Self Introduction



Yanzhou Tang

CS PhD Student

Georgia Institute of Technology

About Me

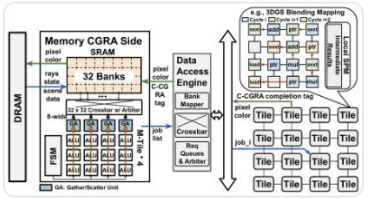
Hi! I am **Yanzhou Tang**, a first-year CS Ph.D. student at the Georgia Institute of Technology, advised by Prof. Yingyan (Celine) Lin in the Efficient and Intelligent Computing (EIC) Lab.

My research interests lie broadly in **computer systems** and **computer architecture**, and I currently focus on **efficient agentic AI systems** and **efficient spatial computing systems**.

Before joining Georgia Tech, I was a research intern at the SAI Lab at New York University, advised by Prof. Sai Qian Zhang and working closely with Dr. Cheng Tan. I received both my bachelor's and master's degrees from Shanghai Jiao Tong University.

I am always open to collaborations. If my research interests you, feel free to email me!

MICRO 2026

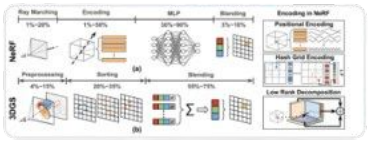


RAICHU: Rendering Acceleration Integrating CGRA arcHitecture for Unified VR Pipelines

Yanzhou Tang, Haiyu Wang, Cheng Tan, Sai Qian Zhang

In 59th IEEE/ACM International Symposium on Microarchitecture (MICRO), 2026.

ICCAD 2026



ViViD: A Versatile CGRA-Based Framework for Visual Rendering Acceleration on VR Devices

Yanzhou Tang, Tianhua Xia, Cheng Tan, Sai Qian Zhang

In 45th IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2026.

Georgia Tech

GaTech System Seminar

RAICHU: Rendering Acceleration Integrating CGRA arcHitecture for Unified VR Pipelines

Yanzhou Tang[1,2] Haiyu Wang[2] Cheng Tan[3] Sai Qian Zhang[2]

[1]Georgia Institute of Technology [2]New York University [2]Google / Arizona State University



Georgia Institute of Technology







Georgia Tech

What is rendering?

Rendering generates a 2D image from a 3D scene and a camera viewpoint.

Scene and rendered views



The same scene, seen from new viewpoints

Rendering in VR



Games, training and virtual tours

90 Hz: 11.1 ms/frame **120 Hz: 8.3 ms/frame**

Georgia Tech

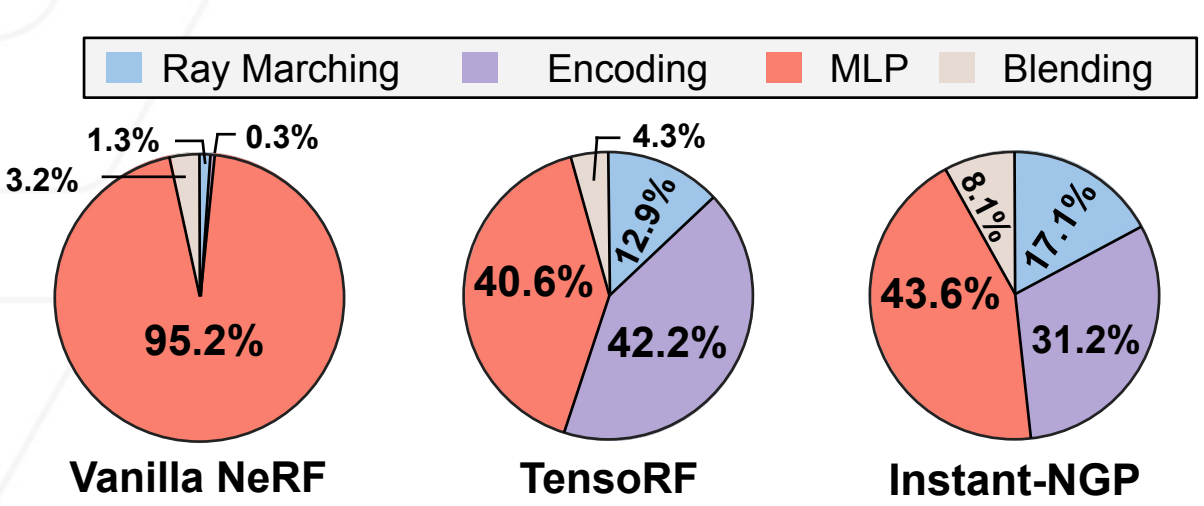
Multiple rendering needs in VR

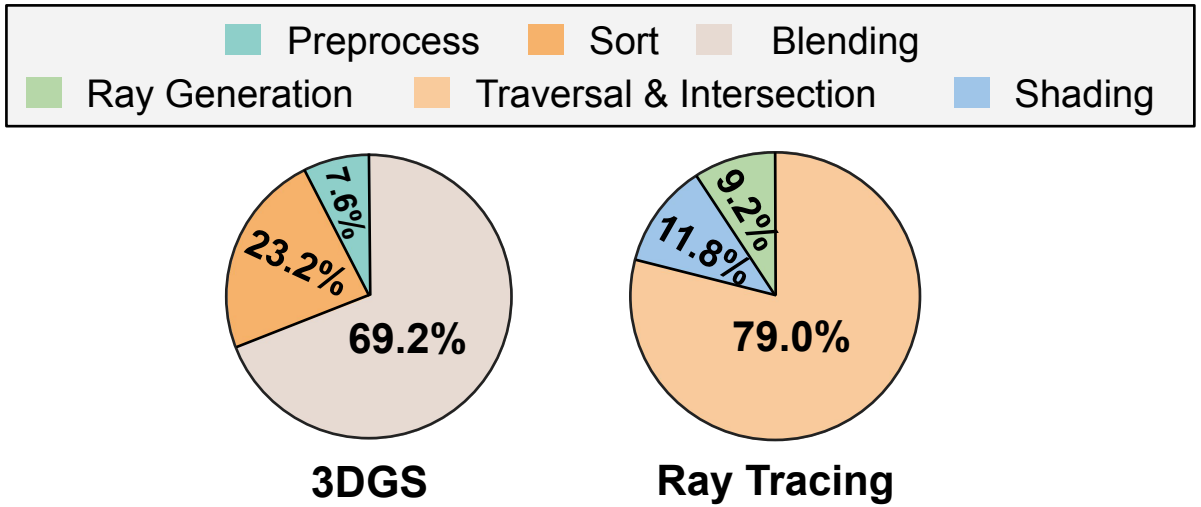
Attributes	M-NeRF	H-NeRF	L-NeRF	3DGS	RT
FPS (Orin NX)	1.7	1.1	5.9	6.4	0.5
Speed	Medium	High	High	High	Low
Quality	Medium	High	High	High	Very High
Mem. Eff.	Very High	High	High	Medium	Medium
Typical Use Case	Legacy Low-End	Static BG	Static BG	FG Objects Avatars	Dynamic Effects

Note: M-NeRF denotes MLP-based NeRF; H-NeRF denotes Hash Grid NeRF; L-NeRF denotes Low-Rank Decomposition NeRF; RT denotes ray tracing.

