. A ticket enters I_P when it is submitted. It enters C_P only after the completion event has been observed on every tensor-parallel rank. For a candidate Prefill ticket q, the submission horizon admits the ticket only if

$$D_P(t) + \widehat{R}_q(t) \le B_P, \tag{2}$$

where B_P is the configured debt budget. The horizon therefore limits future old-epoch commitments at each admission decision. Proposition C.3 gives the activation result, conditional on the debt value observed when a transition is requested.

Proposition C.1 (Epoch-consistent publication)**.** *Assume that (i) no new old-epoch submission is issued after the corresponding tail event τ_ℓ^e is recorded, (ii) each target stream waits on both τ_P^e and τ_D^e, and (iii) the scheduler-visible lane mappings are published only after these waits are installed. Then, on the two LSV lanes, no work that resolves through epoch $e+1$ can execute concurrently with unfinished work from epoch e, and every post-publication submission observes a mutually consistent pair of target capsules.*

Proof. By CUDA stream order, τ_ℓ^e comes after every old-epoch operation already submitted on lane ℓ. Each target stream waits on both tail events. So target-stream work cannot begin before both old tails complete. Assumption (i) prevents new old-epoch work from being appended after a recorded tail. Assumption (iii) changes the scheduler-visible mapping only after the target dependencies are installed. A post-publication submission therefore resolves through X_ℓ^{e+1}, not a mix of old and new profile-bound state. Old and new epochs do not overlap across the group boundary. Publication is group-consistent. □

Proposition C.2 (Monotone, exactly-once Prefill progress)**.** *Under the ticket discipline above, completed Prefill progress never exceeds issued progress, and an epoch transition neither drops nor duplicates an issued Prefill quantum.*

Proof. A Prefill quantum contributes to issued progress exactly once, at ticket creation. It contributes to completed progress only after every tensor-parallel rank has observed the head ticket's completion. Hence $C_P(t) \subseteq I_P(t)$ at all times. A transition does not delete old tickets or reissue them in the target epoch. Old tickets stay with epoch e until their recorded completion events fire. Later submissions resolve through epoch $e+1$. Each issued quantum is therefore committed exactly once, and completion advances monotonically. □

Proposition C.3 (Conditional activation-delay bound)**.** *Consider a transition request at time t_r. Let R_P be the actual wall-clock time from t_r until the old Prefill tail τ_P^e completes, R_D the corresponding time for the Decode tail, and δ the residual boundary-to-first-use overhead after both tails are satisfied, including host/event publication and target dispatch. Suppose the debt estimator satisfies, for this transition,*

$$R_P \le \kappa D_P(t_r) + \varepsilon \tag{3}$$

for some calibration factors $\kappa \ge 1$ and $\varepsilon \ge 0$. If the submission horizon gives $D_P(t_r) \le B_P$, then the first target-epoch work can activate no later than

$$T_{\text{act}} - t_r \le \max\{\kappa B_P + \varepsilon,\ R_D\} + \delta. \tag{4}$$

Proof. By Proposition C.1, target-epoch work waits for both old lane tails. Activation therefore occurs after the slower tail, plus the residual first-use overhead δ. This gives $T_{\text{act}} - t_r \leq \max\{R_P, R_D\} + \delta$. Substituting Equation 3 and $D_P(t_r) \leq B_P$ gives Equation 4. □

Interpretation. Equation 4 states what the mechanism can and cannot control. Reducing B_P shrinks the controllable Prefill part of transition delay. It cannot preempt a running kernel. It cannot remove the already-recorded Decode tail or residual publication/dispatch overhead. Appendix B shows the same trade-off. A very large debt budget permits more look-ahead but leaves a longer old-epoch tail when a lease changes.

Why this is not a real-time guarantee. The constants κ and ε in Proposition C.3 are not certified bounds. Kernel duration can change with co-runner interference. A running kernel is also non-preemptive. Figure 8(b) provides empirical calibration evidence, not a formal worst-case guarantee. The main text therefore describes LSV as *debt-budgeted* rather than debt-bounded.