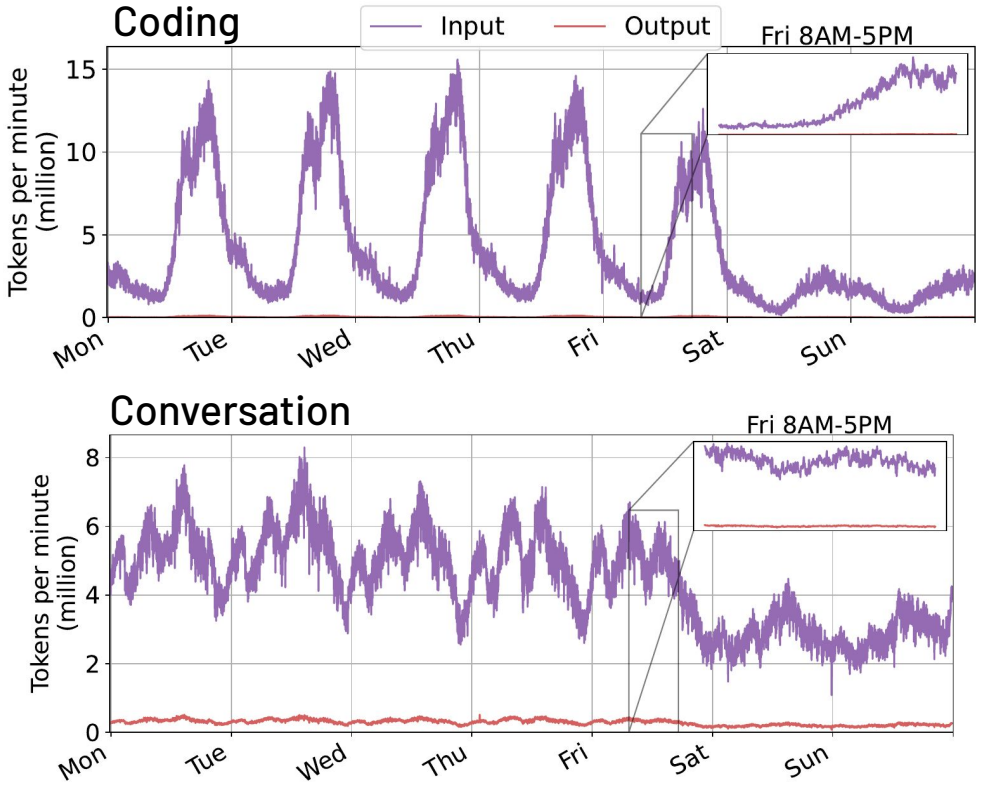
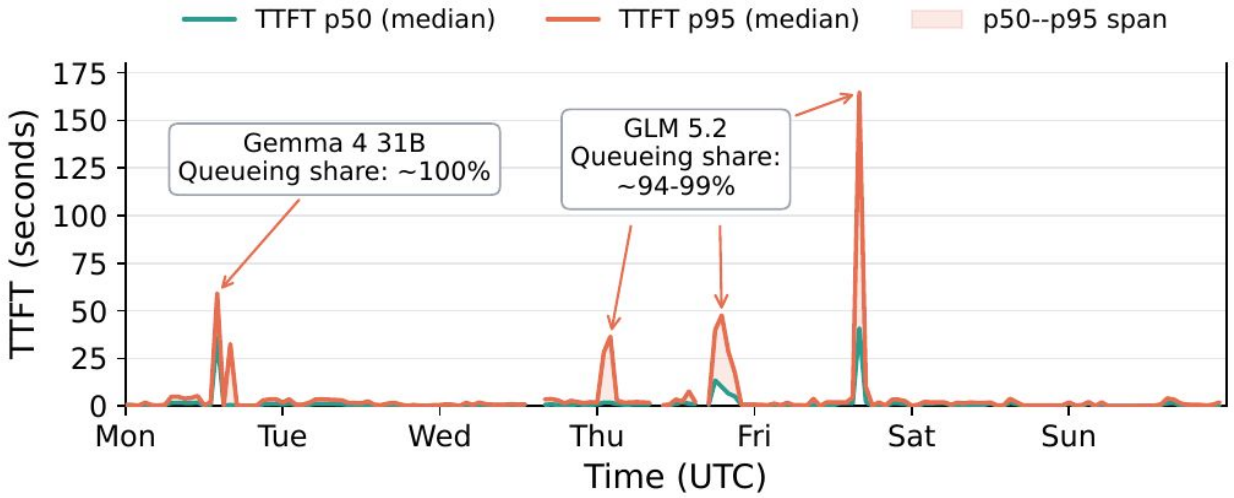
Parallel Restore-Optimized Snapshot Compression for Cost-Efficient Inference Serving

Radostin Stoyanov – PhD Student, Scientific Computing
Collaboration with Fergus Finn, Dan Feigin, and Andrei Vagin
Supervisors: Prof. Rodrigo Bruno, Prof. Wes Armour

Elastic Scaling for LLM Inference



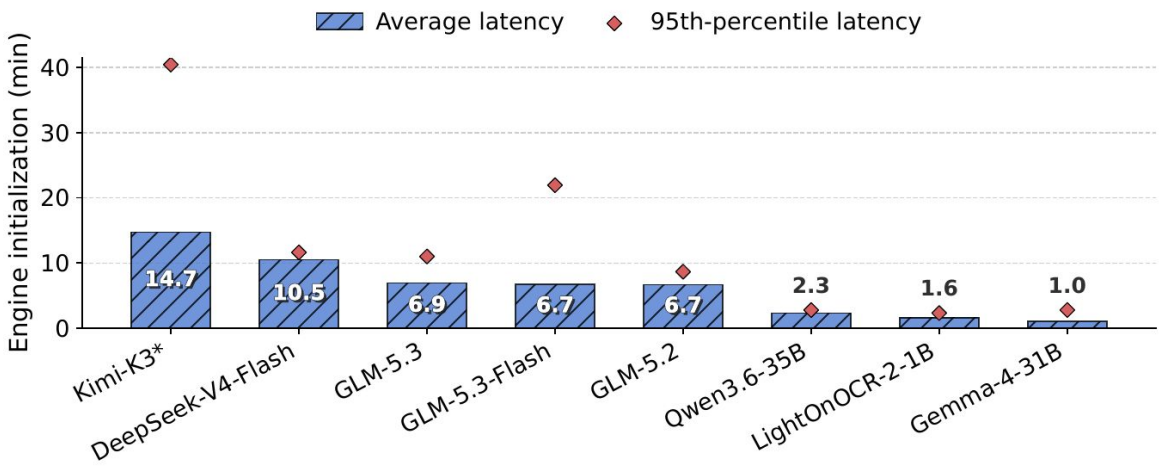
Azure LLM token usage patterns across coding and conversational workloads.