Georgia Tech

Different rendering bottlenecks





Different bottlenecks need different resources

Encoding in NeRF
Sorting in 3DGS
Traversal in Ray Tracing

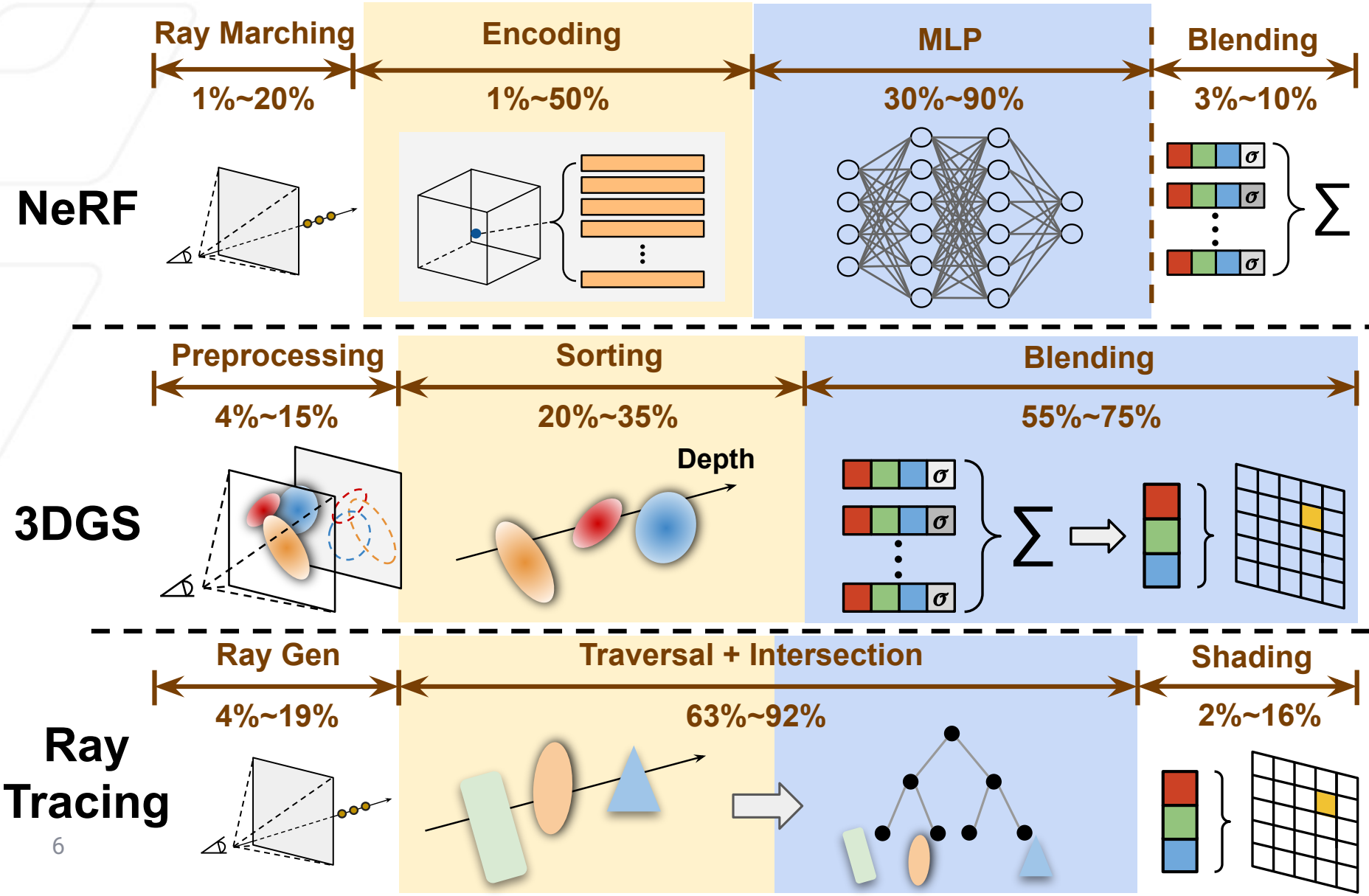
Memory Bound

MLP in NeRF
Blending in 3DGS
Intersection in Ray Tracing

Compute Bound

Georgia Tech

Same pipeline pattern

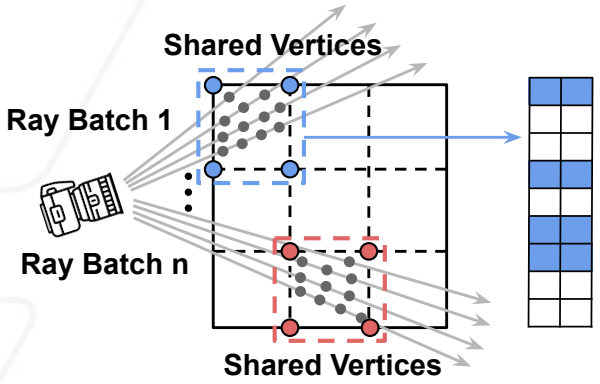


Memory Bound

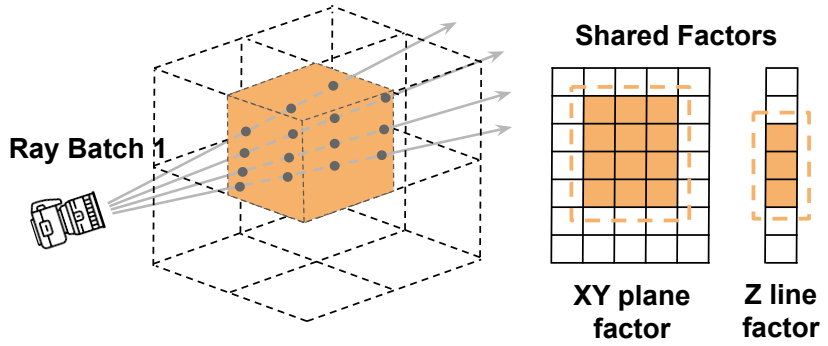
Compute Bound

Georgia Tech

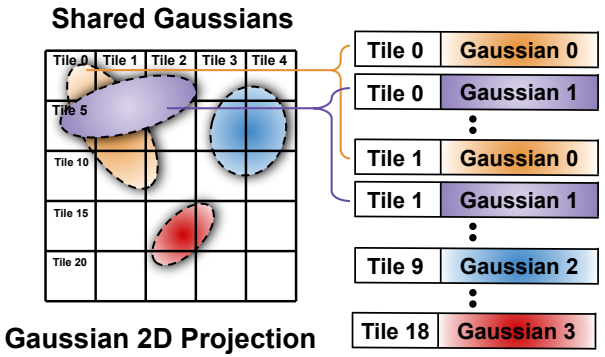
Shared data behind irregularity



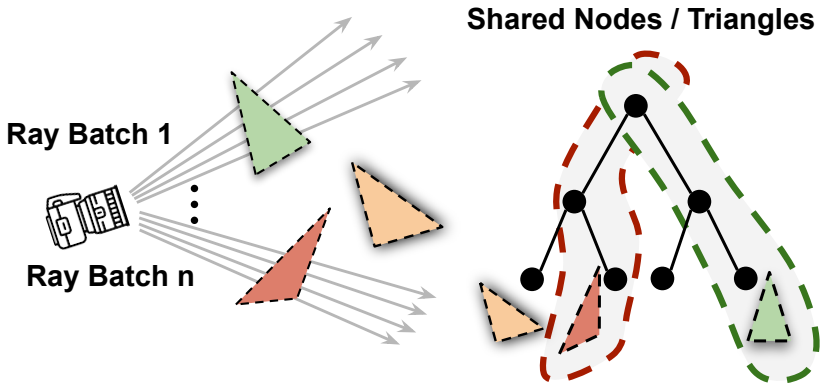
(a) Sharing in Hash Grid NeRF



(b) Sharing in Low Rank NeRF



(c) Sharing in 3DGS



(d) Sharing in Ray Tracing

Key observation

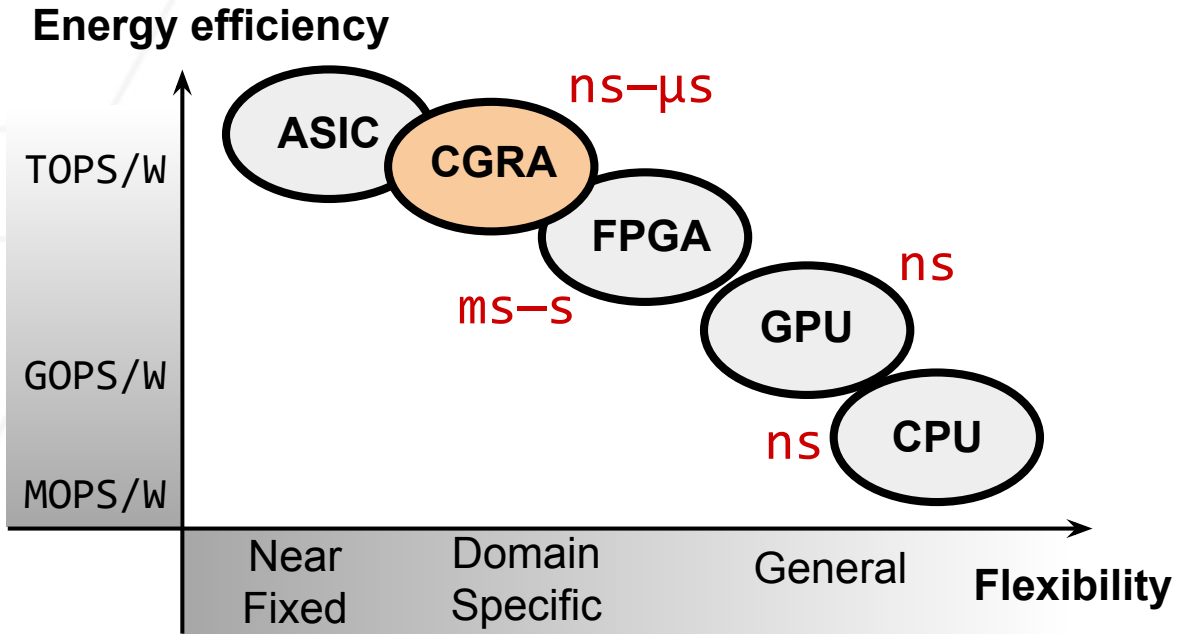
Nearby rays and pixels often access shared scene data, despite irregular access patterns.

Design implication

Gather shared data in batches and decouple memory orchestration from parallel computation.

Georgia Tech

Coarse-Grained Reconfigurable Architecture (CGRA)



CGRA: Coarse-Grained Reconfigurable Array

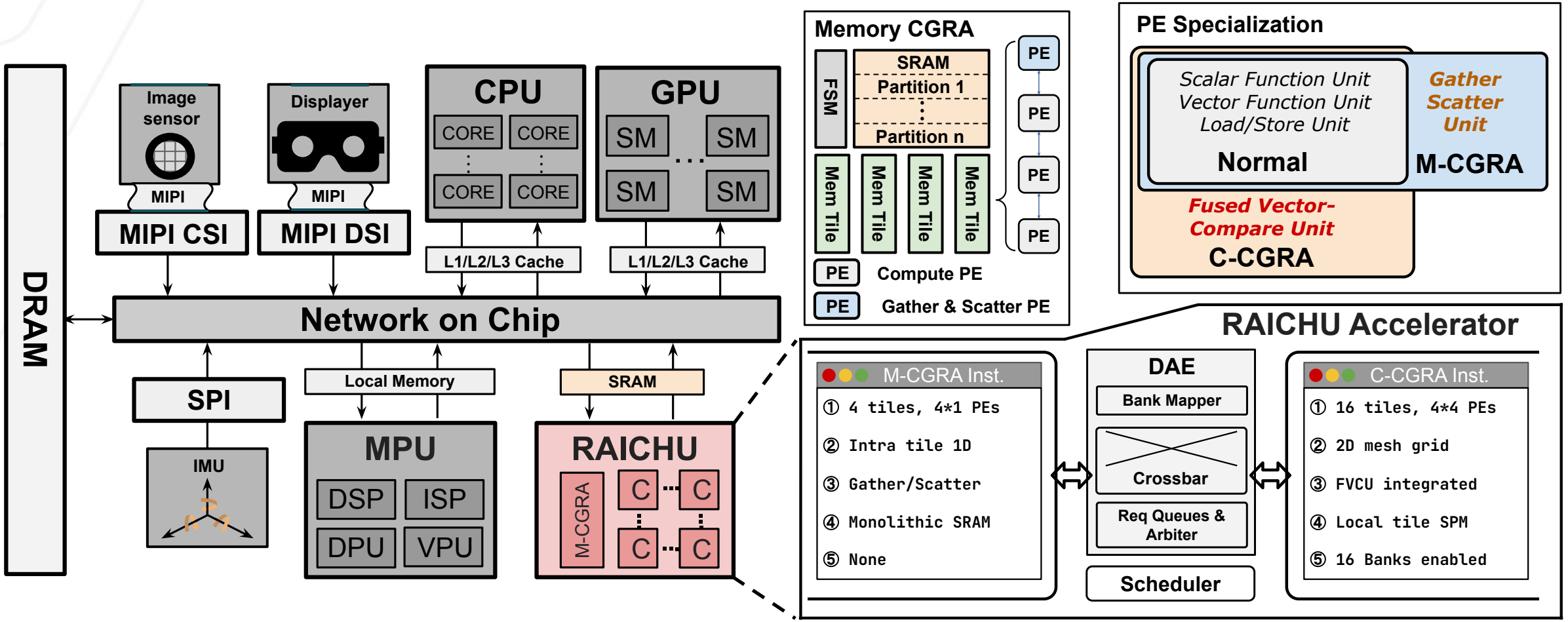
- PE level reconfigurable dataflow architecture.
- Trade-off between flexibility and energy efficiency.
- Available dedicated hardware module in specific filed.

