GainSight: Application-Guided Profiling for Composing Heterogeneous On-Chip Memories in Next-Generation AI Accelerators

Peijing Li, peli@stanford.edu

Thierry Tambe, ttambe@stanford.edu

July 2, 2025

Stanford | ENGINEERING
Electrical Engineering

At a Glance

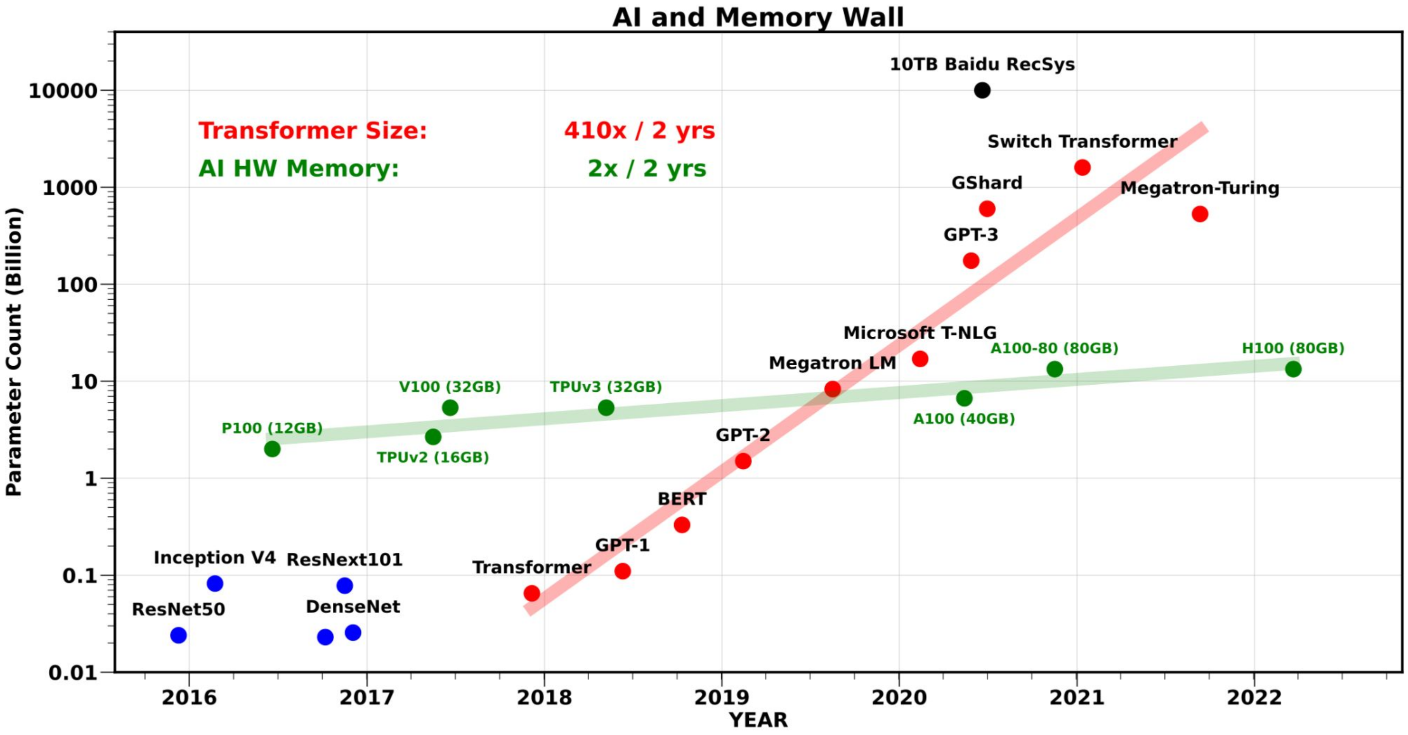
1 Motivation for Differentiated Access Memories and Fine-Grained Data Cache Access Pattern Profiling

2 Methodologies: Retargetable Hardware Backend and Flexible Analytical Frontend

3 Preliminary Experiments and Results

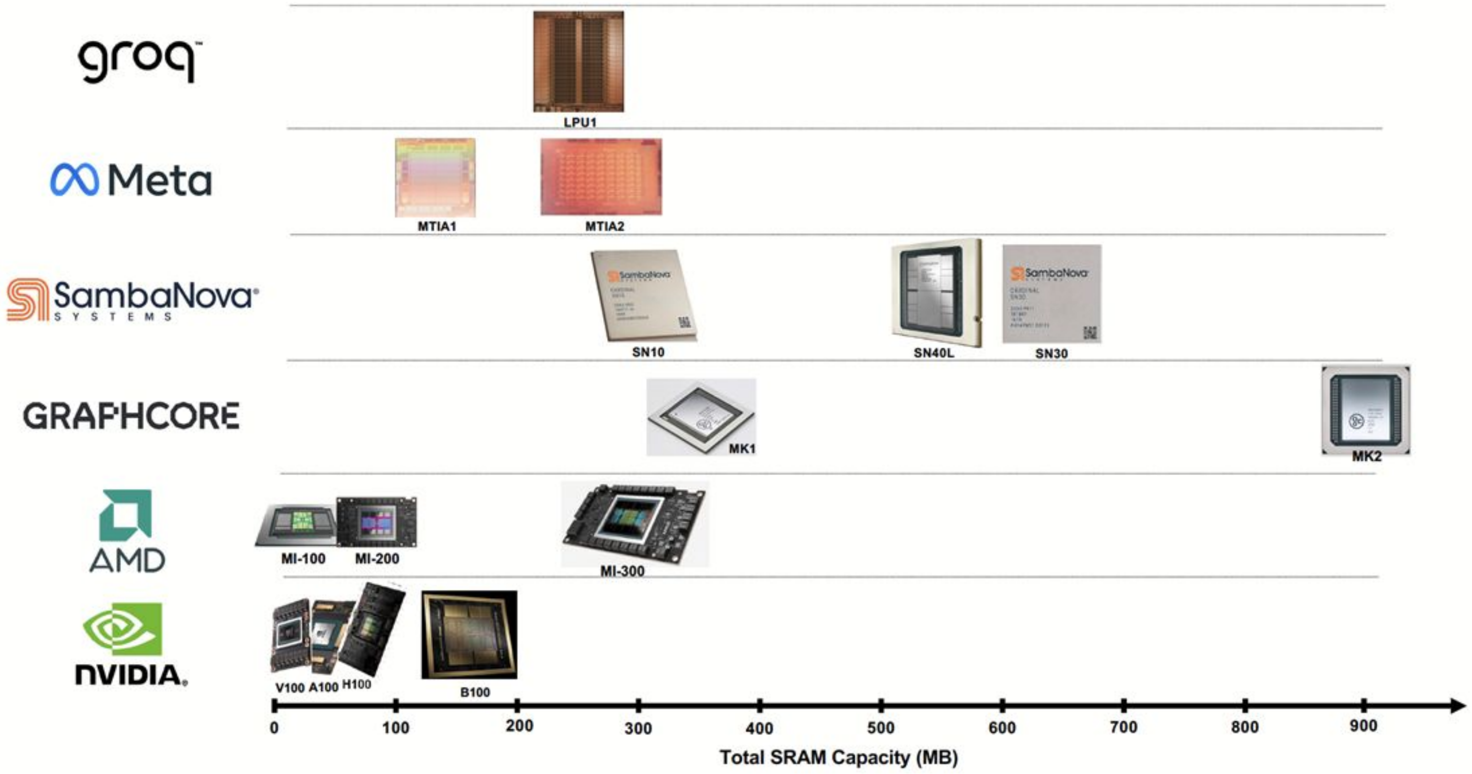
Motivations for Fine-Grained On-Chip Memory Profiling