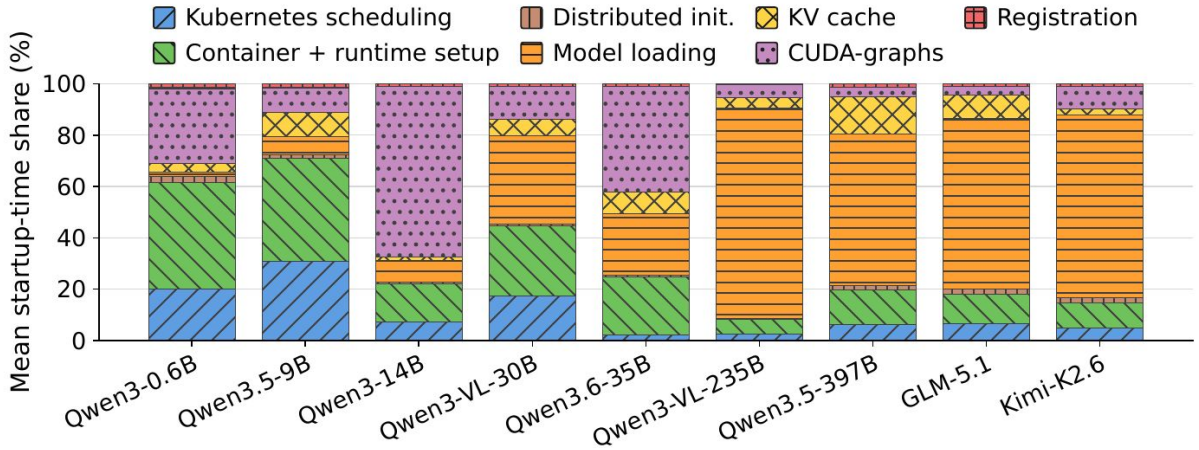
Production trace across two 8×B200 Kubernetes nodes and 26 model configurations, showing hourly p50/p95 TTFT and queueing contributions.

"Engine-Agnostic Model Hot-Swapping for Cost-Effective LLM Inference". CANOPIE-HPC 2025
"Transparent and Unified CPU-GPU Snapshots for LLM Inference Platforms". Middleware 2026

Elastic Scaling for LLM Inference



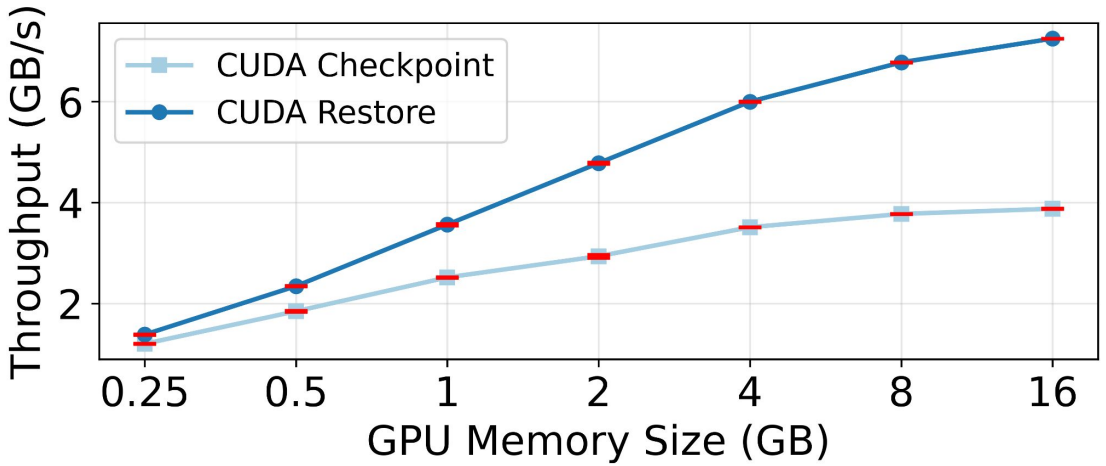
Cold-start latency over one week showing mean and p95 values across SGLang and vLLM*.



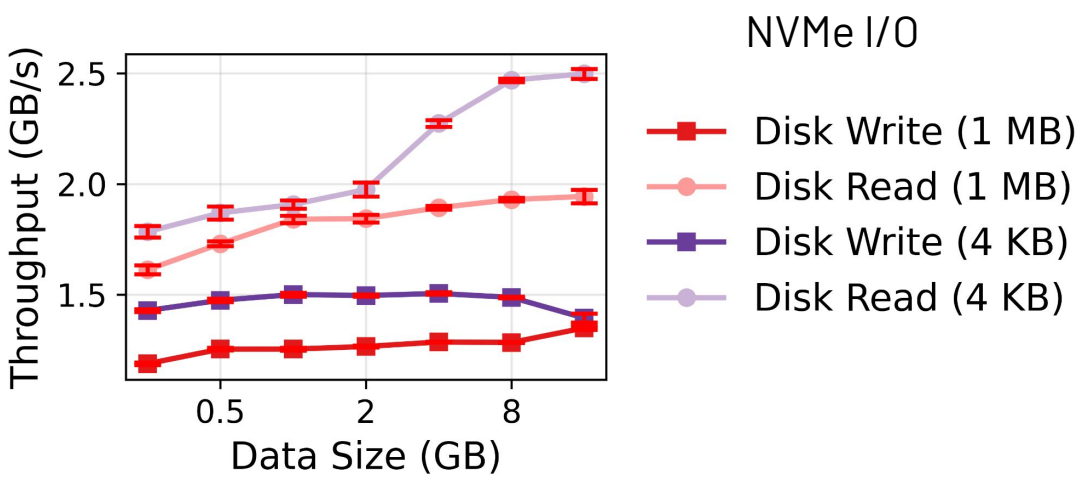
Cold-start breakdown from Pod creation to model registration.