Demands	Attributes
Support different rendering methods	Flexibility
Edge VR systems	Energy efficiency

Due to the balance between flexibility and energy efficiency, CGRAs are well-suited for accelerating and supporting various rendering methods.

Georgia Tech

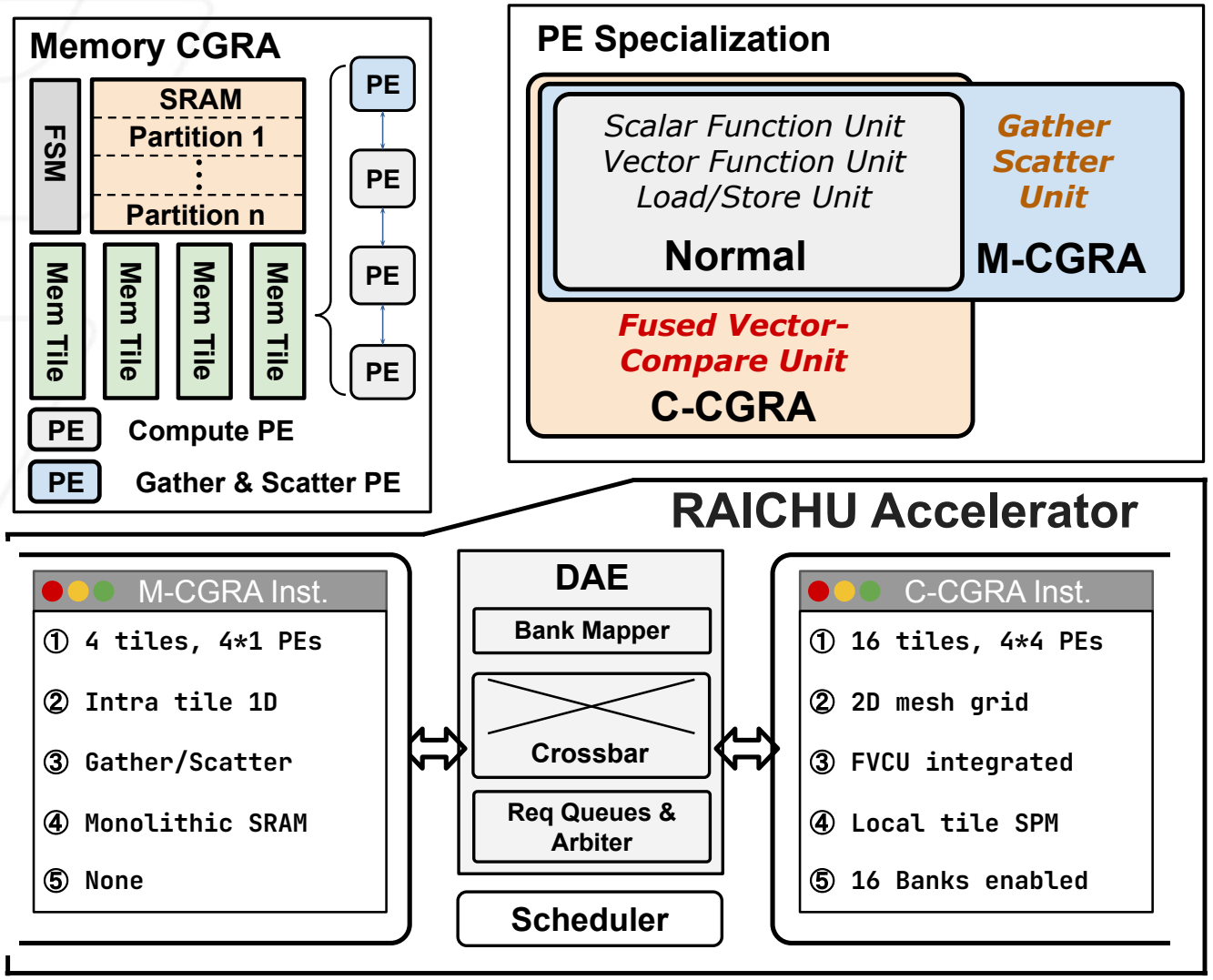
RAICHU architecture



RAICHU decouples memory orchestration from parallel computation.

Georgia Tech

RAICHU architecture



M-CGRA

Gather shared scene data and prepare batches.

DAE + shared buffers

Route prepared batches to the compute tiles.

C-CGRA

Execute regular rendering kernels in parallel.

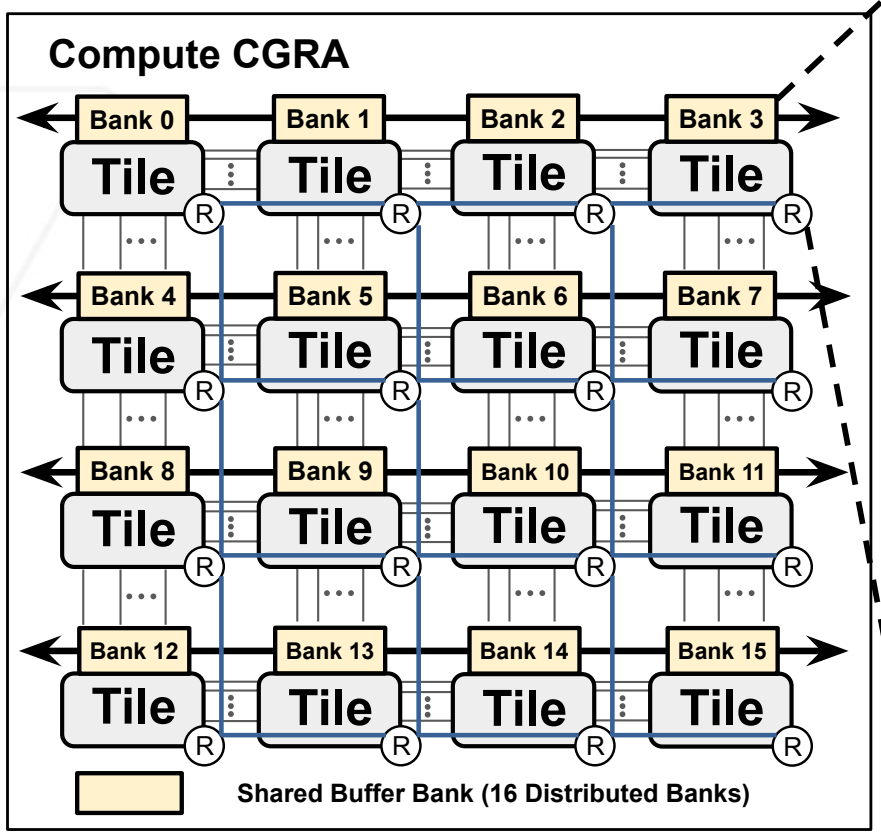
Overlap memory and compute

Georgia Tech

RAICHU architecture

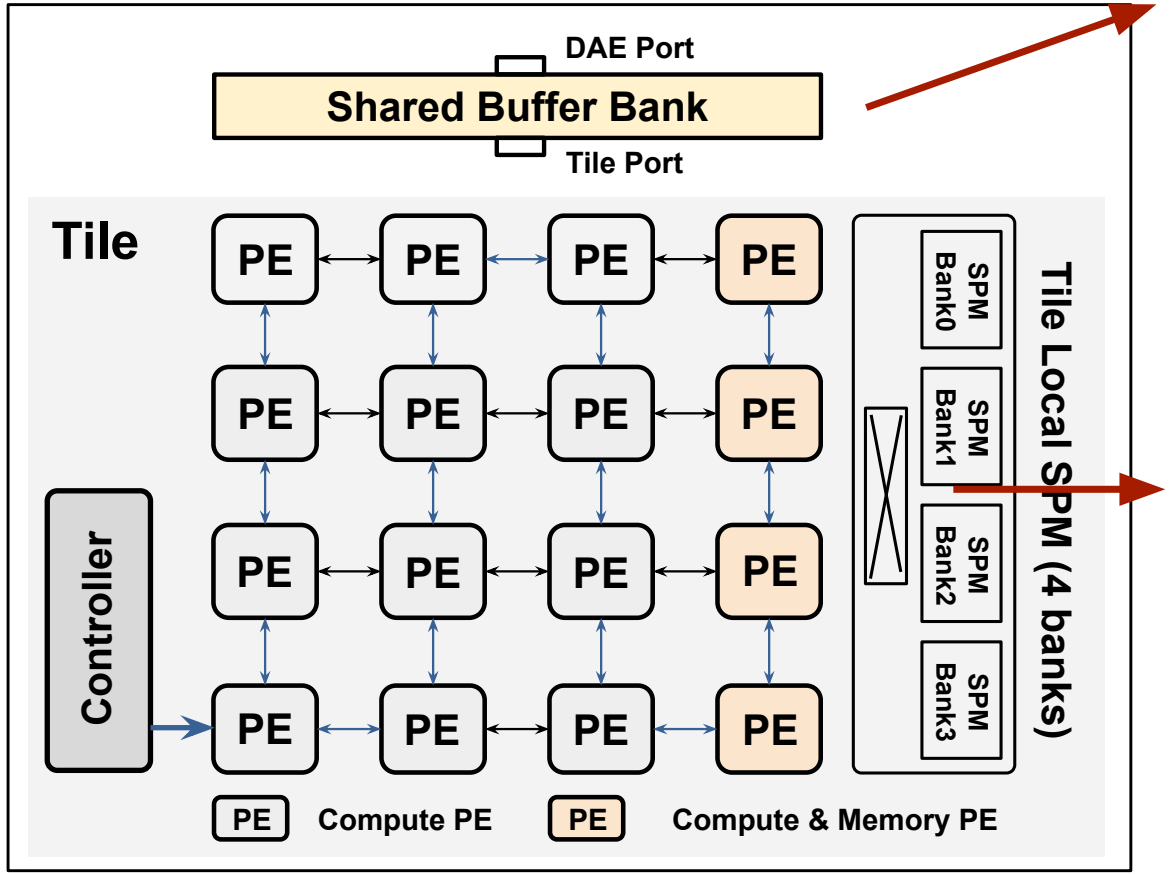
Configuration Template

① Array Dimension
② Network Topology
③ PE Specialization
④ Memory Hierarchy
⑤ Shared Buffering