AI and the Memory Capacity Wall

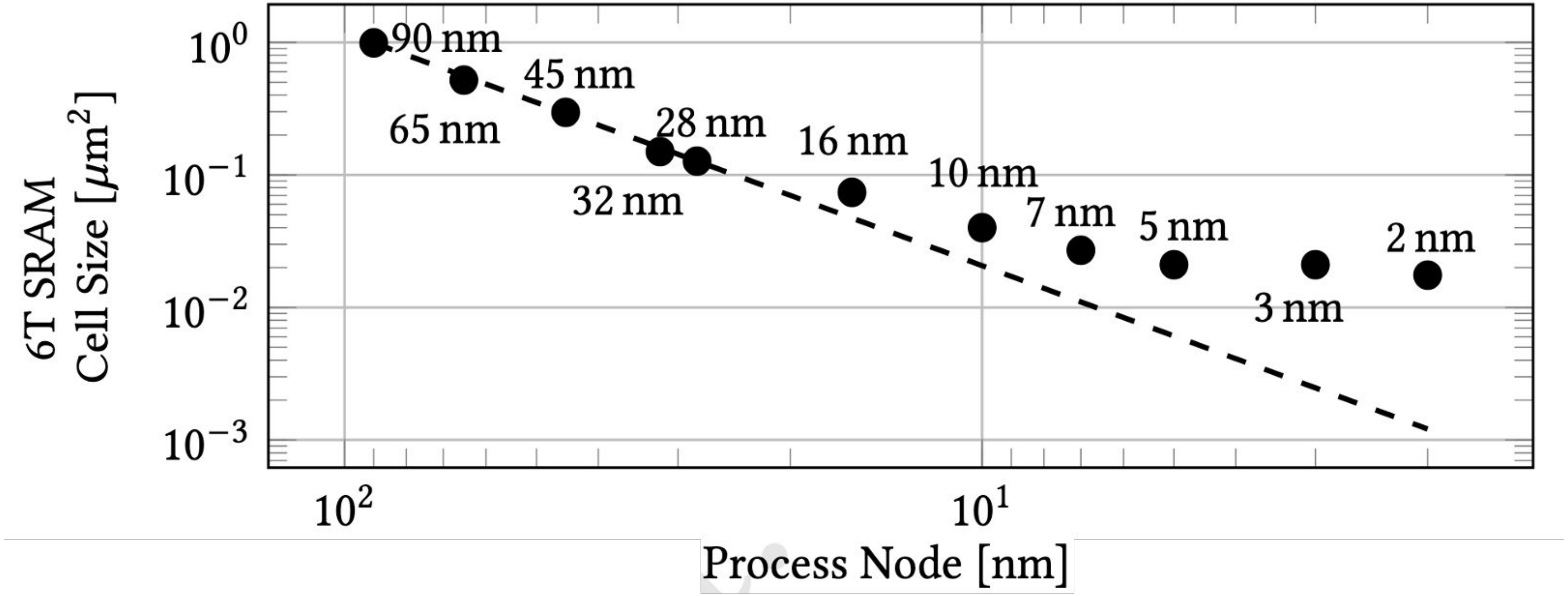


Source: [1] A. Gholami, Z. Yao, S. Kim, C. Hooper, M. W. Mahoney, and K. Keutzer, “AI and Memory Wall,” Mar. 21, 2024, arXiv: arXiv:2403.14123. doi: 10.48550/arXiv.2403.14123.

Trend towards Increasing On-Chip SRAM



SRAM Scaling is Plateauing



Source: [1] K. Zhang, "1.1 Semiconductor Industry: Present & Future," in 2024 IEEE International Solid-State Circuits Conference (ISSCC), Feb. 2024, pp. 10–15. doi: 10.1109/ISSCC49657.2024.10454358.

SRAM is over-provisioned for AI Accelerators

- Empirical observation: **short- and long-lived data** in AI models
  - Short-lived data: activation values, frequently modified
  - Long-lived data: weight values, frequently read, rarely modified
- SRAM offers great read/write latency, endurance and data retention at the cost of area density and static power
- Its performance may be **over-provisioned** for both short- and long-lived AI/ML model data
- Alternative devices can make trade-offs to attain higher density and lower power

Source: [1] S. Legtchenko et al., "Managed-Retention Memory: A New Class of Memory for the AI Era," Jan. 16, 2025, arXiv: arXiv:2501.09605. doi: 10.48550/arXiv.2501.09605.

Alternative On-Chip Memories

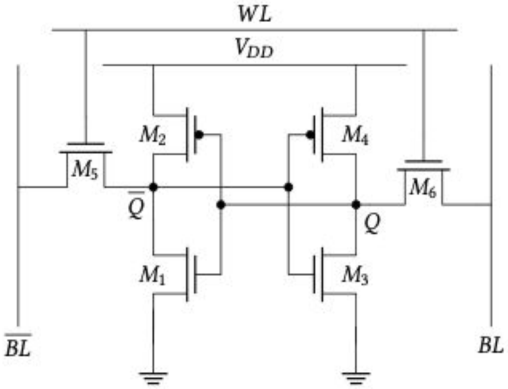
	SRAM	DRAM	Block Flash	Long-term RAM	Short-term RAM
Structure	6T	1T1C	1G	FeRAM, MRAM, RRAM, FRAM	2T or 3T gain cells
Benefits	Fast, easy to integrate, low static power	Dense	Huge capacity	Dense, low read energy	Dense, low energy
Drawbacks	Sparse	No logic, high power	No logic, low endurance, expensive & slow erases, block access only, low bandwidth	Expensive & slow writes, limited endurance	Short retention times, expensive refreshes, active research
Uses	Fast R/W caches	Large, random access R/W data	Large, mostly read data	Rare writes, static data caches	Fast write-and-read operations

Alternative On-Chip Memories

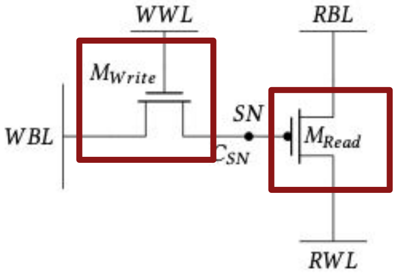
	SRAM	DRAM	Block Flash	Long-term RAM	Short-term RAM
Structure	6T	1T1C	1G	FeRAM, MRAM, RRAM, FRAM	2T or 3T gain cells
Benefits	Fast, easy to integrate, low static power	Dense	Huge capacity	Dense, low read energy	Dense, low energy
Drawbacks	Sparse	No logic, high power	No logic, low endurance, expensive & slow erases, block access only, low bandwidth	Expensive & slow writes, limited endurance	Short retention times, expensive refreshes, active research
Uses	Fast R/W caches	Large, random access R/W data	Large, mostly read data	Rare writes, static data caches	Fast write-and-read operations

Alternative On-Chip Memories

	SRAM	Short-term RAM
Structure	6T	2T or 3T gain cells
Benefits	Fast, easy to integrate, low static power	Dense, low energy
Drawbacks	Sparse	Short retention times, expensive refreshes, active research
Uses	Fast R/W caches	Fast write-and-read operations



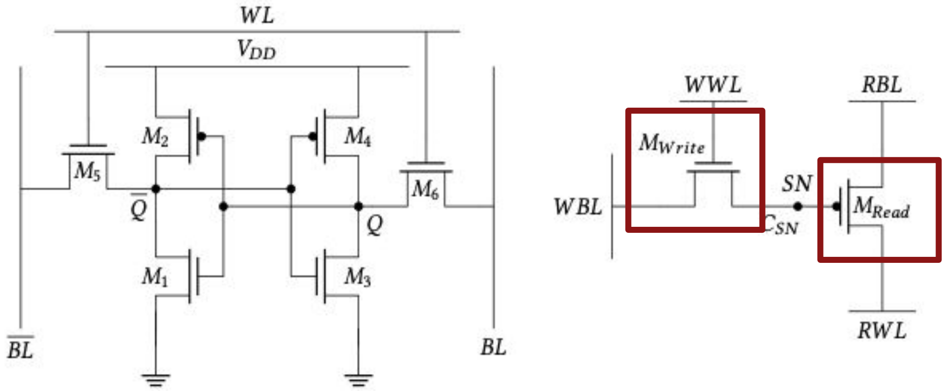
(a) 6T SRAM



(b) 2T GCRAM

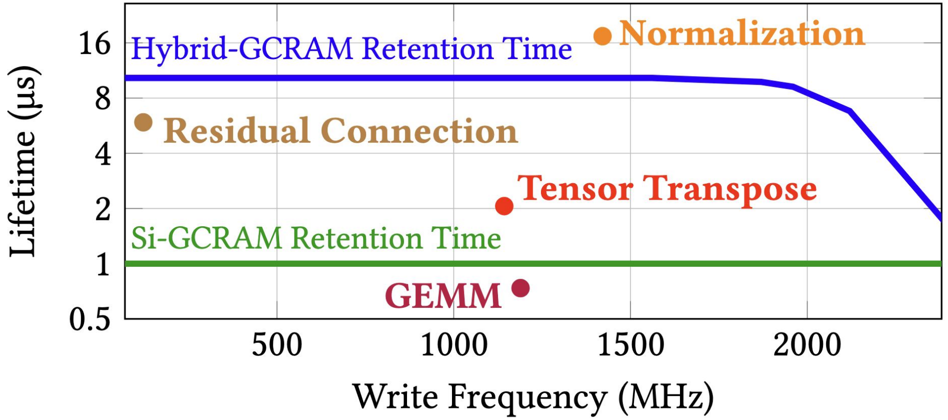
Alternative On-Chip Memories

	SRAM	Short-term RAM
Structure	6T	2T or 3T gain cells
Benefits	Fast, easy to integrate, low static power	Dense, low energy
Drawbacks	Sparse	Short retention times, expensive refreshes, active research
Uses	Fast R/W caches	Fast write-and-read operations