"Transparent and Unified CPU-GPU Snapshots for LLM Inference Platforms". Middleware 2026

Infrastructure-level GPU Snapshots



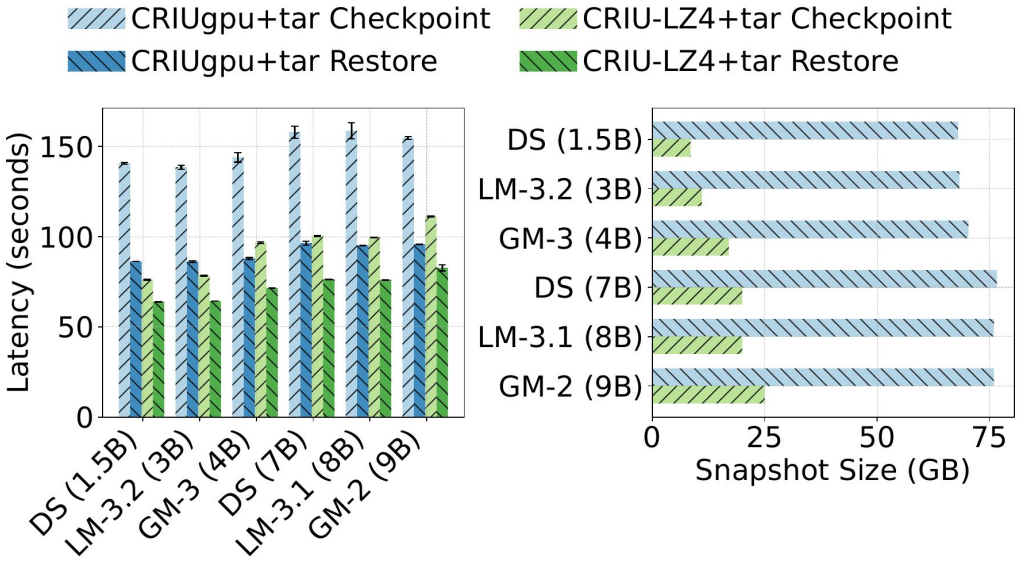
NVIDIA RTX 5090 (PCIe 5.0)

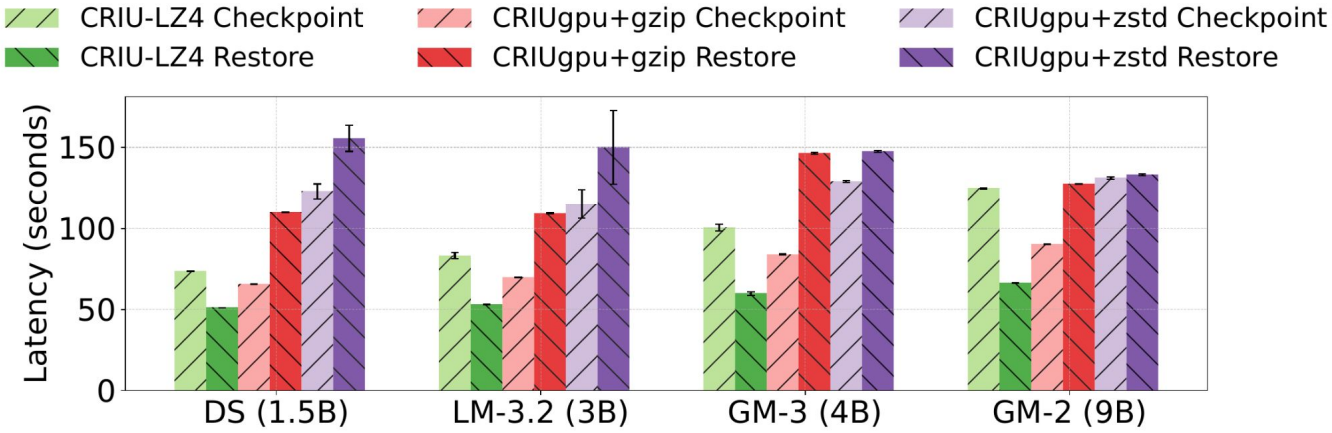


Storage bandwidth, not GPU memory transfer, is the primary bottleneck on the restore critical path because NVMe disk I/O is significantly slower than CUDA C/R throughput.

"Towards On-the-Fly Snapshot Memory Compression for Low-Latency Elastic Inference Serving Systems". EuroMLSys '26