Parallel execution

16 tiles × 16 PEs = 256 PEs



Shared buffering

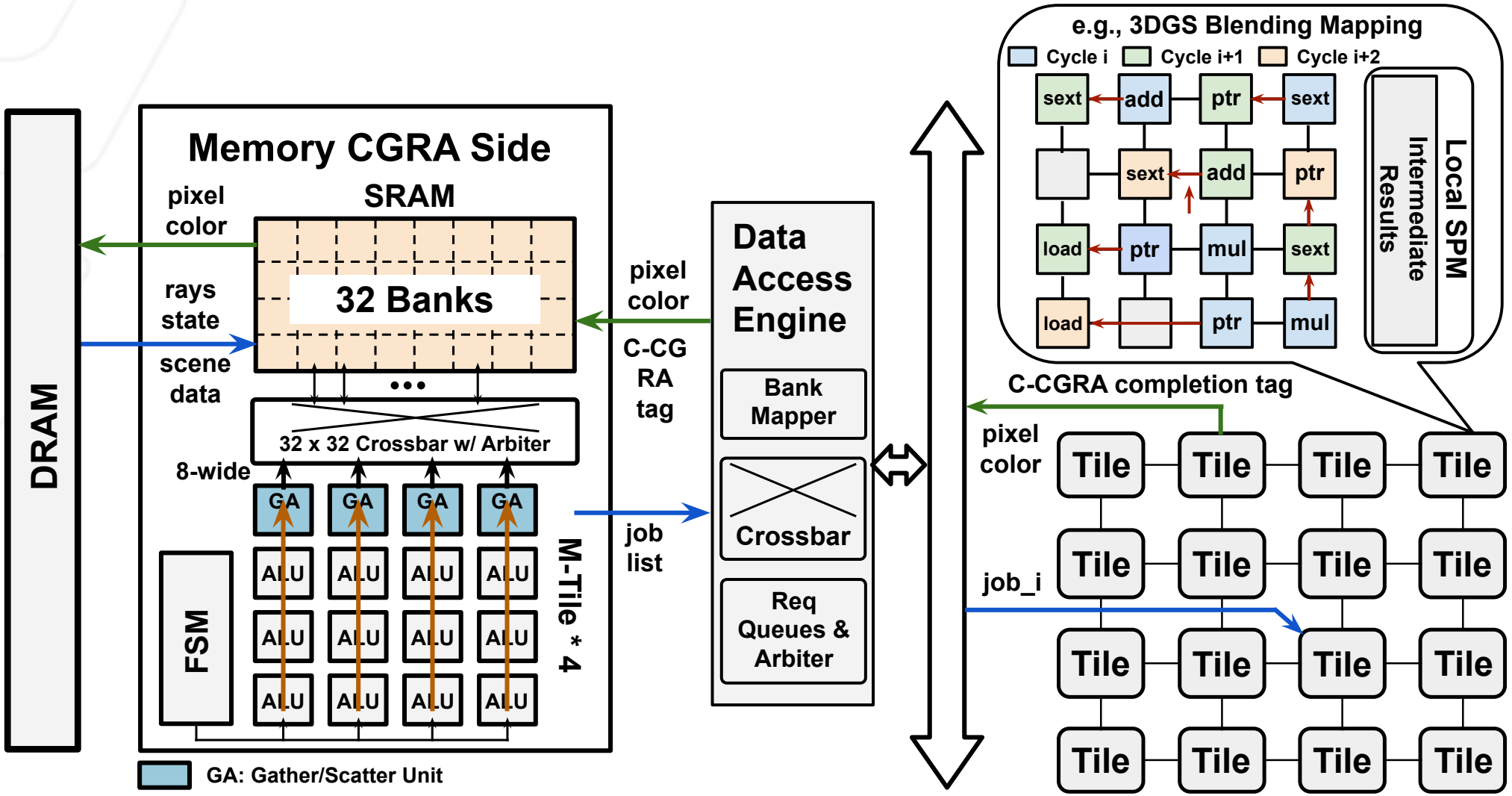
16 dual-port banks, one per tile

Local reuse

4 SPM banks store tile intermediates

Georgia Tech

RAICHU execution pipeline

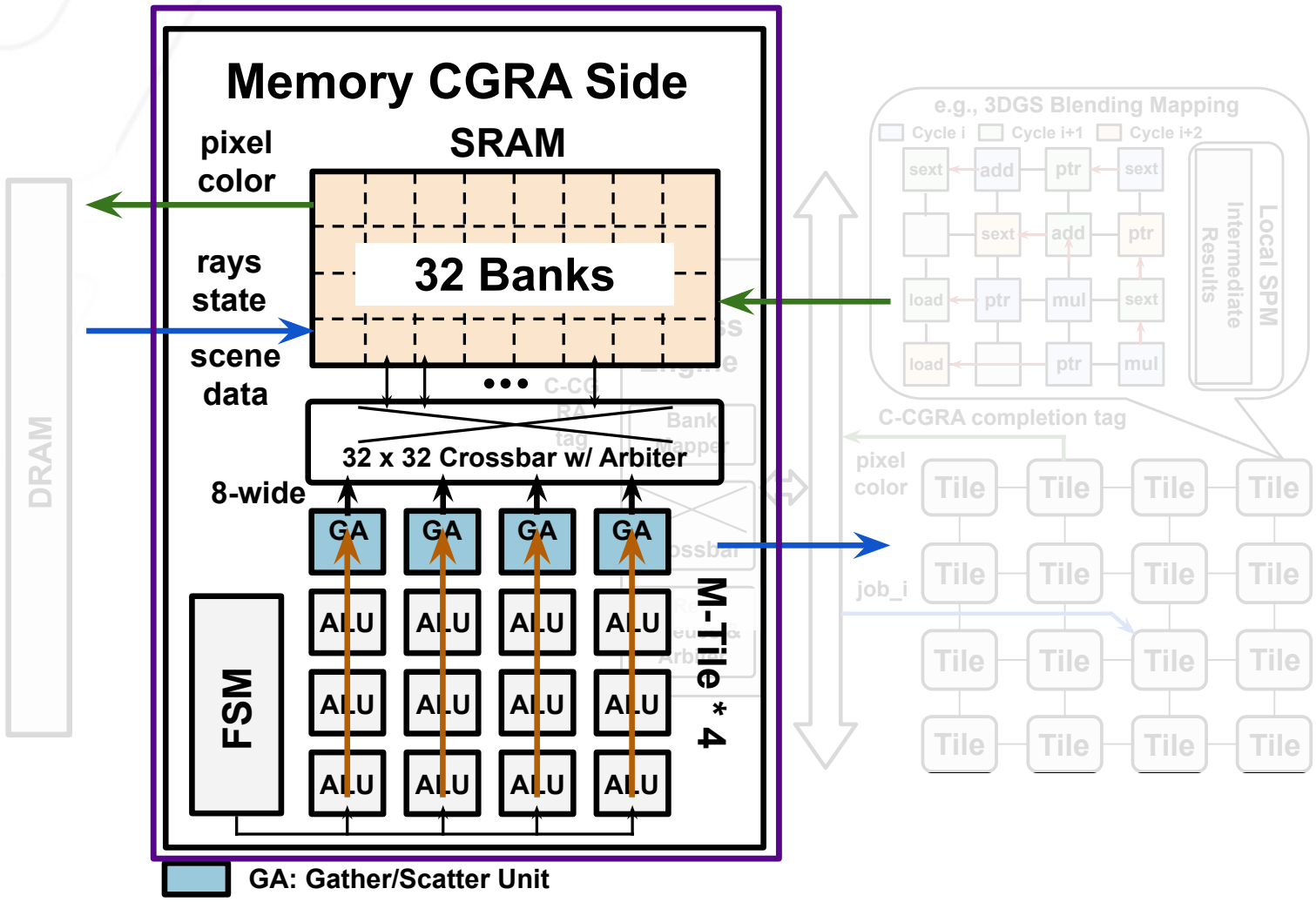


Overview

Irregular accesses become batched jobs for parallel compute.

Georgia Tech

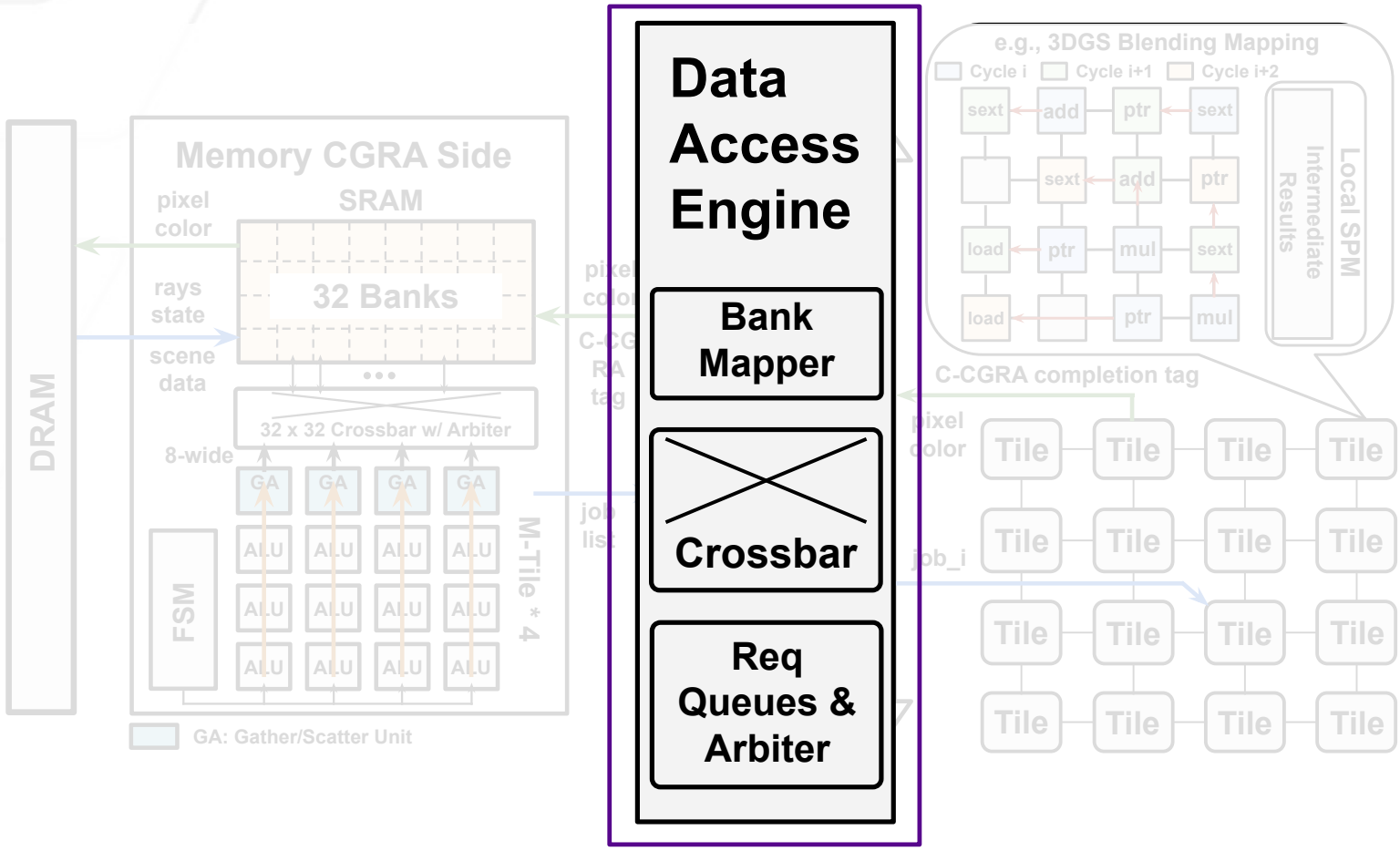
Gather on the M-CGRA



The M-CGRA gathers **shared scene primitives** and organizes them into **batched job lists**.