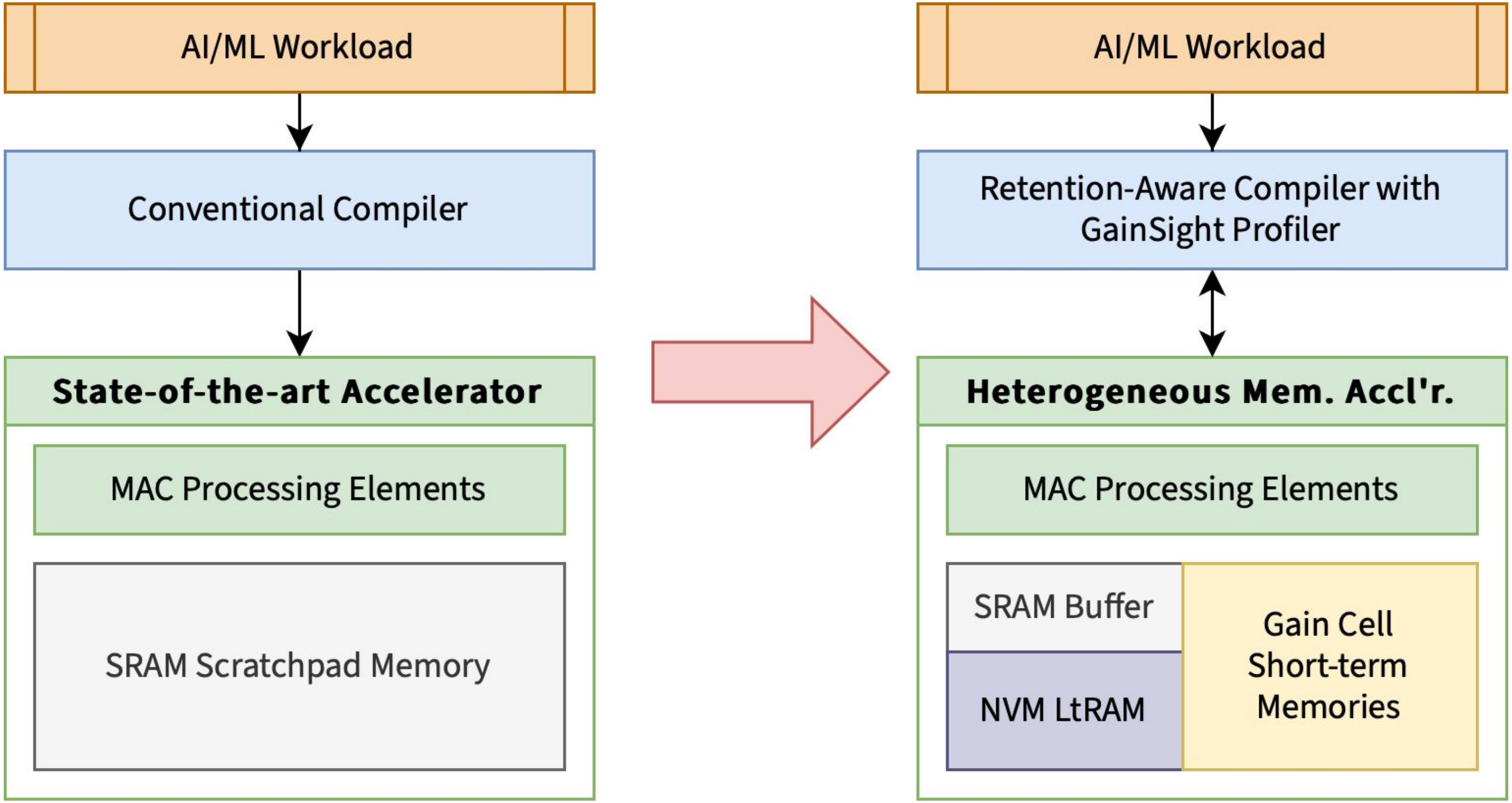


(a) 6T SRAM

(b) 2T GCRAM



Our Vision



AI/ML Workload

Conventional Compiler

State-of-the-art Accelerator

MAC Processing Elements

SRAM Scratchpad Memory

AI/ML Workload

Retention-Aware Compiler with GainSight Profiler

Heterogeneous Mem. Accl'r.

MAC Processing Elements

SRAM Buffer

NVM LtRAM

Gain Cell Short-term Memories

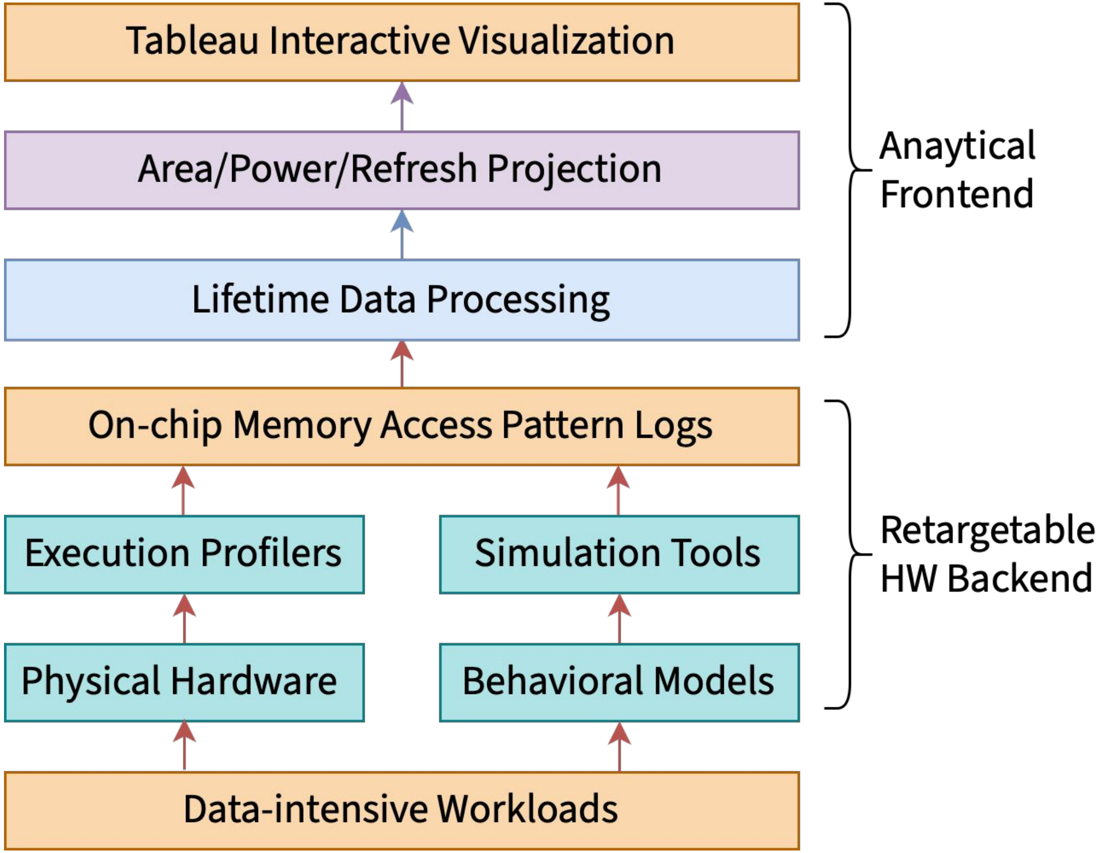
Our Vision

- Research questions
  - *Augment existing accelerators* by replacing SRAM with heterogeneous arrays featuring gain-cell RAM (GCRAM)
  - Understand how to provision specific GCRAM technologies and how workloads interact with them
- **“Memory profile-guided HW-SW codesign”**
  - ***GainSight***: profiler to offer **insights** in gain cell device design
  - “What would happen if we replace on-chip SRAM with gain cells?”
- Build a **profiler** that can measure **lifetimes** and other **fine-grained** memory access **patterns**; Output **projections in area & power** for different devices to understand their benefits over SRAM for a workload

GainSight Organization

GainSight High-Level Organization

- **Hardware Backend**
  - Measure fine-grained on-chip memory access patterns
  - Variety of processing elements
- **Analytical Frontend**
  - Calculate lifetimes & other memory statistics
  - Utilize domain knowledge to project area/power/refresh of gain cell devices
  - One frontend to work with multiple backends