Restore Latency & Compression Ratio

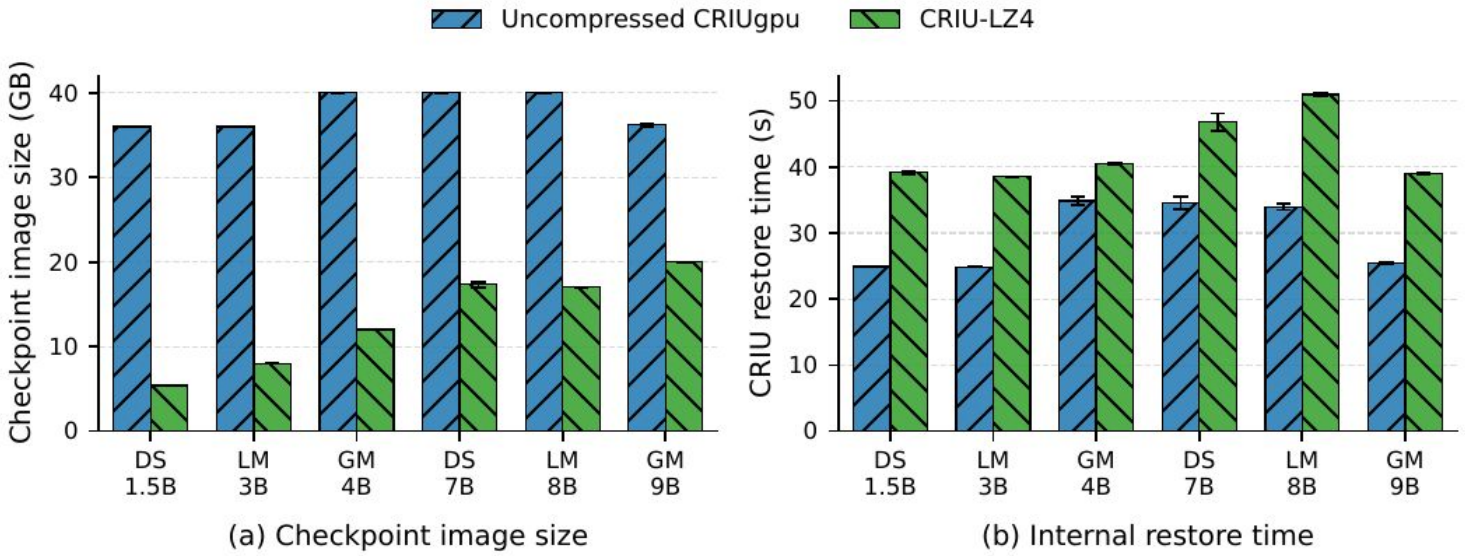




Across DeepSeek (DS), Gemma (GM), and LLaMA (LM) on vLLM (NVIDIA H100), on-the-fly compression reduces snapshot size by up to 6× and restore up to 3× faster than offline compression.

"Towards On-the-Fly Snapshot Memory Compression for Low-Latency Elastic Inference Serving Systems". EuroMLSys '26

Page-level Compression Limitations



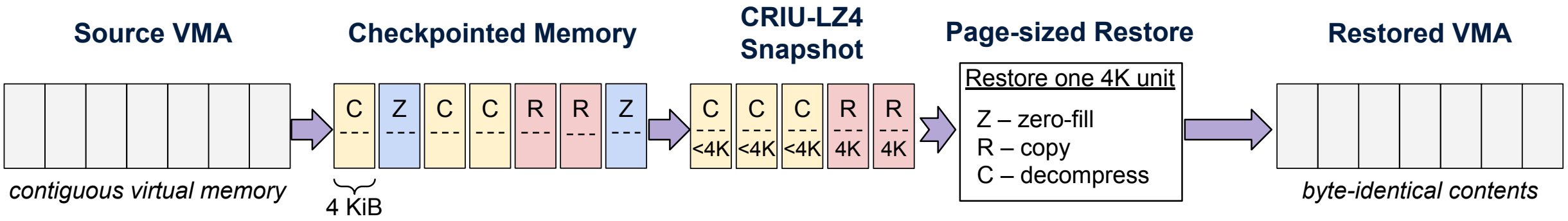
vLLM workloads on NVIDIA A100 (SXM4 40GB)

CRIU-LZ4 Results

- Smaller Snapshots (≈45–85%)
- Increased Restore Latency (+15–60%)

"Towards On-the-Fly Snapshot Memory Compression for Low-Latency Elastic Inference Serving Systems". EuroMLSys '26

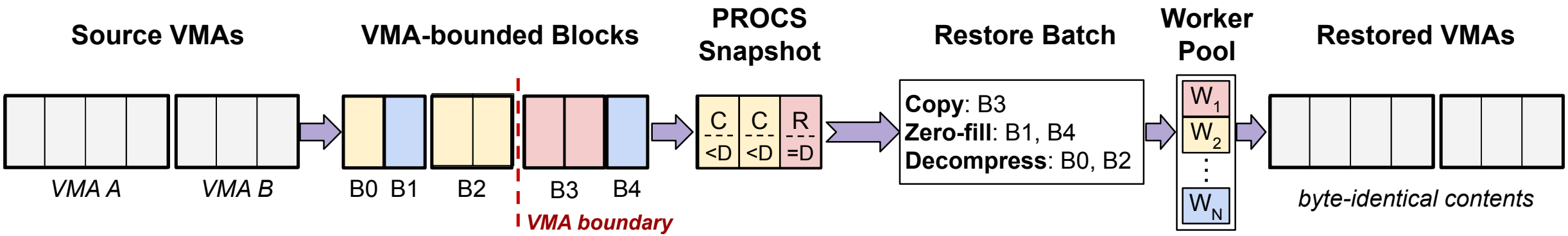
Page-level Compression Limitations



CRIU-LZ4 compresses and restores each **4 KiB page sequentially on a single thread**, limiting compression efficiency and creating a restore bottleneck.

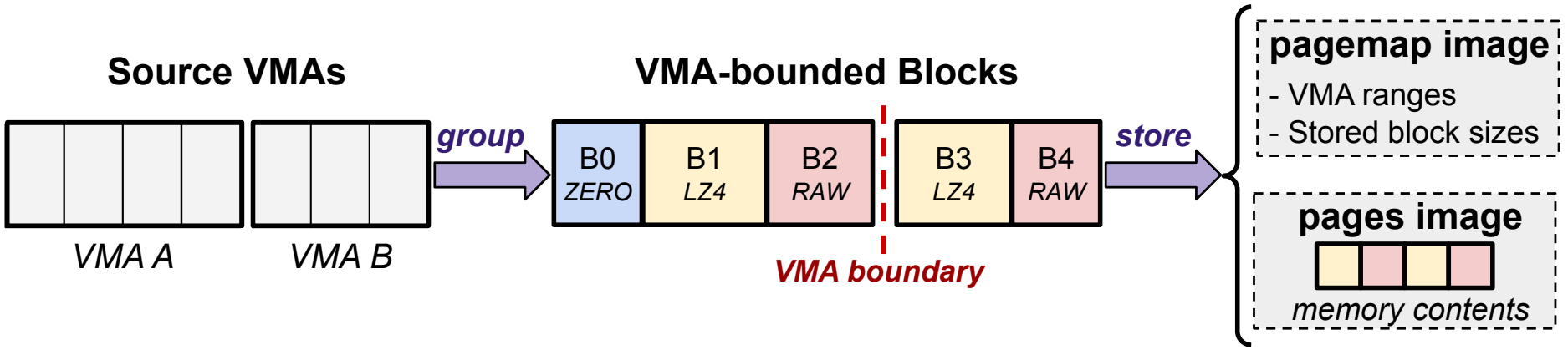
"Towards On-the-Fly Snapshot Memory Compression for Low-Latency Elastic Inference Serving Systems". EuroMLSys '26