Georgia Tech

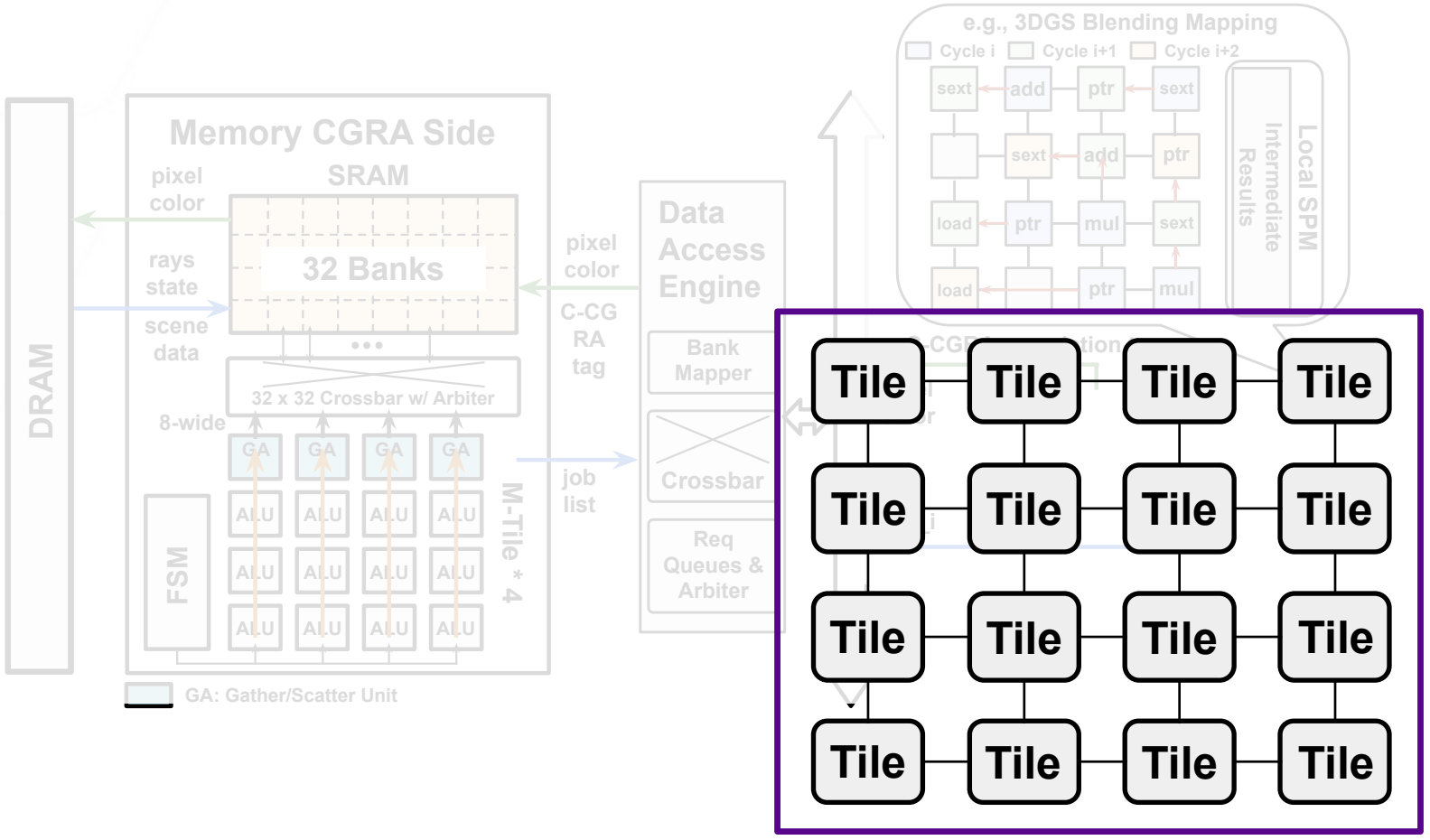
Dispatch through the DAE



The DAE maps jobs to **shared buffer banks** and routes them to the **C-CGRA tiles**.

Georgia Tech

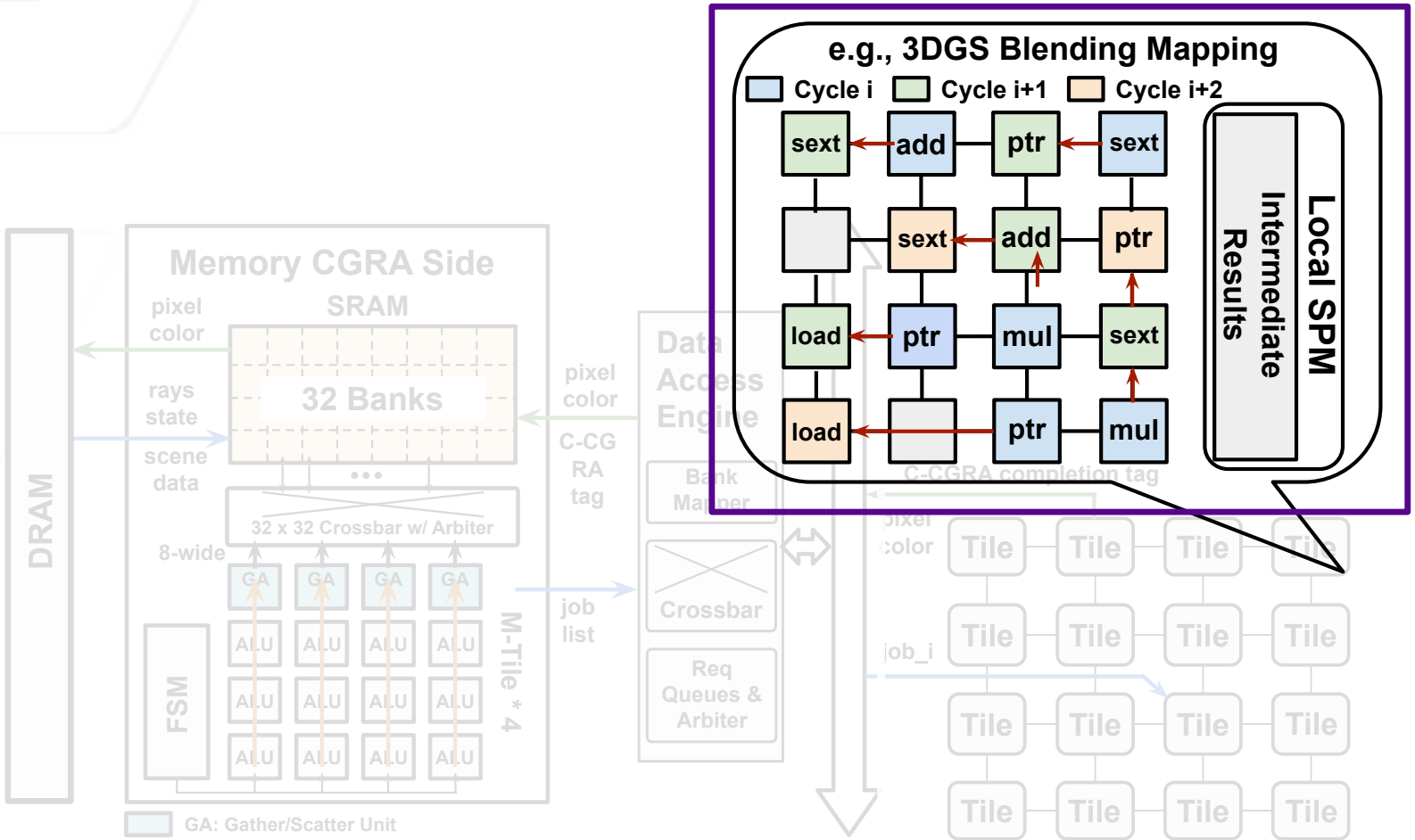
Execution on C-CGRA tiles



Tiles execute assigned jobs and return **intermediate results** or pixel colors with **completion tags** to the M-CGRA.

Georgia Tech

Mapping inside a C-CGRA tile



3DGS blending maps onto PEs across cycles **within one tile**. **Local SPM** stores intermediate results.

Georgia Tech

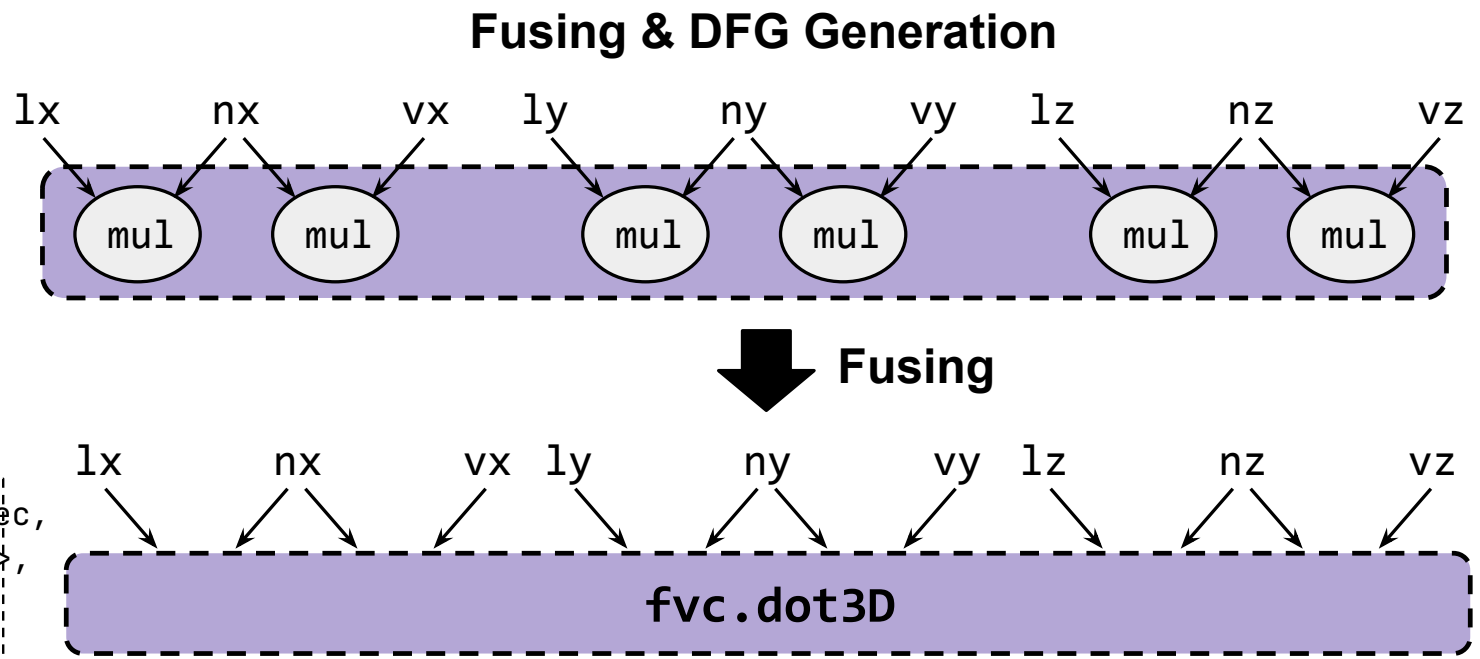
Fused vector arithmetic

```
%t0 = arith.mulf %nx, %lx : f32
%t1 = arith.mulf %ny, %ly : f32
%t2 = arith.mulf %nz, %lz : f32

%u0 = arith.mulf %nx, %vx : f32
%u1 = arith.mulf %ny, %vy : f32
%u2 = arith.mulf %nz, %vz : f32
```