Retargetable Hardware Backends

- Acquire fine-grained, **on-chip memory access patterns** for a physical or simulated processing element
- **Simulators** for C++/SystemC/RTL models
  - NVIDIA GPUs: Accel-Sim and GPGPU-Sim
  - Systolic arrays: SCALE-Sim-v2
- Sampling workloads to reduce runtime for simulations
- Designed for **extensibility**

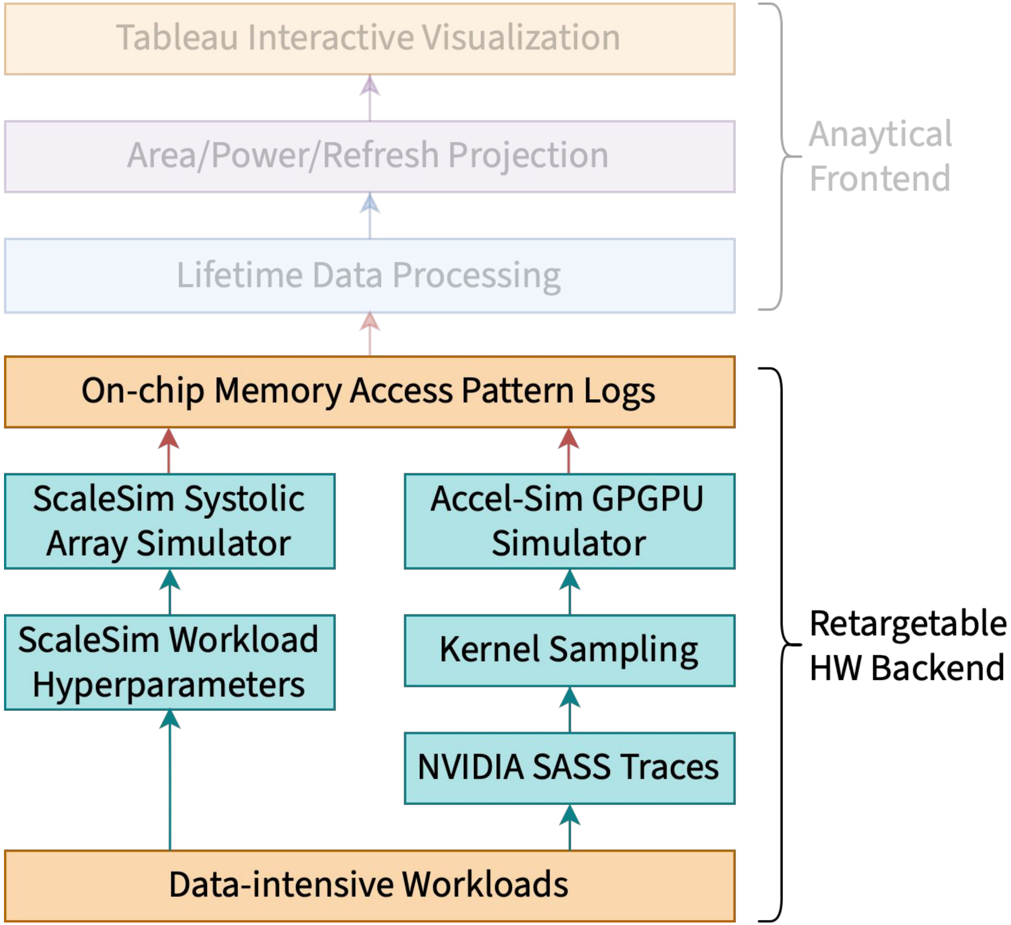


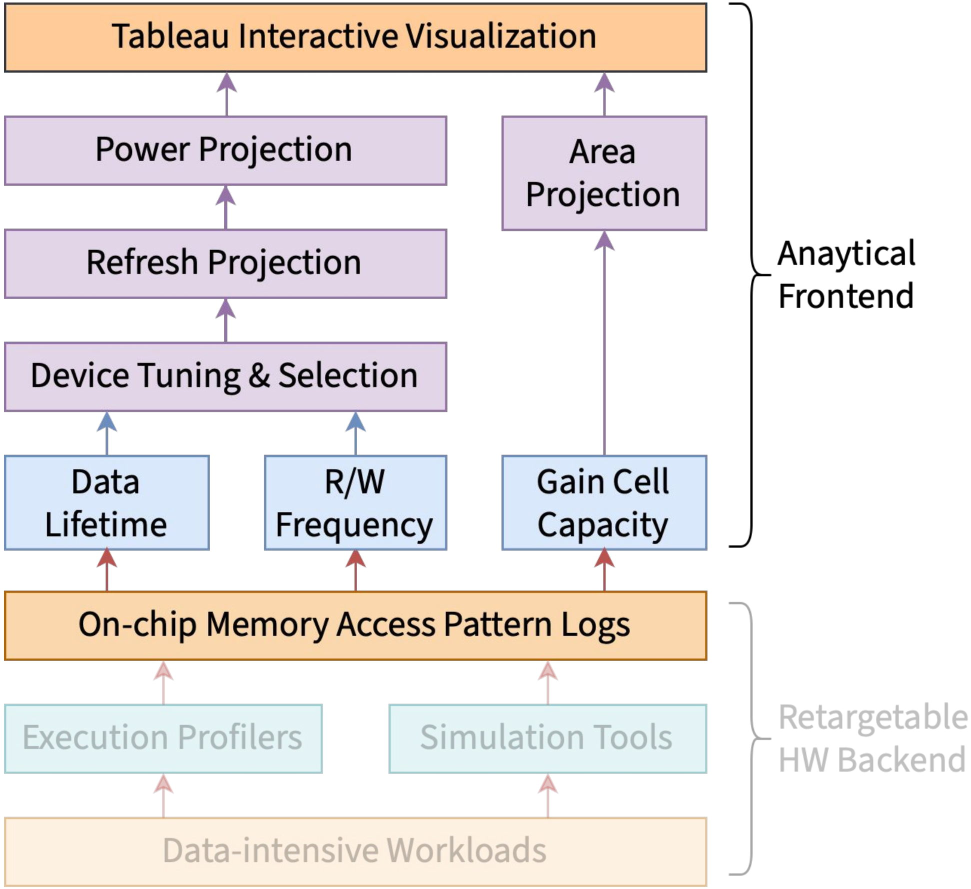
Tableau Interactive Visualization
Area/Power/Refresh Projection
Lifetime Data Processing
Anaytical Frontend
On-chip Memory Access Pattern Logs
ScaleSim Systolic Array Simulator
Accel-Sim GPGPU Simulator
ScaleSim Workload Hyperparameters
Kernel Sampling
NVIDIA SASS Traces
Data-intensive Workloads
Retargetable HW Backend

Stanford | ENGINEERING
Electrical Engineering

[1] A. Samajdar, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, “SCALE-Sim: Systolic CNN Accelerator Simulator,” Feb. 02, 2019, arXiv: arXiv:1811.02883. doi: 10.48550/arXiv.1811.02883.
[2] M. Khairy, Z. Shen, T. M. Aamodt, and T. G. Rogers, “Accel-Sim: An Extensible Simulation Framework for Validated GPU Modeling,” in 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA), May 2020, pp. 473–486. doi: 10.1109/ISCA45697.2020.00047.
[3] C. Avalos Baddouh, M. Khairy, R. N. Green, M. Payer, and T. G. Rogers, “Principal Kernel Analysis: A Tractable Methodology to Simulate Scaled GPU Workloads,” in MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture, in MICRO ’21. New York, NY, USA: Association for Computing Machinery, Oct. 2021, pp. 724–737. doi: 10.1145/3466752.3480100.