Parallel Restore-Optimized Compression



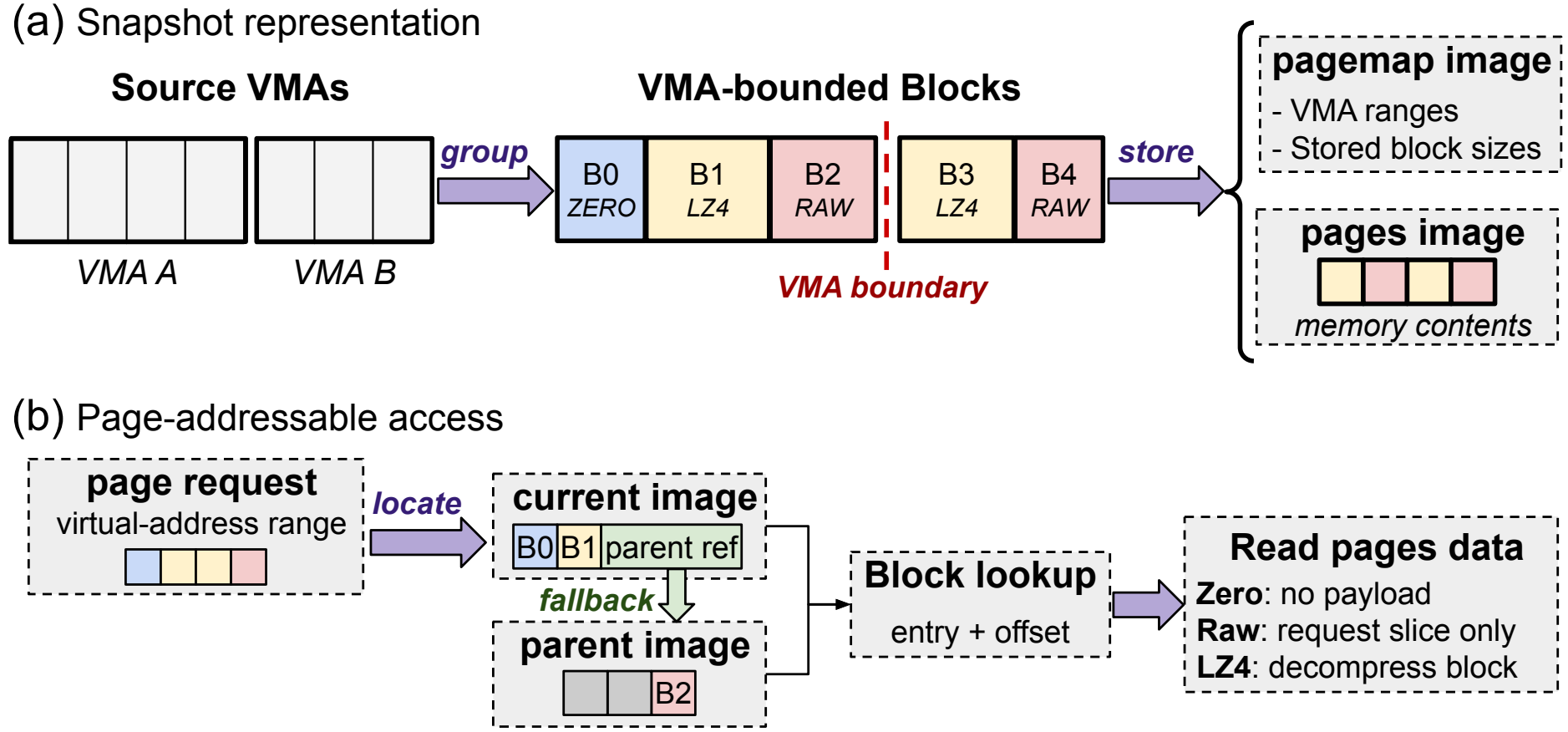
Key insight: Grouping contiguous virtual memory into variable-size blocks before compressing.