Fusing

```
%ndotl, %ndotv = raichu.fvc.dot3D %n_vec,
%l_vec, %n_vec, %v_vec : (vector<3xf32>,
vector<3xf32>, vector<3xf32>,
vector<3xf32>) → (f32, f32)
```

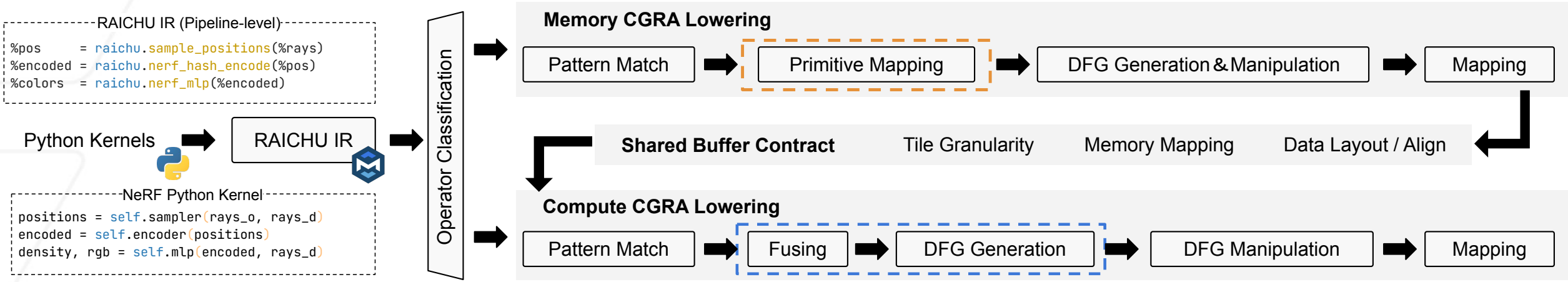


Paired 3D dot products, one FVCU primitive

The compiler fuses n · l and n · v into **fvc.dot3D** for a compact DFG.

Georgia Tech

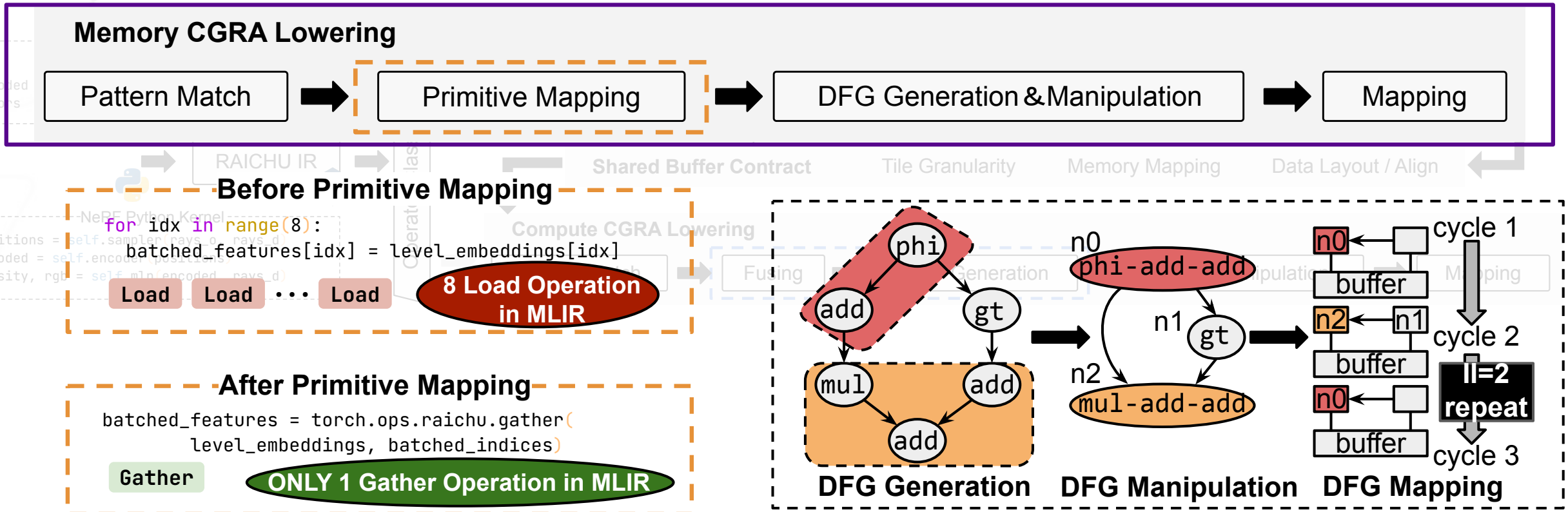
Two coordinated lowering paths



The compiler lowers rendering kernels through two paths, coordinated by an explicit shared buffer contract.

Georgia Tech

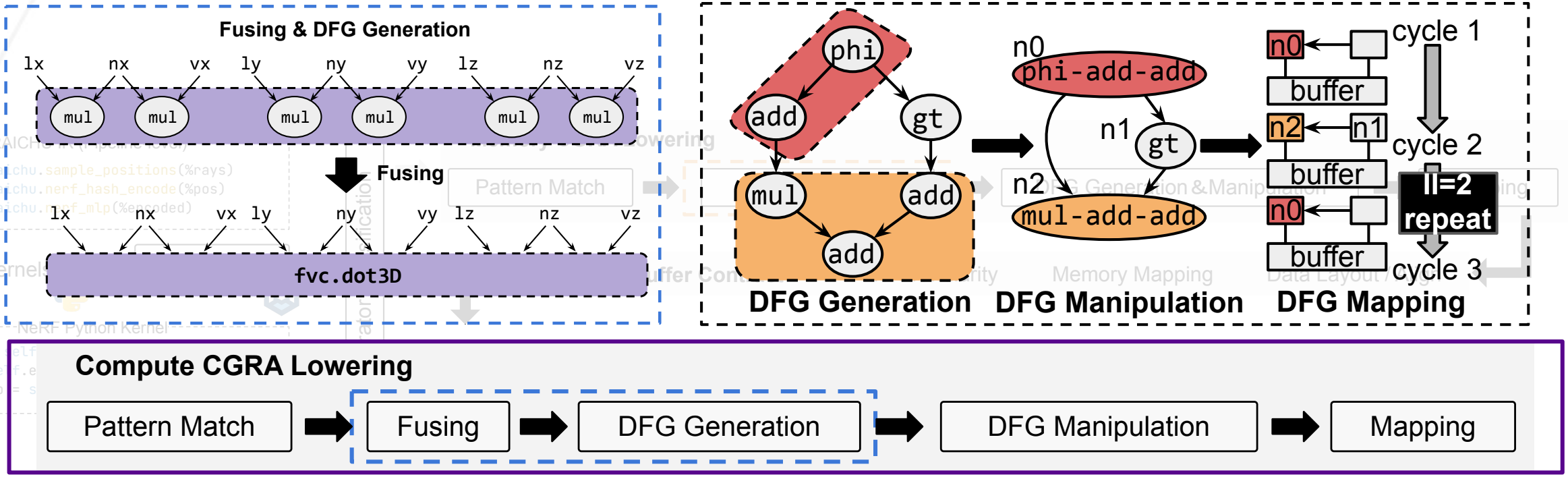
Memory CGRA lowering



Map gather/scatter to **memory primitives** and organize **batch-level dataflow** for the M-CGRA.

Georgia Tech

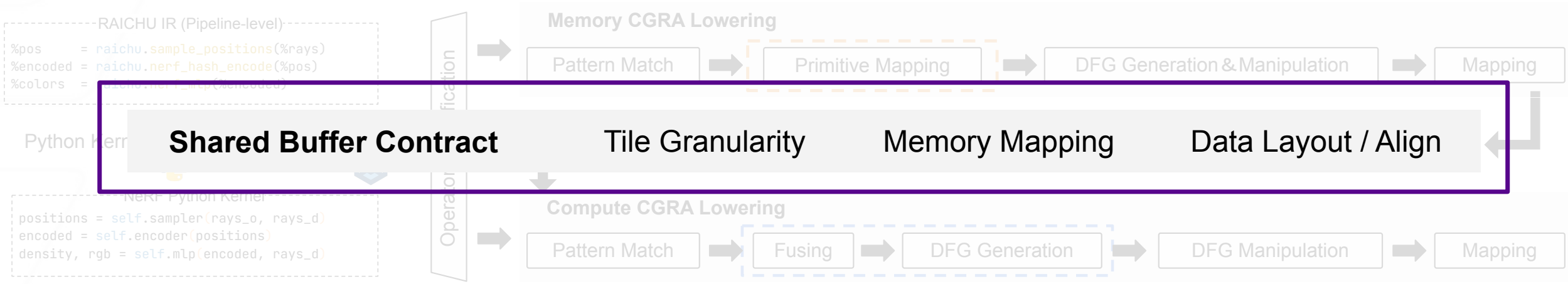
Compute CGRA lowering



Fuse arithmetic into **FVCU primitives** and map **compact DFGs** onto the C-CGRA.

