CUDA Parallelization of Sciara-fv2 Lava Flow Simulator

Performance Analysis and Optimization

Nhat Quang Dang
Reg. No. 279990

University of Calabria
dngntq02p19z251l@studenti.unical.it

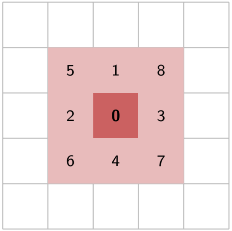
January 15, 2025

Outline

Context: Sciara-fv2 Lava Flow Simulator

What is Sciara-fv2?

- Cellular Automata model for **lava flow simulation**
- Simulates Mt. Etna 2006 eruption
- Grid: $517 \times 378 =$ **195,426 cells**
- Each cell: altitude, lava thickness, temperature



Moore Neighborhood

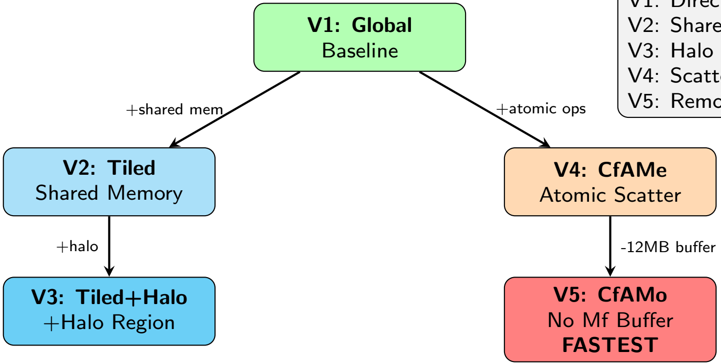
Challenge:

- Moore neighborhood (9 cells stencil)
- Memory-intensive computation
- Need GPU parallelization

GPU Target

NVIDIA GTX 980 (Maxwell)
2048 CUDA cores, 2MB L2 cache

Roadmap: 5 CUDA Optimization Strategies

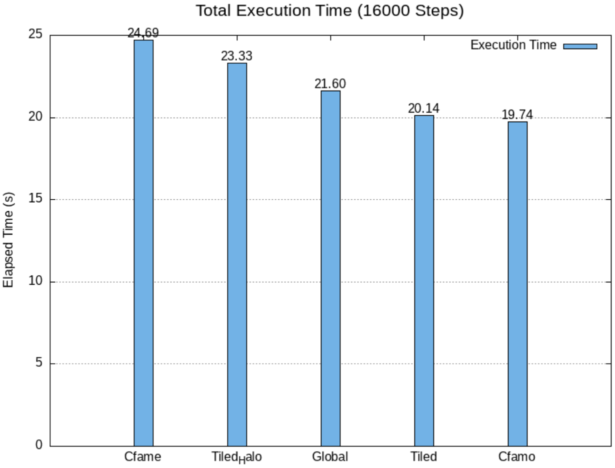


Techniques:
V1: Direct global memory
V2: Shared memory tiling
V3: Halo for border cells
V4: Scatter with atomics
V5: Remove flow buffer

← Tiled Branch
Reduce memory latency

Atomic Branch →
Reduce memory footprint

Execution Time Comparison



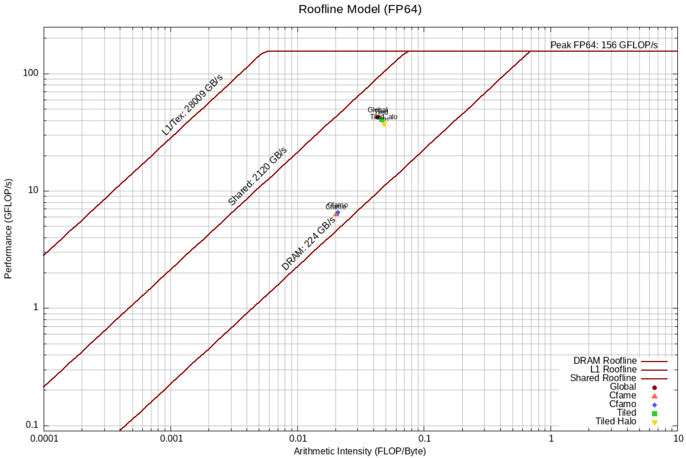
Version	Time (s)	Speedup
CfAMe	24.69	0.88×
Tiled+Halo	23.33	0.93×
Global	21.60	1.00×
Tiled	20.14	1.07×
CfAMo	**19.74**	**1.09×**

Key Finding

CfAMo is fastest despite lower occupancy!