Parallel Restore-Optimized Compression

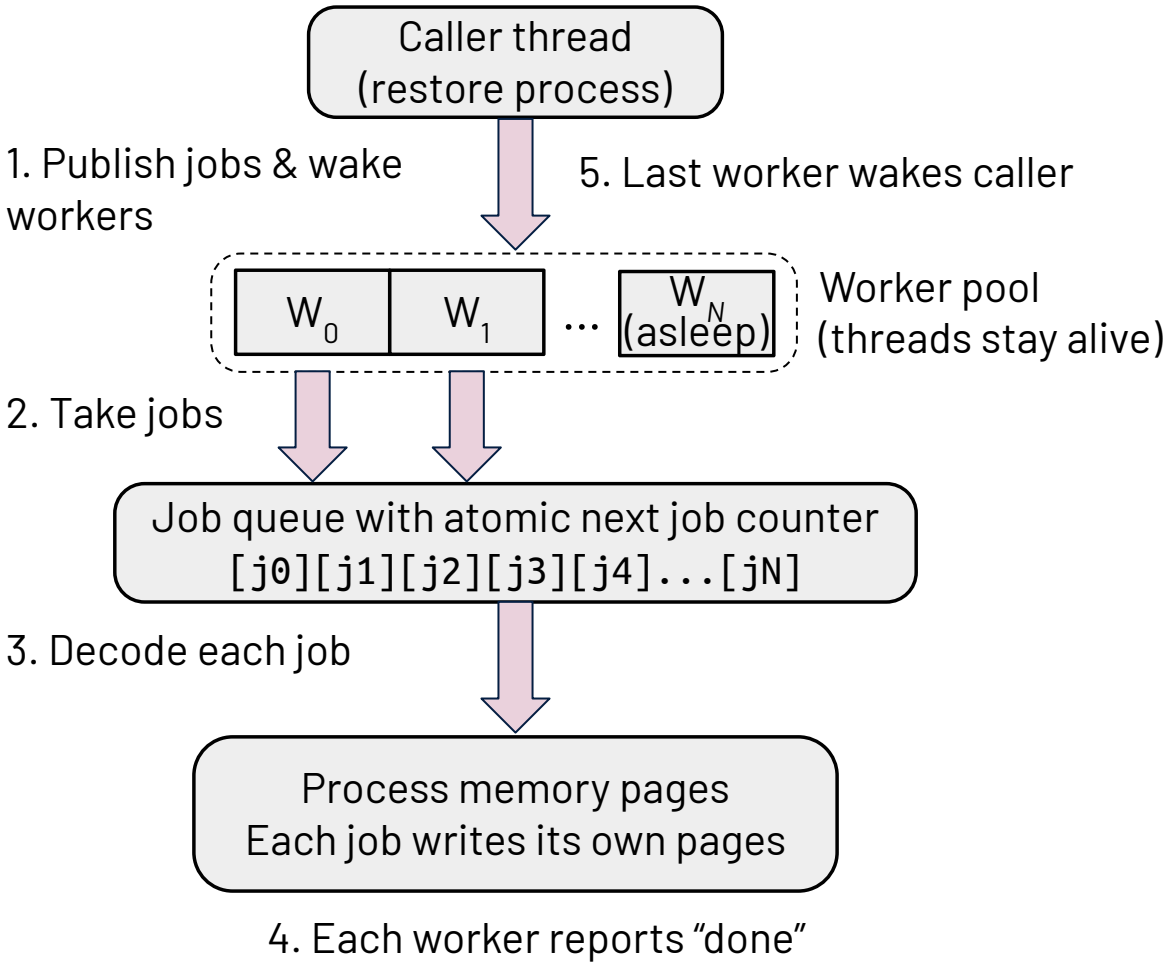


Each block is stored as zero (no data), LZ4-compressed, or raw if LZ4 does not reduce it below 87.5% of its original size.

Page-level Incremental Restore



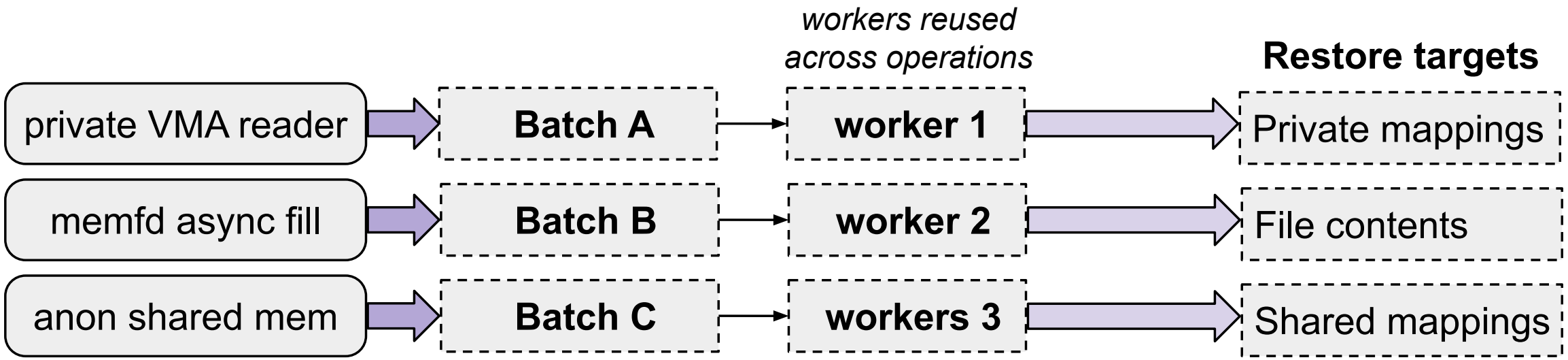
Coordinating Parallel Restore



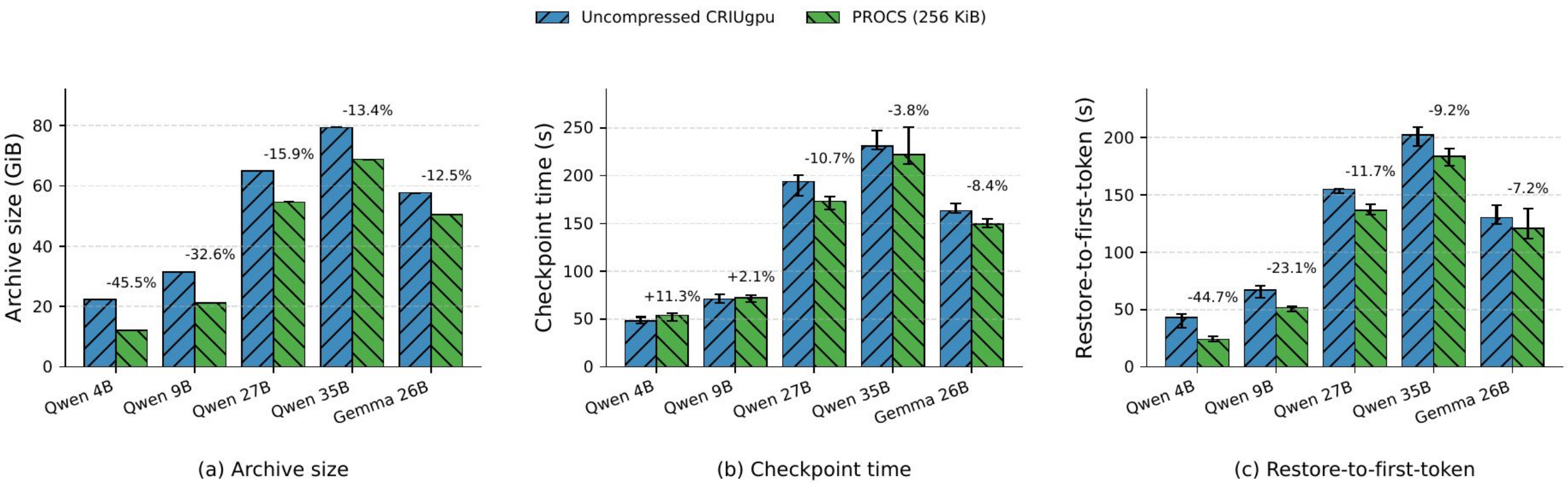
Parallel Page Decompression

- Caller wakes only needed workers
- All threads claim jobs atomically
- Each job writes own pages
- Last worker wakes the caller
- Caller then reuses its buffers