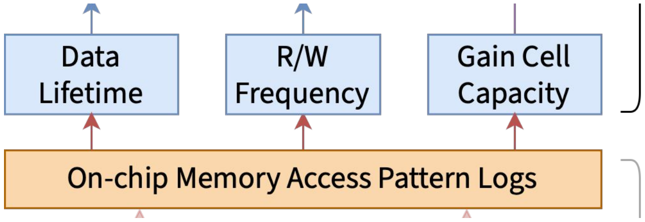
Analytical Frontend

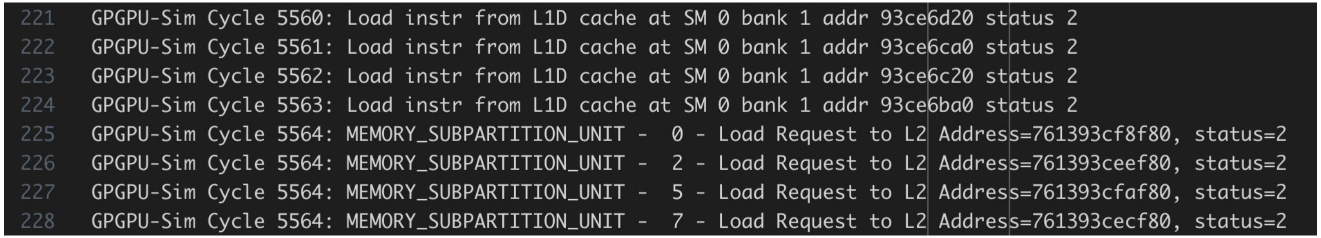
- **Translate runtime statistics into hardware implications**
- Use on-chip memory access pattern logs as input
- Extract on-chip memory stats
- Performance projection
- Interactive visualization



[1] S. Liu et al., "Design Guidelines for Oxide Semiconductor Gain Cell Memory on a Logic Platform," IEEE Transactions on Electron Devices, vol. 71, no. 5, pp. 3329–3335, May 2024, doi: 10.1109/TED.2024.3372938.
[2] R. Giterman, A. Shalom, A. Burg, A. Fish, and A. Teman, "A 1-Mbit Fully Logic-Compatible 3T Gain-Cell Embedded DRAM in 16-nm FinFET," IEEE Solid-State Circuits Letters, vol. 3, pp. 110–113, 2020, doi: 10.1109/LSSC.2020.3006496.
[3] S. Liu et al., "Gain Cell Memory on Logic Platform – Device Guidelines for Oxide Semiconductor Transistor Materials Development," in 2023 International Electron Devices Meeting (IEDM), Dec. 2023, pp. 1–4. doi: 10.1109/IEDM45741.2023.10413726.

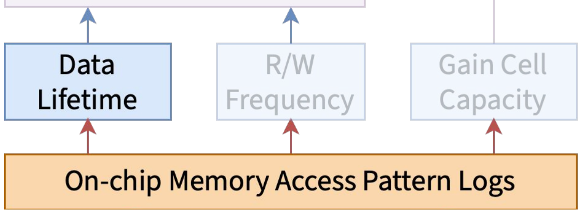
On-Chip Memory Statistics





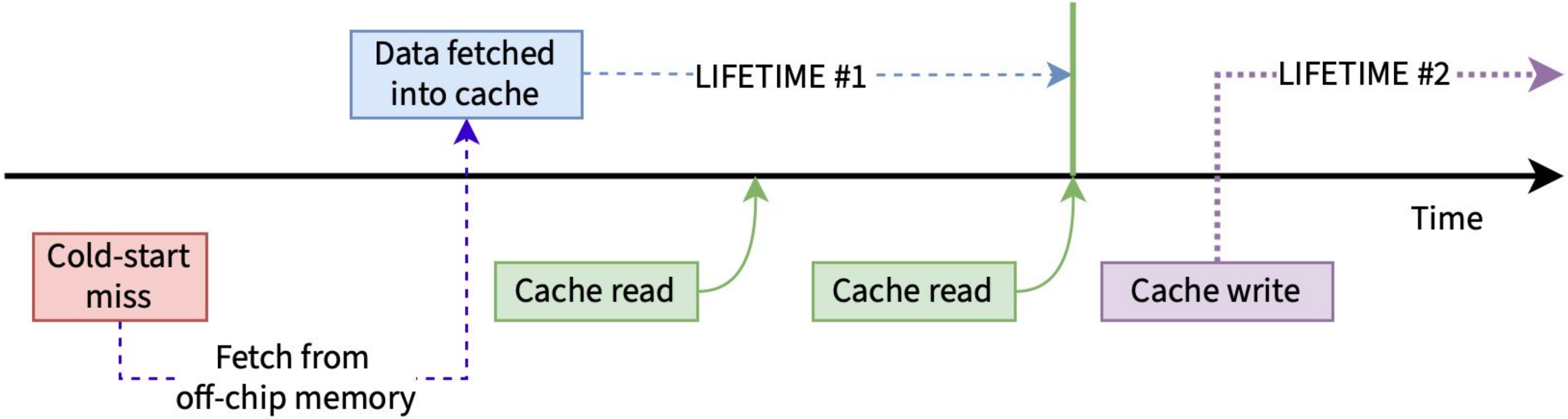
```
GPGPU-Sim Cycle 5560: Load instr from L1D cache at SM 0 bank 1 addr 93ce6d20 status 2
GPGPU-Sim Cycle 5561: Load instr from L1D cache at SM 0 bank 1 addr 93ce6ca0 status 2
GPGPU-Sim Cycle 5562: Load instr from L1D cache at SM 0 bank 1 addr 93ce6c20 status 2
GPGPU-Sim Cycle 5563: Load instr from L1D cache at SM 0 bank 1 addr 93ce6ba0 status 2
GPGPU-Sim Cycle 5564: MEMORY_SUBPARTITION_UNIT -  0 - Load Request to L2 Address=761393cf8f80, status=2
GPGPU-Sim Cycle 5564: MEMORY_SUBPARTITION_UNIT -  2 - Load Request to L2 Address=761393ceef80, status=2
GPGPU-Sim Cycle 5564: MEMORY_SUBPARTITION_UNIT -  5 - Load Request to L2 Address=761393cfaf80, status=2
GPGPU-Sim Cycle 5564: MEMORY_SUBPARTITION_UNIT -  7 - Load Request to L2 Address=761393cecf80, status=2
```

- On chip memory access pattern logs
  - Every read/write operation of the on-chip memory or cache
  - Record timestamps and cache miss or hit state
- Able to trivially compute some statistics
  - Read/write frequency
  - Number of unique memory blocks/cache lines accessed
- Calculating data lifetimes is more nuanced…

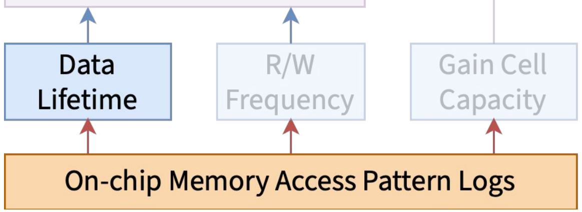


Defining Data Lifetimes

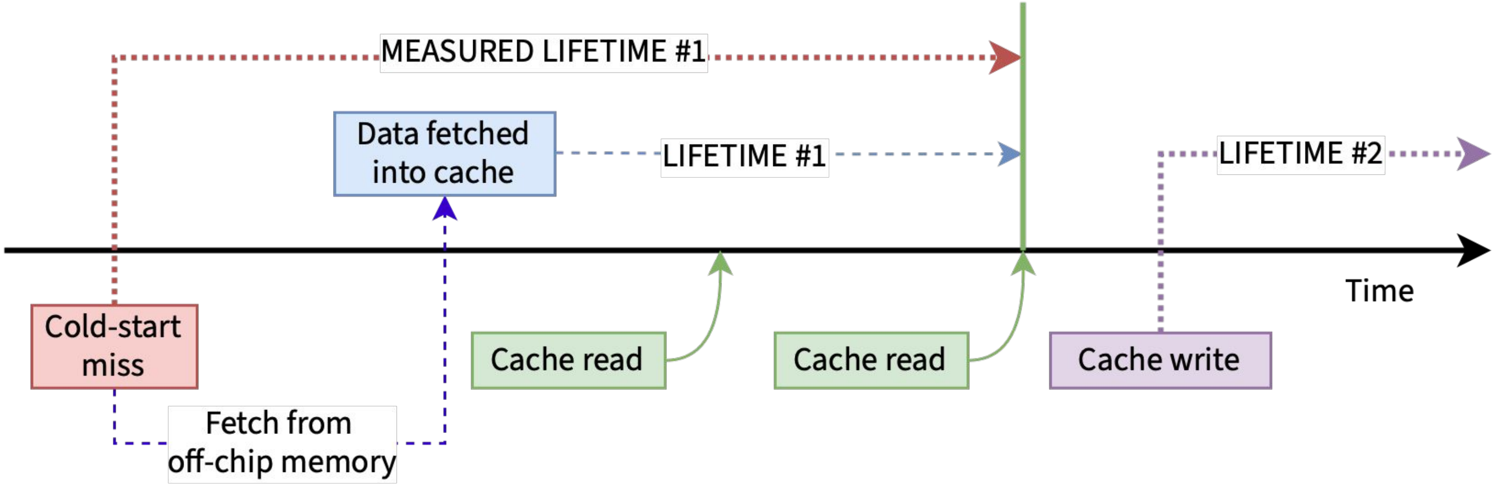
1. Consider accesses to a single address of the on-chip memory
   a. From a cold-start cache miss, to multiple cache reads, to an eventual write operation that overwrites the initial data
   b. Consider lifetimes of *operands to instructions*
2. The ideal way to define lifetimes are "from when data is first written into the memory, to when the data is last consumed"



