

Chunked Prefill & LLM Inference Scheduling

Co-Scheduling Prefill and Decode Phases, Eliminating ITL Jitter, and Maximizing MFU

EXECUTIVE ABSTRACT & SUMMARY FINDINGS: In production Large Language Model (LLM) serving, prompt processing (prefill) and token generation (decode) exhibit fundamentally divergent microarchitectural characteristics. Prefills are compute-bound and monopolize GPU Tensor Cores, while decodes are memory-bandwidth-bound. Co-scheduling these workloads naively creates massive Inter-Token Latency (ITL) spikes. This handbook audits Chunked Prefill algorithms, piggyback iteration batching, KV-cache fragmentation mechanics, and serving throughput optimizations in vLLM and TensorRT-LLM.

Scheduling Engine	Decode Jitter (p99)	Prefill Compute MFU	GPU VRAM Overhead
Continuous Batching	1,200ms - 2,400ms	32% - 45%	Baseline (High)
Decoupled Prefill/Decode	15ms - 25ms	65% - 78%	2x Network Transfer
Chunked Prefill (C=512)	22ms - 38ms	72% - 84%	Single Instance (Optimal)
Chunked + PagedAttention	18ms - 28ms	78% - 88%	Zero Internal Frag

1. The Microarchitectural Conflict Between Prefill and Decode

Production LLM serving engines execute two distinct computational phases: (1) The Prefill phase processes the input prompt in parallel across all tokens, generating initial key-value (KV) states. Because matrix dimensions are large, prefill is compute-bound, achieving 60-80% Model FLOPs Utilization (MFU) on modern Tensor Cores.

(2) The Decode phase generates one token per iteration sequentially. The computation reduces to a Matrix-Vector Multiplication (GEMV), which is severely memory-bandwidth-bound, achieving less than 15% MFU as hundreds of gigabytes of KV-cache are read from High Bandwidth Memory (HBM) to compute a single output token.

In traditional Continuous Batching (iteration-level scheduling), when a long 8,000-token prompt enters the batch, the scheduler allocates the entire GPU execution cycle to that prefill. Consequently, all ongoing user sessions waiting for their next decode token are stalled for 800ms to 2,000ms, causing intolerable tail latency jitter.

2. Chunked Prefill Mechanics and Budgeted Piggybacking

Chunked Prefill (first popularized by Sarathi-Serve and integrated into vLLM and TensorRT-LLM) eliminates iteration stalls by breaking long prompts into uniform chunks of fixed length (e.g., C = 512 or 1024 tokens).

Rather than evaluating an entire prompt in one massive pass, the serving engine processes one chunk per iteration while concurrently executing decode tokens from all active streams. Because the compute-intensive chunk saturates the GPU's Tensor Cores, while the decode tokens utilize the remaining memory bandwidth, the GPU hardware is simultaneously utilized across compute and memory subsystems, boosting overall system throughput by 2.2x.

3. Attention Masking and Intermediate KV-Cache Management

Chunking a prompt breaks the self-attention causal triangular matrix into separate sub-matrices: a full cross-attention block for previously cached tokens and a causal triangular block for the current chunk. Serving engines implement specialized CUDA/Triton fused attention kernels that compute: $Q_chunk * K_cached^T$ and $Q_chunk * K_chunk^T$ in a single fused pass, preventing round-trip memory spills back to HBM.

Combining Chunked Prefill with PagedAttention ensures that the incremental KV-cache generated by each chunk is allocated into non-contiguous virtual memory blocks, eliminating internal fragmentation and memory reservations for uncomputed prompt tokens.

4. Tuning the Chunk Budget for Optimal TTFT vs. ITL

Sizing the chunk parameter C involves a direct Pareto tradeoff between Time-To-First-Token (TTFT) and Inter-Token Latency (ITL). Setting C too small (e.g., $C=128$) increases kernel launch overhead and lengthens TTFT for long prompts. Setting C too large ($C=2048$) reintroduces decode latency jitter. Empirical telemetry on NVIDIA H100 clusters demonstrates that a dynamic chunk budget of $C = 512$ to 1024 achieves the optimal balance, holding p99 decode jitter below 30ms.

CRITICAL ENGINEERING TAKEAWAYS & DEPLOYMENT DIRECTIVES:

1. Continuous batching naively stalls active decode streams when processing large prompt prefills.
2. Chunked Prefill divides prompts into discrete budgets ($C=512$), co-scheduling prefill and decode tasks in the same step.
3. Fused compute-and-memory scheduling boosts GPU Model FLOPs Utilization (MFU) from 45% to over 80%.
4. Combining chunking with PagedAttention eliminates internal memory fragmentation and guarantees stable p99 ITL.