Parallel Restore Workers

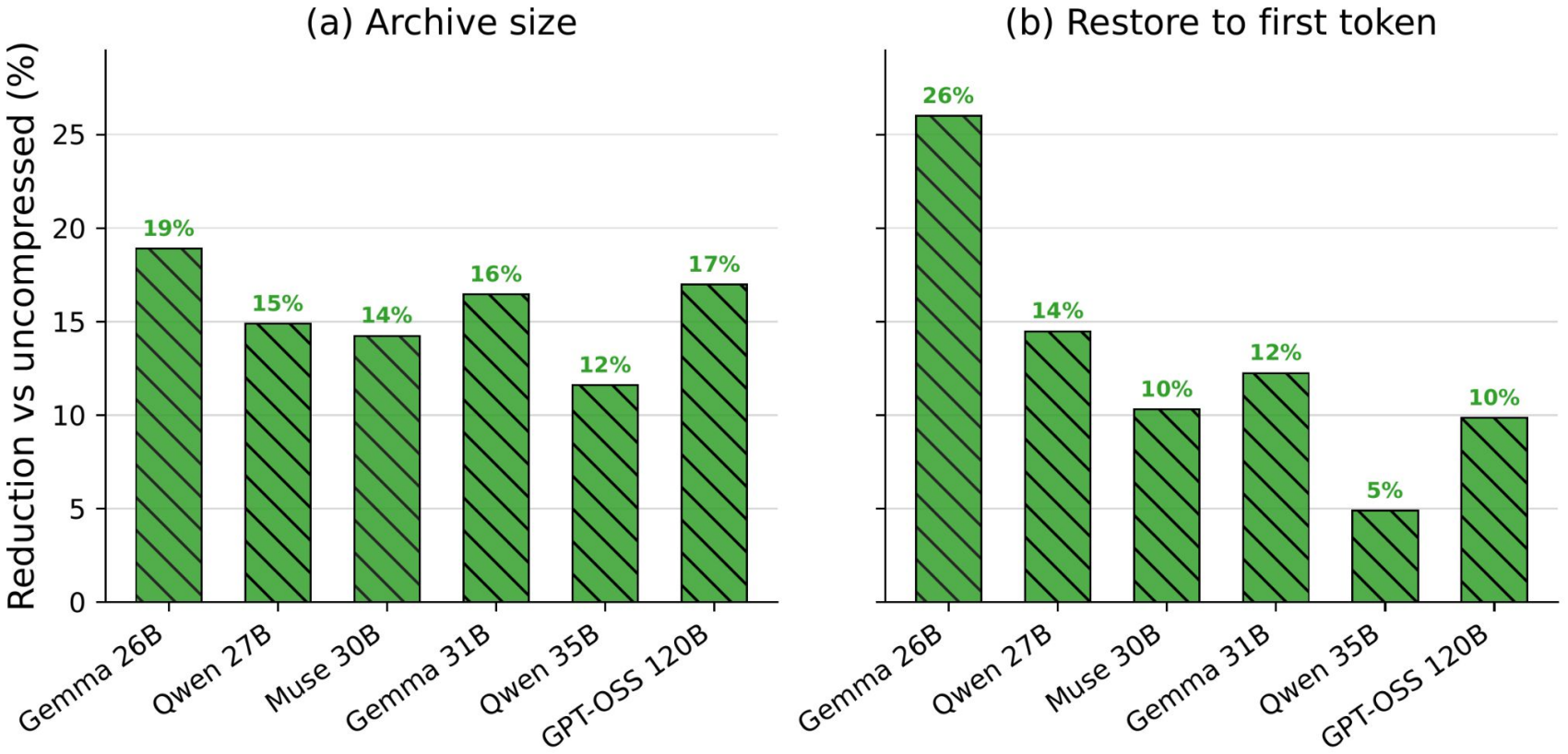


End-to-End Performance Results



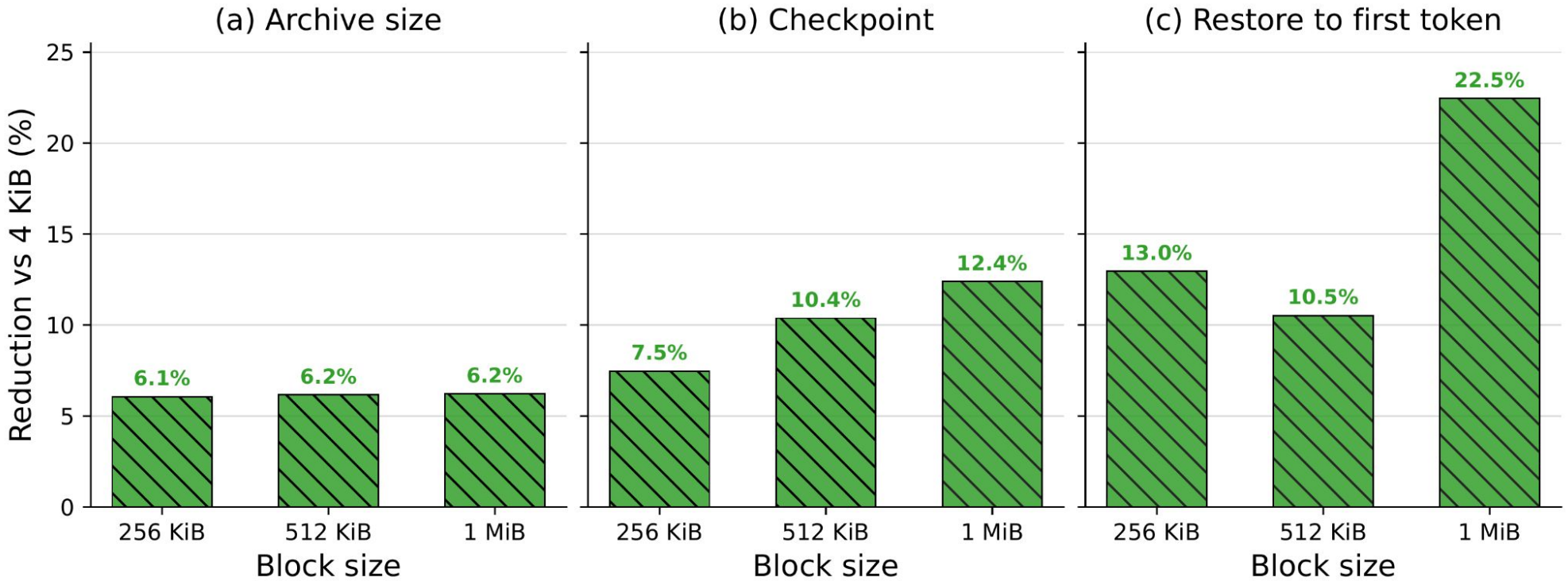
On NVIDIA A100 (SXM4 40GB), PROCS with 256 KiB blocks reduces archive size by 13–46% and restore-to-first-token latency by 7–45% compared with CRIUgpu.