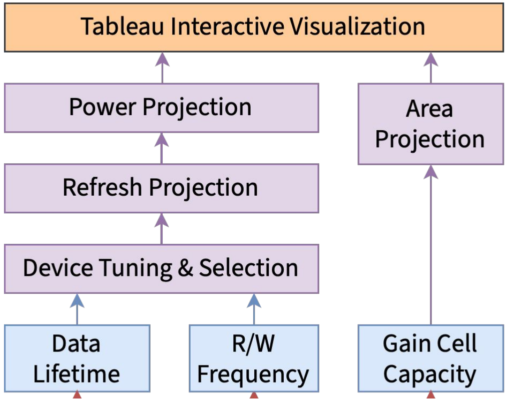
Measuring Data Lifetimes



1. The time between a cold-start miss and when the fetched data is actually written into memory is harder to measure
2. The actual values we are measuring:
   a. Start of lifetime: cache **fetch** operation, or **write** operation
   b. End of lifetime: **last read** before a write or before end of program



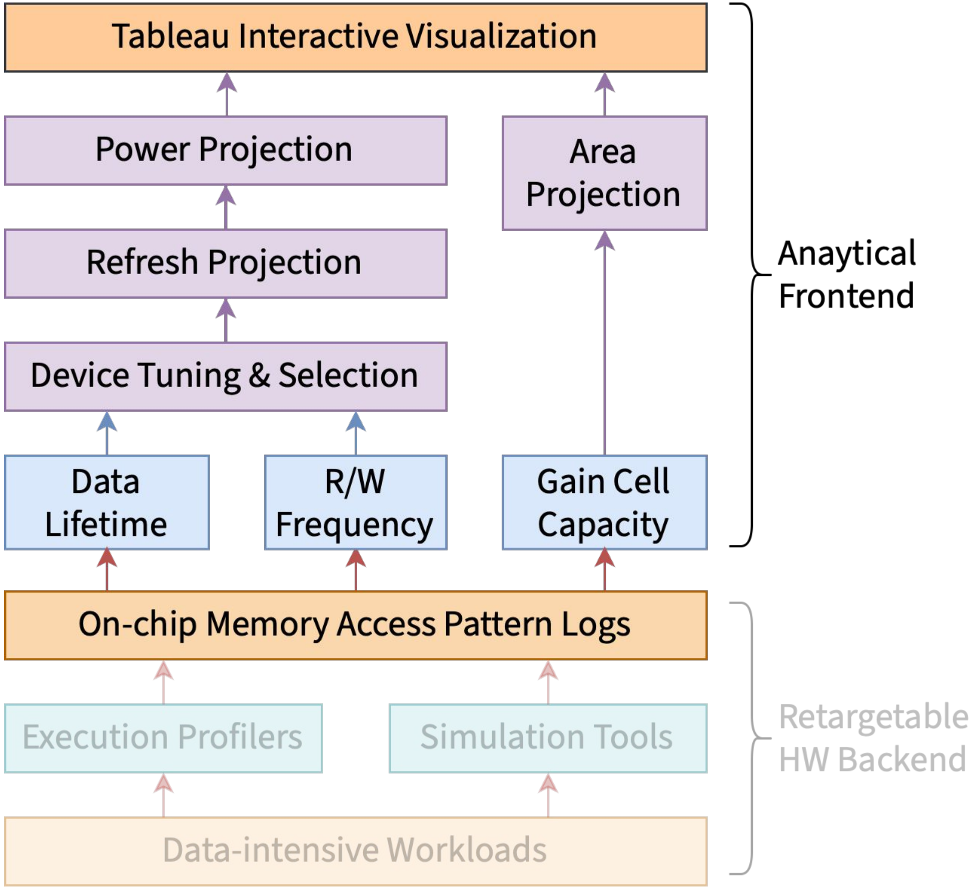
Performance Projection



- Correlating measured stats with memory device
  - 5nm (N5) process
  - SRAM vs. silicon gain cell vs. hybrid gain cell
  - Determine retention time of cell using measured write frequencies
- Project number of **refreshes** for each on-chip memory block or cache line
  - Divide measured data lifetime by tuned retention times of each cell
- Project **bit cell area** using unique number of cache lines accessed
  - Multiply gain cell capacity requirement with area per cell for each device
  - Assume 60% area efficiency when considering peripheral circuits
- Project **total energy** using counts of reads, writes, refreshes
  - A refresh is achieved by a read followed by a write
  - Multiply total R/W counts by energy per bit for each device

Interactive Visualization

- Produce following plots in Tableau
  - The **distribution** of data lifetimes for each workload
  - Comparison of **area versus power** for each device for each workload
- Enable user to filter by workload/kernel/device
- Explore the impact of gain cells for ranges of workloads