Georgia Tech

Shared buffer contract



Coordinate **tile granularity**, **bank mapping**, and **data layout** so C-CGRA tiles can directly consume M-CGRA output.

Georgia Tech

Evaluation methodology

Five rendering pipelines

KiloNeRF, Instant-NGP, TensoRF, 3DGS, and ray tracing

Performance and correctness

Compiler-driven cycle-level simulation
RTL module and integration tests

Hardware configuration

C-CGRA: 256 PEs at 1 GHz
M-CGRA: 4 tiles × 4 PEs

Power & area

Component	Power	Area
PE Array	3.22 W	7.42 mm²
SRAM	0.93 W	2.95 mm²
CGRA Ctrl	0.17 W	0.28 mm²
M-CGRA	0.70 W	1.82 mm²
Total	**5.02 W**	**12.5 mm²**
Process	45 nm	

Georgia Tech

Performance speedup

Platform	Work	M-NeRF	G-NeRF	L-NeRF	3DGS	RT
EGP	Orin NX	33.92×	109.65×	5.67×	16.68×	7.18×
	Nano	135.59×	175.40×	7.43×	19.36×	288.02×
VRA	Uni-Render	4.85×	1.32×	4.36×	1.85×	-
	FlexNeRFer	1.88×	6.09×	0.31×	-	-
ASRA	-	MetaVRain 0.44×	Instant-3D **7.92×**	RT-NeRF **13.08×**	GSCore 1.01×	AQB8 **3.95×**

EGP: Edge GPU platform VRA: Versatile rendering accelerator
ASRA: Application-specialized rendering accelerator

Georgia Tech

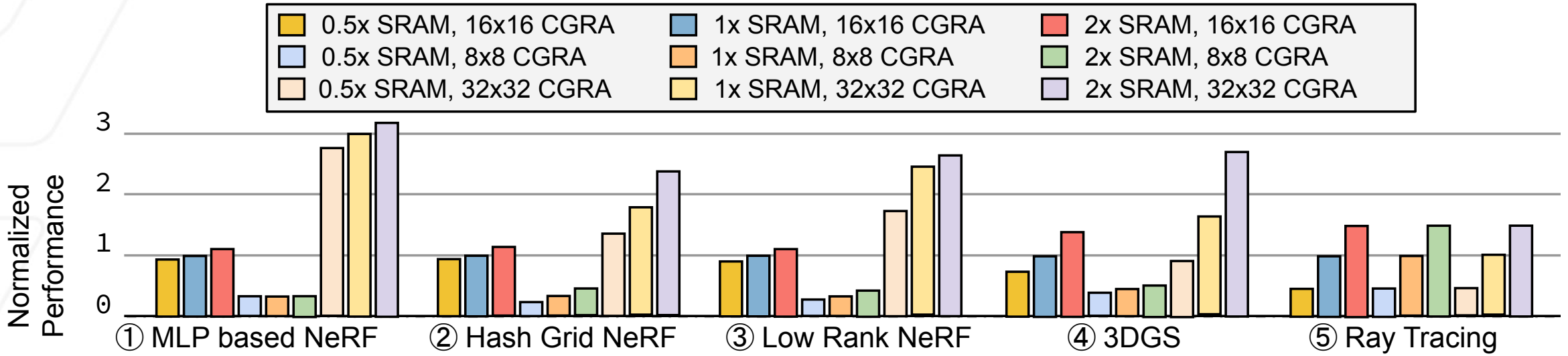
Energy efficiency improvement

Platform	Work	M-NeRF	G-NeRF	L-NeRF	3DGS	RT
EGP	Orin NX	87.87×	151.60×	10.83×	20.78×	17.58×
	Nano	428.78×	375.70×	19.67×	38.54×	883.01×
VRA	Uni-Render	7.77×	2.12×	6.99×	2.97×	-
	FlexNeRFer	3.81×	12.33×	0.63×	-	-
ASRA	-	MetaVRain 0.16×	Instant-3D **4.17×**	RT-NeRF **27.18×**	GSCore 0.25×	AQB8 **11.43×**

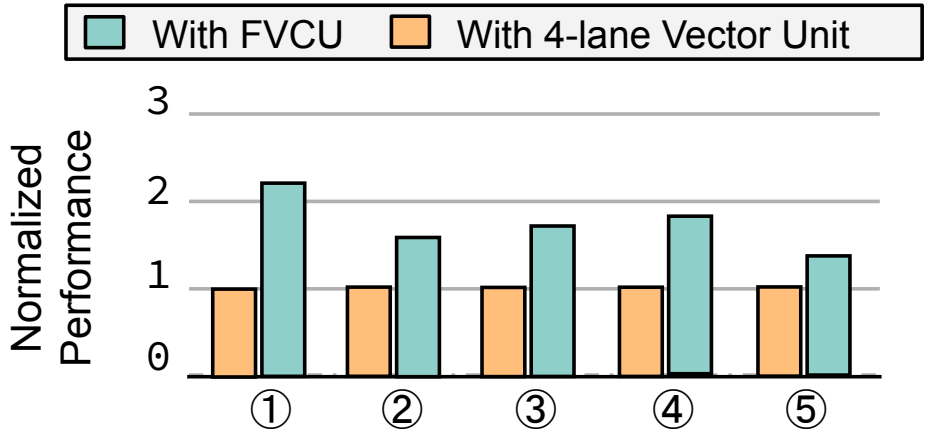
EGP: Edge GPU platform VRA: Versatile rendering accelerator
ASRA: Application-specialized rendering accelerator

Georgia Tech

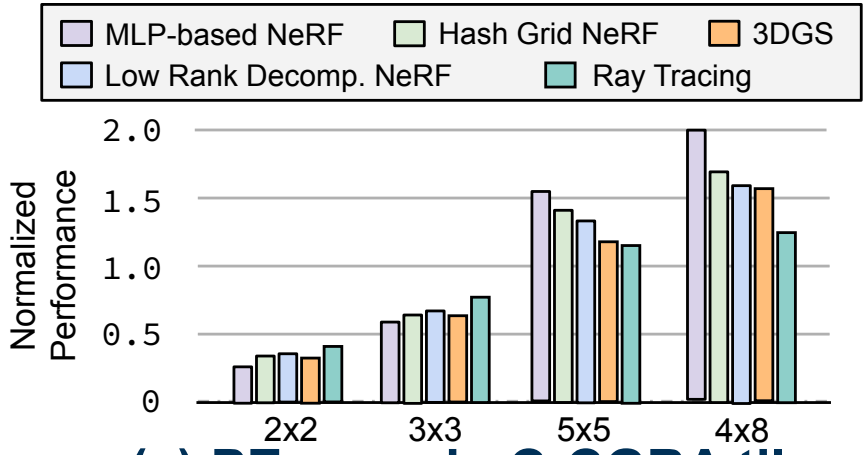
Memory and compute balance



(a) Balance with varying M-CGRA SRAM capacity and C-CGRA array size



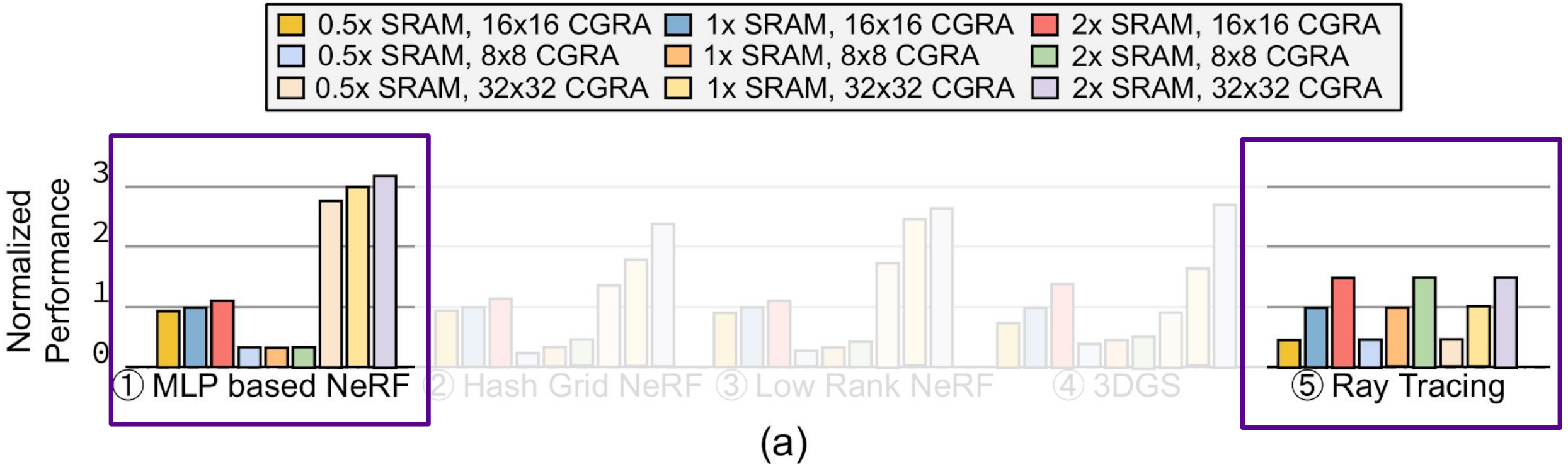
(b) FVCU v.s vector unit



(c) PE array in C-CGRA tile

Georgia Tech

Memory and compute scaling



MLP NeRF

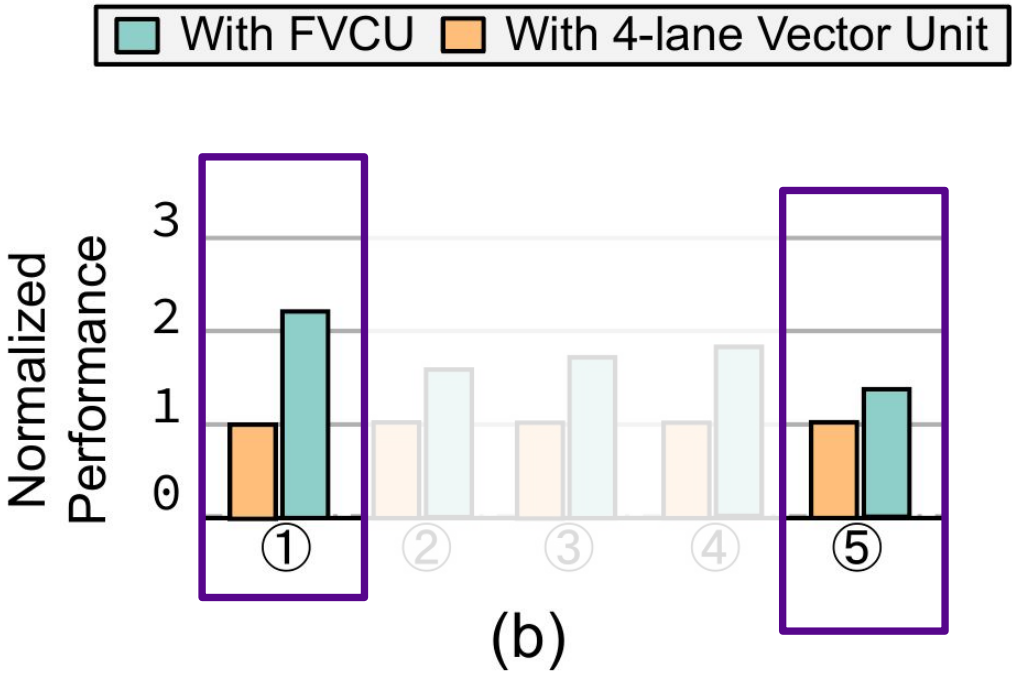
Larger C-CGRA arrays help.
M-side changes have little effect.