End-to-End Performance Results



On a NVIDIA B200, PROCS reduces archive size by 12–19% and restore-to-first-token latency by 5–26% across six vLLM models.

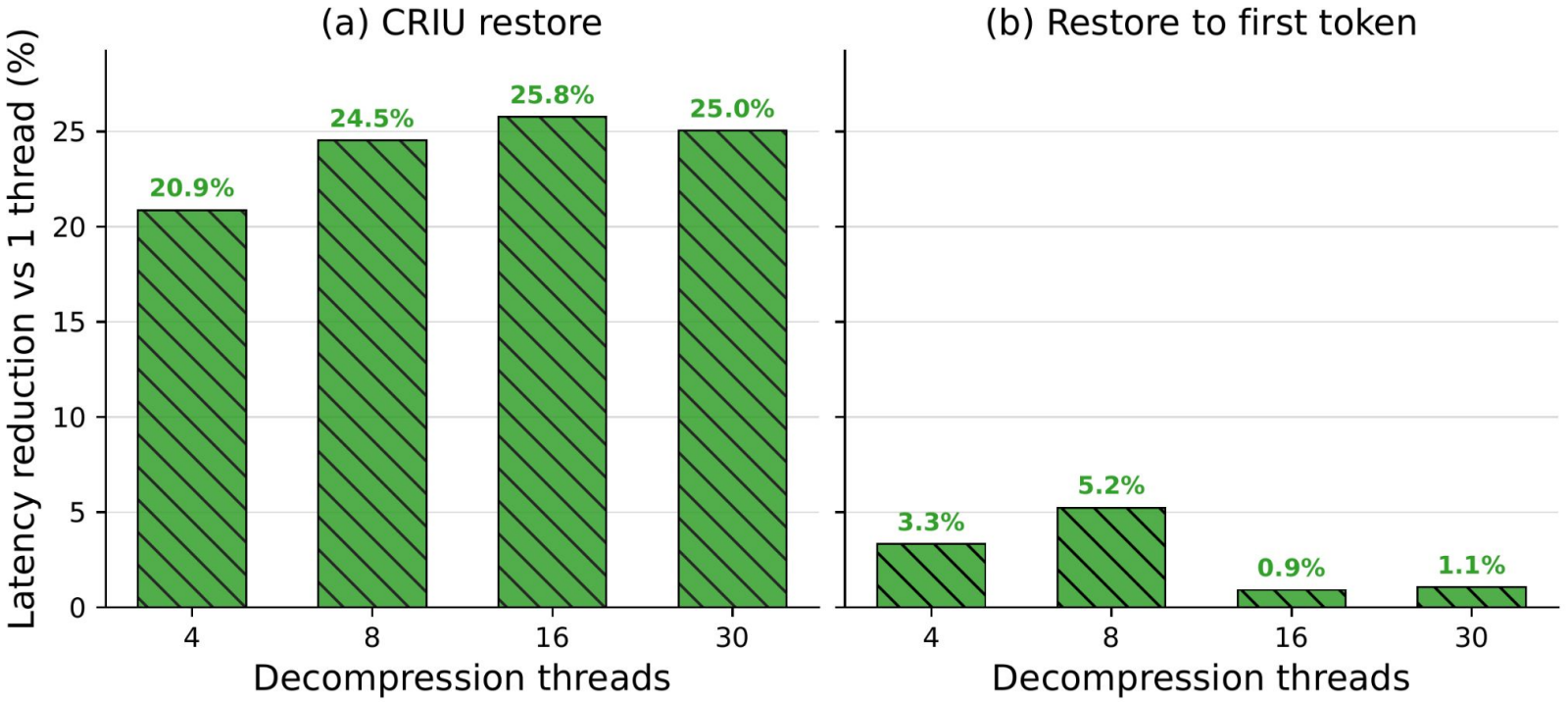
Impact of Compression Block Size



Increasing block size mainly helps speed rather than storage.

On NVIDIA RTX PRO 6000 (96 GB), larger LZ4 blocks make snapshots only 6% smaller but reduce checkpoint time by 8–12% and restore-to-first-token time by 11–23%.

Scaling Decompression Parallelism



PROCS with 256 KiB blocks for Qwen3.5-27B-FP8 (SGLang) running on B300 GPU.

Increasing the number of decompression threads further reduces the restore latency.
As restore time becomes a smaller, container runtime overheads become the bottleneck.

Key Takeaways

- Optimal snapshot compression parameters are workload-specific
- Larger compression blocks improve mainly restore latency
- Parallel decompression has limited end-to-end impact
- Container runtime overheads now account for most restore latency

Summary & Questions

- Block-level compression enables faster restore & smaller checkpoints
- Block size and decompression threads are simple tuning parameters
- The remaining bottlenecks are outside CRIU, e.g., container runtime.



github.com/checkpoint-restore/criu

radostin.io/blog/criu-memory-compression