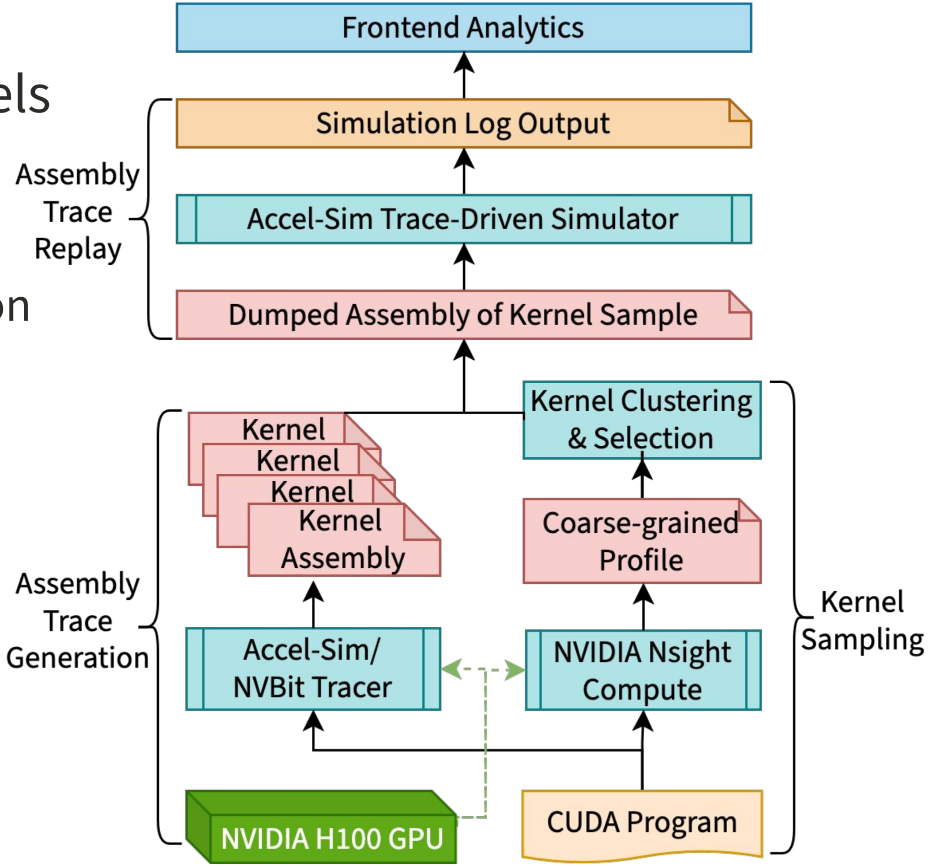
Case Studies and Experiment Results

Premise and Methodology

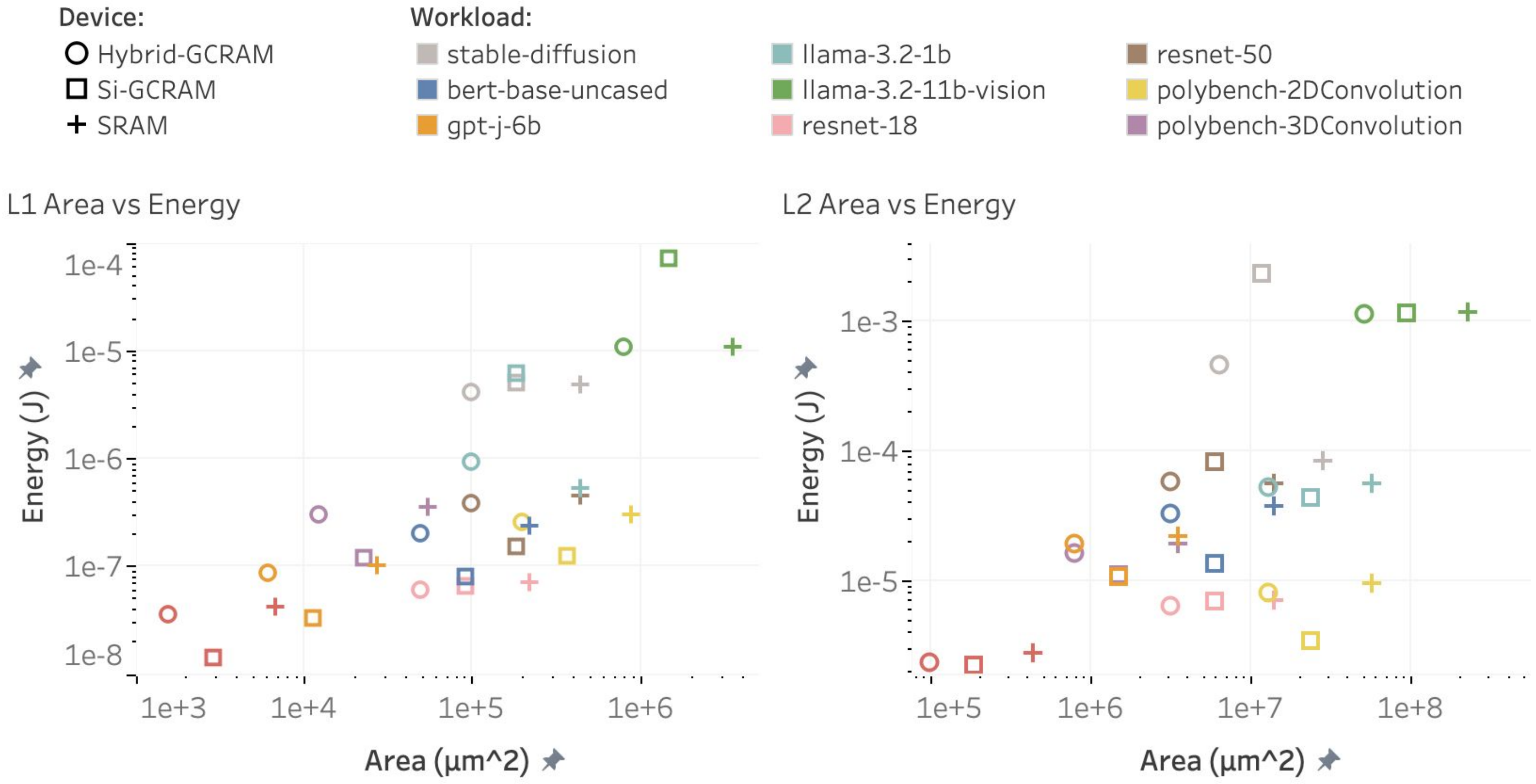
1. ***What if we replace on-chip SRAM memories on different AI Accelerators with gain cells?***
    a. Measure **lifetimes** and estimate number of **refreshes** needed
    b. Compare **area & power** figures with those achievable using SRAM
2. Experiment methods
    a. Use different runs of inference tasks from **MLPerf Inference** and PolyBench
    b. Run tasks for **both GPU and systolic array backends**
    c. Analyze cache access logs to obtain **lifetime, R/W frequency, etc. statistics**
    d. Use profiled statistic to **estimate area, refresh, and power requirement** for hypothetical gain cell replacements for on-chip memory structures

Case Study 1: GPU Simulation

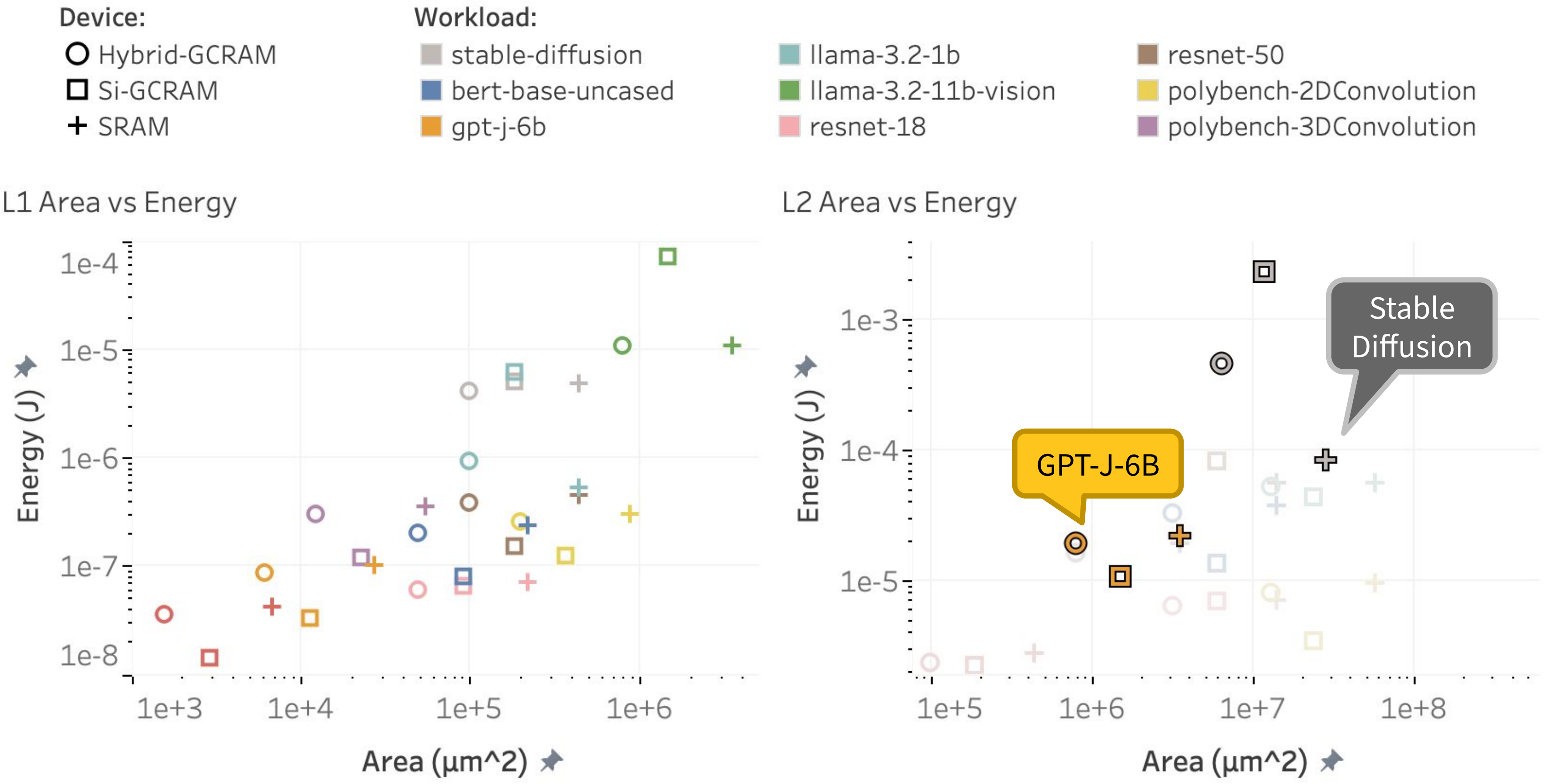
1. Extract **SASS assembly** with NVBit
2. Perform **sampling analysis** on kernels
   a. Measure per-kernel, coarse-grained memory behavior in Nsight Compute
   b. K-means clustering of kernels based on memory metrics
   c. Select subset of SASS traces from cluster centroids
3. **Replay SASS traces** in Accel-Sim
4. Ablation study: GPU cache pollution and write allocation policy



