CUDA Performance Analysis of Sciara-fv2 Lava Flow Simulator

Nhat Quang Dang

University of Calabria dngntq02p19z251l@studenti.unical.it

Abstract. We implement five CUDA versions of Sciara-fv2 lava flow simulator: Global, Tiled, Tiled+Halo, CfAMe, and CfAMo. Roofline analysis on GTX 980 shows all versions are memory-bound (AI < 0.05). CfAMo achieves 1.09× speedup via kernel fusion. CfA versions produce different checksums due to floating-point non-associativity in atomic operations—an acceptable numerical variation.

Keywords: CUDA · Cellular Automata · Roofline · GPU

1 Introduction

Sciara-fv2 is a Cellular Automata model simulating lava flows on a 2D grid with Moore neighborhood (9-cell stencil). Each time step has four phases: lava emission, outflow computation (minimization algorithm), mass balance, and solidification. We parallelize by mapping each cell to one CUDA thread.

Setup: GTX 980 (224 GB/s bandwidth, 156 GFLOP/s FP64), Mt. Etna 2006 dataset (16,000 steps), block size 16×16.

2 CUDA Implementations

We implement five versions exploiting different memory hierarchies:

Global (Baseline): Direct global memory access with Unified Memory. Constant memory for neighborhood offsets.

Tiled: Loads 16×16 tiles into shared memory (6 KB). Boundary cells still access global memory.

Tiled+Halo: Extended 18×18 tiles with 1-cell halo (7.8 KB). All neighbor accesses from shared memory.

CfAMe: Fuses outflow computation and mass balance using atomic scatter pattern (`atomicAdd`).

CfAMo: Memory-optimized CfAMe eliminating 12.5 MB intermediate buffer.

2.1 Checksum Mismatch Explanation

Global, Tiled, and Tiled+Halo produce **identical checksums**. CfAMe/CfAMo differ due to: (1) **floating-point non-associativity**—atomic additions execute

in non-deterministic order, and $(a + b) + c \neq a + (b + c)$ in IEEE-754; (2) scatter-based approach applies flows immediately vs. buffer accumulation. These are **acceptable numerical variations**: total emitted lava (46997.805865 m) matches across all versions.

3 Performance Assessment

Table 1. Execution time, CUDA configuration, and Roofline metrics (16,000 steps).

Version	Time (s)	Speedup	Shared	AI	GFLOP/s	Occup.
Global	21.60	1.00×	0 KB	0.043	42.6	55.1%
Tiled	20.14	1.07×	6.0 KB	0.046	40.6	58.7%
Tiled+Halo	23.33	0.93×	7.8 KB	0.048	37.4	62.7%
CfAMe	24.69	0.88×	0 KB	0.020	6.4	58.6%
CfAMo	**19.74**	**1.09**×	0 KB	0.021	6.6	58.6%

All versions use grid 33×24, block 16×16. Dominant kernels: `massBalance` (27%) and `computeOutflows` (24%) for non-CfA versions.

Analysis: Tiled+Halo is slower than Global because synchronization overhead (`__syncthreads()`) exceeds benefits—the small grid (1.5 MB/substate) fits in L2 cache (2 MB). CfAMo wins by eliminating the 12.5 MB flow buffer; sparse lava (<5% active cells) minimizes atomic contention.

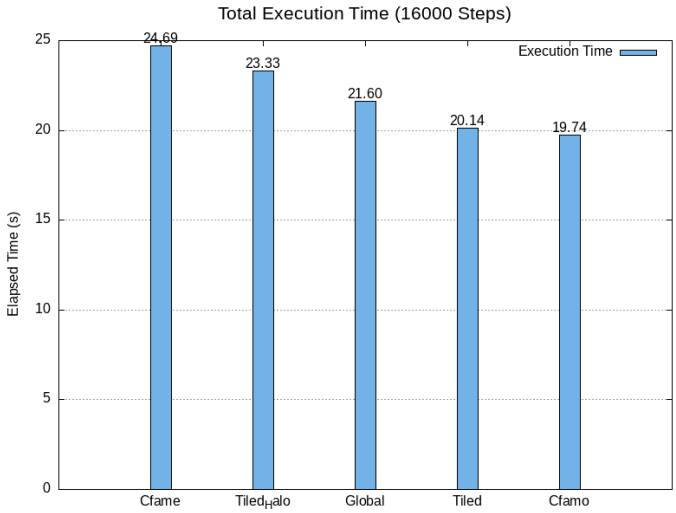

Fig. 1. Execution time comparison (lower is better).

4 Roofline Analysis

GTX 980 specifications from `gpumembench`: global bandwidth 224 GB/s, peak FP64 156 GFLOP/s. Ridge point: $156/224 = 0.695$ FLOP/Byte.

AI Calculation (Global version): For `computeOutflows`: $AI \approx 0.043$.

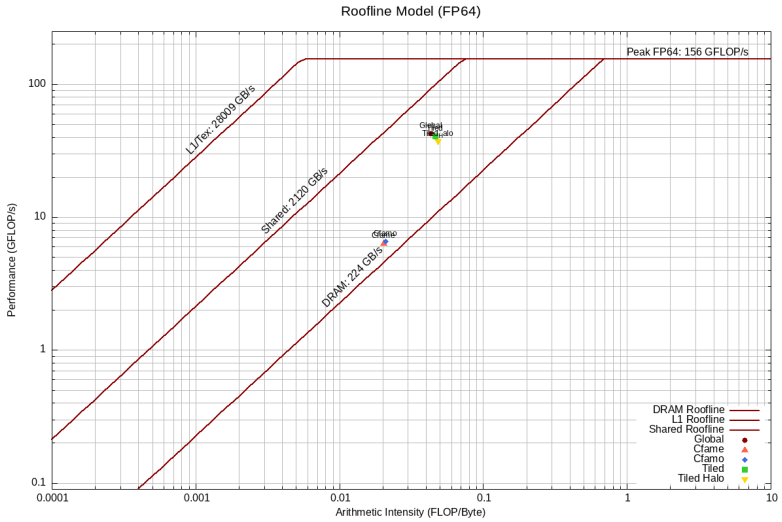

Fig. 2. Roofline model. All versions memory-bound (AI $\ll$ 0.639 ridge point).

All kernels are **memory-bound** (AI < 0.05 vs ridge 0.639). Low AI stems from stencil access pattern (9 neighbors $\times$ 3 substates $\times$ 8 bytes). Tiled+Halo achieves highest occupancy (62.7%) but worst performance—confirming that **occupancy $\neq$ performance** for memory-bound workloads [1].

5 Conclusions

Findings: (1) All versions memory-bound; memory efficiency is critical. (2) Tiling ineffective for small grids fitting L2 cache. (3) CfAMo achieves best speedup (1.09×) via buffer elimination. (4) CfA checksums differ due to FP non-associativity but remain physically valid.

Bottleneck: Memory operations dominate GPU time—34.7% is spent on DtoD buffer swaps, while compute kernels (massBalance 27.7%, computeOutflows 24.0%) account for only 51.7%.

Future: Eliminate double-buffering, texture memory for read-only data, active cell list.

References

1. V. Volkov, "Better performance at lower occupancy," in *Proceedings of the GPU technology conference, GTC*, vol. 10. San Jose, CA, 2010, p. 16.