Eliminating 12MB flow buffer improves cache efficiency.

Roofline Analysis



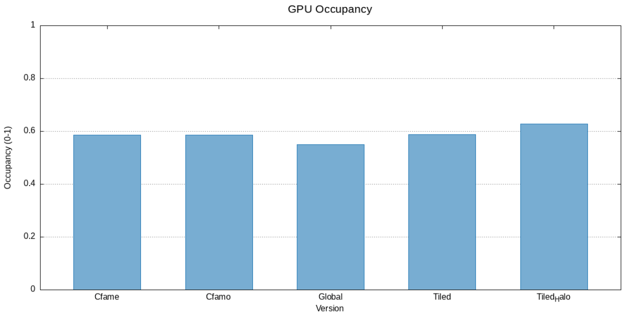
Observations

- All versions: **AI** $<$ **0.05**
- Ridge point: **0.696**
- All are **memory-bound**

Version	**AI**
Global	0.043
Tiled	0.046
Tiled+Halo	0.048
CfAMe	0.020
CfAMo	0.021

Low AI = stencil access pattern (9 neighbors × 3 substates)

GPU Occupancy Analysis



Version	Occupancy
Global	55.1%
Tiled	58.7%
Tiled+Halo	**62.7%**
CfAMe	58.6%
CfAMo	58.6%

Important Insight

High occupancy $\neq$ Fast!

Tiled+Halo has highest occupancy (62.7%) but is **slower** than Global.

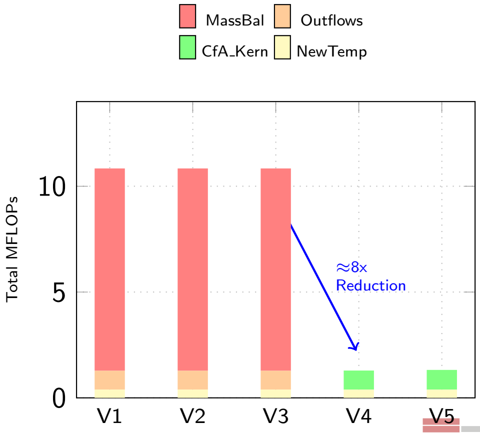
`__syncthreads()` overhead exceeds shared memory benefits.

FLOP Count Analysis: Detailed Breakdown

Kernel FLOPs Heatmap (MFLOPs)

Kernel	V1	V2	V3	V4	V5
MassBal	9.56	9.56	9.56	-	-
Outflows	0.89	0.89	0.89	-	-
CfA_Kern	-	-	-	0.90	**0.93**
NewTemp	0.39	0.39	0.39	0.39	0.39
Total	**10.8**	**10.8**	**10.8**	**1.3**	**1.3**

Insight: The computation-heavy kernels (Red) are replaced by efficient atomic scatter operations (Green).

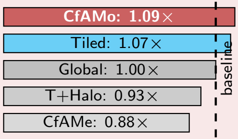


Conclusions & Key Takeaways

Main Results

1. **CfAMo is fastest** (19.74s, 1.09×)
2. All versions are **memory-bound**
3. **High occupancy ≠ performance**
4. Memory footprint > compute optimization

Performance Summary



Why CfAMo Wins

- Eliminates 12MB flow buffer
- Better cache utilization
- Sparse lava (<5%) → low atomic contention

Lesson Learned

For small grids fitting in L2 cache, **reducing memory footprint** beats shared memory tiling.

Thank You!

Questions?

Nhat Quang Dang
Reg. No. 279990
dngntq02p19z251l@studenti.unical.it
University of Calabria

January 15, 2025