

Solution Contribution for LLM Serving Engines

Applying Coordinate Addressing to Open-Source LLM Infrastructure

Taeho Lee*
SSCCS Initiative
ssccs.org

August, 2026

1 Abstract

LLM serving engines span the two representative axes of the LLM ecosystem. Enterprise data-center serving runs vLLM, which splits the KV cache into fixed-size blocks and maps logical token positions to physical blocks through a block table. Personal and edge devices, the deployment class of the Rem device, run llama.cpp, which assigns each token a cache cell and passes per-token index vectors into the attention graph. Both axes pay the same cost: every attention computation and every write resolves positions through an indirection that scales with sequence length and batch size.

This work applies the SSCCS tagma coordinate principle to both engines. Covering the two axes demonstrates the syntagma infrastructure technology in the most representative category of AI and contributes the result back to the two open-source projects. A request holds its KV blocks as contiguous ranges, so the physical position is a closed form of the logical position, with no per-position table load. The same structural pattern was measured end to end in the CERN ROOT-Coord work; those numbers do not project a serving speedup. The vLLM backend is implemented at the software level on the SSCCS fork with the GPU benchmark pending. The llama.cpp investigation targets a GPU-free proof on CPU and Apple Silicon, where the reference benchmark runs without an accelerator.

2 What SSCCS Brings

SSCCS is an independent, open-source computing infrastructure advancing open structural computing. For LLM serving 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

The KV cache indirection is shared across engines. PagedAttention reduced fragmentation but introduced a per-request block table; llama.cpp assigns each token a cache cell and passes per-token index vectors into the attention graph. Both pay one extra memory access per token position that scales with sequence length and batch size. Community optimizations compact the table or the index but stay inside the paradigm.

4 The Approach

The tagma backend addresses the KV cache directly:

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

```
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.

5 Target Projects

Project	Platform	Proof surface	Status
vLLM	NVIDIA CUDA, data-center serving	Write-path slot mapping without a block-table load; serve benchmark paged against tagma	Implemented at the software level on the SSCCS fork; GPU benchmark pending
llama.cpp	CPU and Apple Silicon (Metal), GPU-free benchmark	KV cache cell mapping replaced by contiguous ranges; llama-bench before and after	Investigation planned

The vLLM topic report is [vLLM](#); the llama.cpp topic is [llama.cpp](#).

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 vLLM integration is a working first version on the SSCCS vLLM fork ([ssccsorg/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, and the host benchmark passing. The GPU serve benchmark and the read-path decision are pending. The llama.cpp investigation is in planning: the KV cache layer exposes a cell-index mapping with a contiguous-slot mode, and the profile spike on long-context decode decides the implementation.

8 References

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