

Solution Contribution for vLLM KVCache

Replacing Block-Table Indirection with Coordinate Addressing

Taeho Lee*
SSCCS Initiative
ssccs.org

August, 2026

1 Abstract

vLLM is the de facto LLM serving engine. Its KV cache, managed by PagedAttention, splits into fixed-size blocks and maps logical token positions to physical blocks through a per-sequence block table. Every attention computation gathers K and V through that table, and the write path loads one table entry per token position.

The tagma backend on the SSCCS vLLM fork replaces this indirection with range arithmetic: a request holds its KV blocks as contiguous ranges, so the physical block is a closed form of the logical position, with no per-position block-table load. The same structural pattern was measured end to end in the CERN ROOT-Coord work; those numbers do not project a vLLM speedup. The software level is implemented, tested, and measured on the host; the GPU serve benchmark is pending. The detailed topic report is the [KV Cache report](#).

2 What SSCCS Brings

SSCCS is an independent, open-source computing infrastructure advancing open structural computing. For vLLM it brings the Tagma Map coordinate space (hash-less, $O(1)$ arithmetic addressing), the Chton materialization layer, the base11172 self-validating encoding, and Apache 2.0 releases. The core principle is validated in production scientific workloads (CERN ROOT-Coord).

3 The Problem

PagedAttention reduced KV cache fragmentation but introduced an indirection layer: the per-request block table. The attention kernels and the write path resolve every token position through it, one extra memory access that scales with sequence length and batch size. Community optimizations compact the table but stay inside the paradigm. The access pattern detail is in the [KV Cache report](#).

4 The Approach

The tagma backend addresses the KV cache directly:

```
n(layer, block, token) = (layer * B + block) * T + token  
offset = n * bytes_per_token
```

Requests hold contiguous block ranges from the coordinate allocator, so the physical block is derived by arithmetic with no table lookup. The mechanism, the allocator, and the implementation boundary are in the [KV Cache report](#).

*Founder & Architect; Corresponding author: lee@ssccs.org

5 The Offer

- Same API: the scheduler-facing `KVCacheManager` contract is preserved
- Same user experience: `vllm serve` with a new `--kv-cache-backend tagma` selector
- Same ecosystem: unsupported configurations fail closed rather than silently degrading

The write path changes; the read path keeps the materialized block table in this version because external attention backends with fixed ABIs consume it.

6 Why This Is a Structural Reset, Not an Optimization

Approach	Nature	Result
Flattened or fused block-table variants	Optimization of the table	Incremental improvement
Block-table caching	Optimization of table access	Incremental improvement
Tagma Map	Removal of the table on the write path	Structural reset

Making the index faster is an optimization; removing the index is a structural change.

7 Status

The implementation is a working first version on the SSCCS vLLM fork ([ssccsorg/vllm](https://github.com/ssccsorg/vllm), branch `52-tagma-kv-vllm`): the C++ engine under `csrc/tagma`, the `vllm.tagma` host extension, the scheduler-facing manager, and the write-path slot mapping, with 16 C++ engine tests, 23 Python tests at the software level, and the host benchmark passing. The software level is complete and measured on the host; the GPU serve benchmark (paged against tagma, ShareGPT at 16K and 128K context), the read-path decision, and MLA and hybrid cache support are pending. The upstream PR is opened only after the measured comparison and a human review of every changed line. The detailed measured results are in the [KV Cache report](#).

8 References

- vLLM `KVCacheManager`
- PagedAttention: vLLM's memory management
- CERN ROOT-Coord report
- Tagma Map documentation
- SSCCS vLLM fork