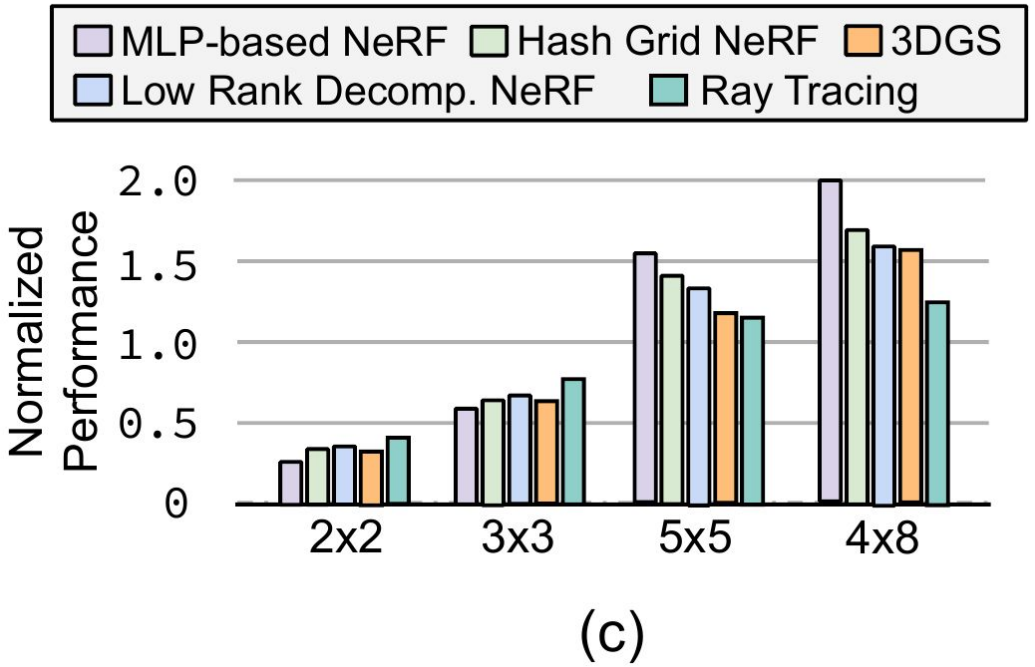
Ray tracing

M-side SRAM / bandwidth helps.
Extra compute adds little.

Georgia Tech

Compute support and tile scaling





FVCU replacement study

MLP NeRF: 2.2×; RT: 1.3×.
Six vs four lanes, not area matched.

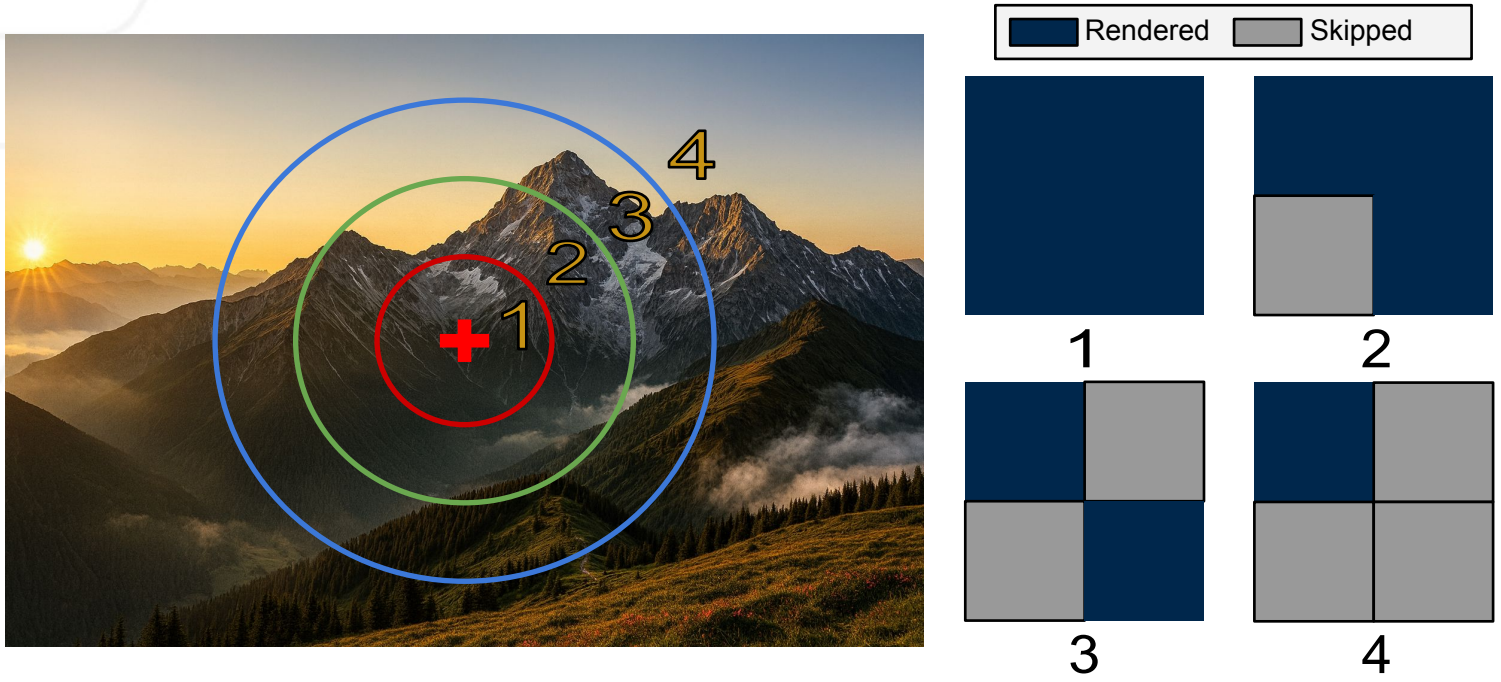
PE array scaling

With 16 tiles, larger PE arrays benefit MLP NeRF most.

Georgia Tech

Foveated rendering algorithm

Sampling density decreases away from the gaze point



Foveated rendering

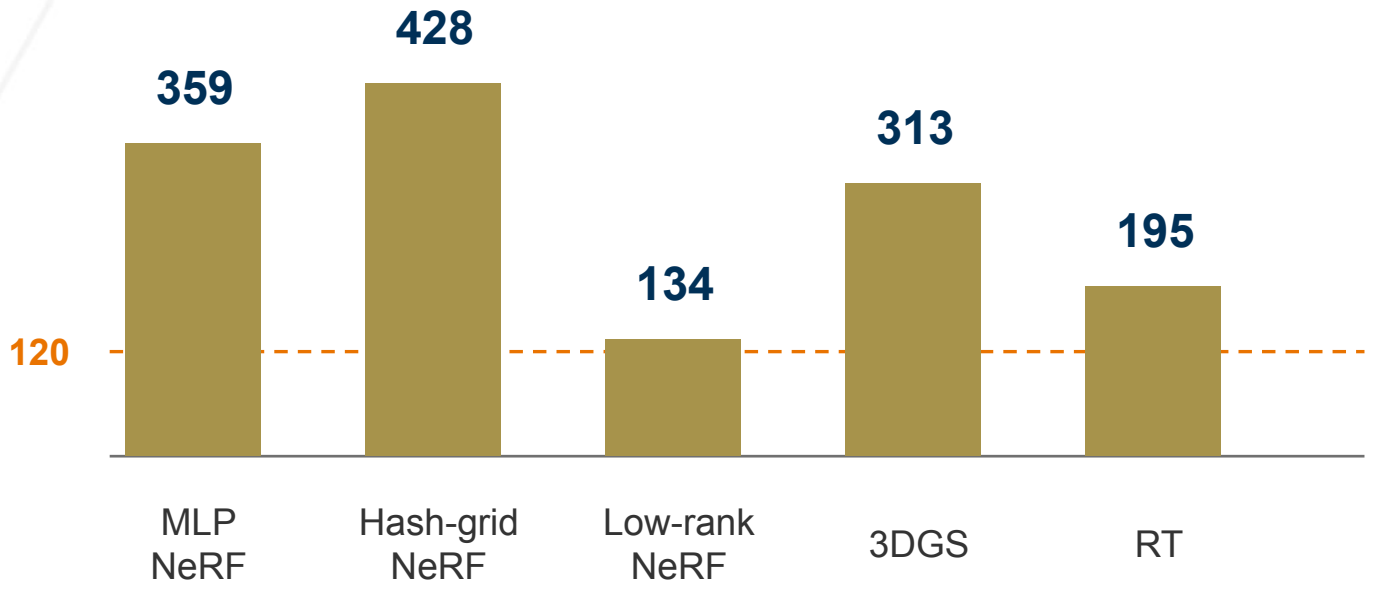
- Use three concentric circles to partition the image into four regions of varying resolutions.
- For lower-resolution regions, perform 2×2 block upsampling.
- A unified foveated rendering method for 3DGS, NeRF, and Ray Tracing.

Georgia Tech

Foveated rendering results

Sampling density decreases away from the gaze point

Rendering throughput (FPS)



3DGS image study

48.2%

RAICHU preferred (50% chance)

7 participants, 4 static scenes
Central fixation

Reordering + quantization

≤ 0.38 dB

Max drop in four PSNR comparisons

Georgia Tech

Unified rendering with RAICHU

A shared execution structure across NeRF, 3DGS, and ray tracing

Code Repo

Full paper in personal website



5 rendering pipelines

One programmable architecture

Up to 109.65× speedup

vs Jetson Orin NX

Georgia Tech