Case Study 1 Results



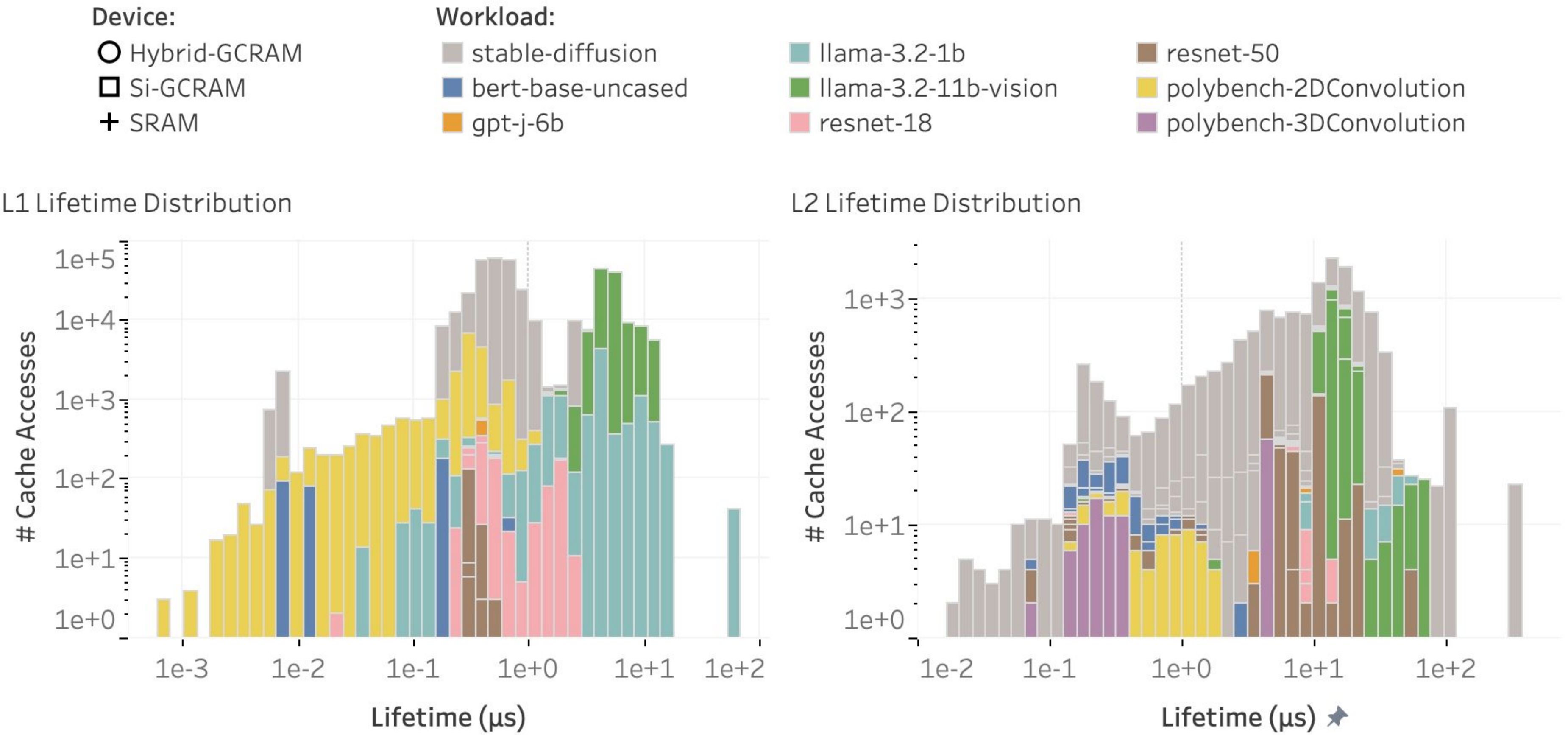
Case Study 1 Results



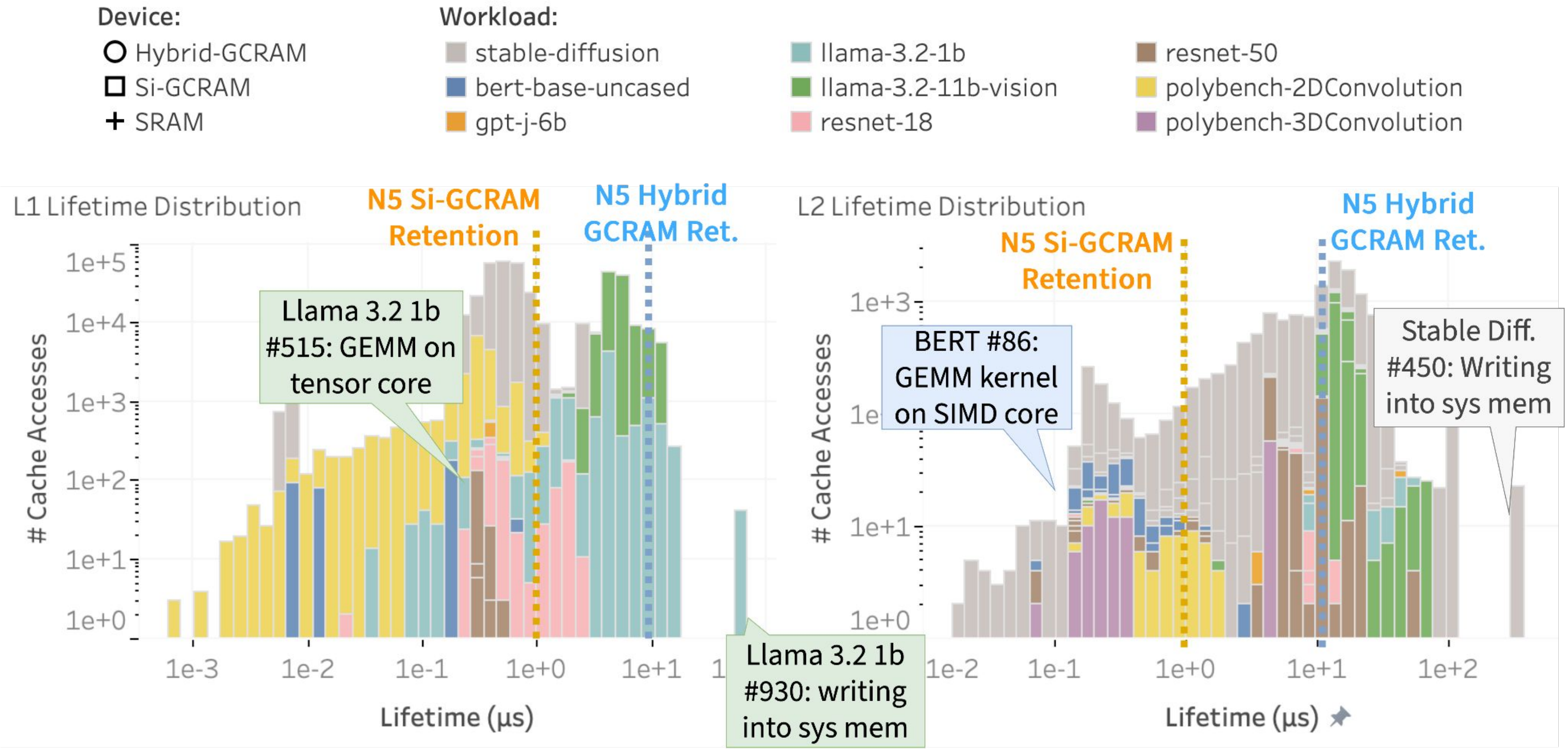
GPT-J-6B

Stable Diffusion

Case Study 1 Results



Case Study 1 Results



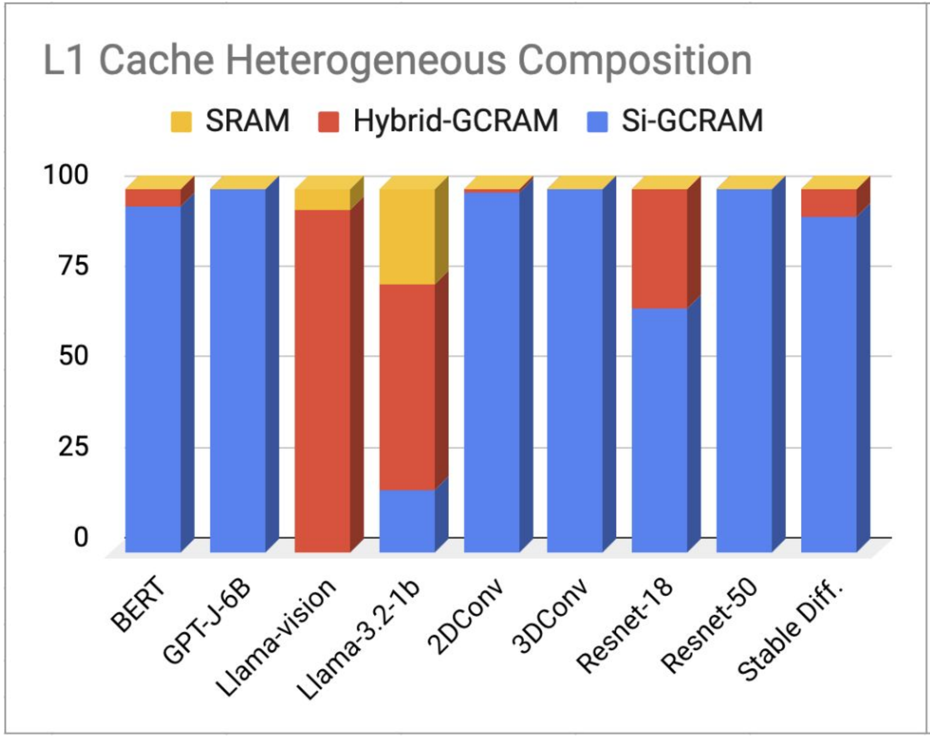
Llama 3.2 1b #515: GEMM on tensor core

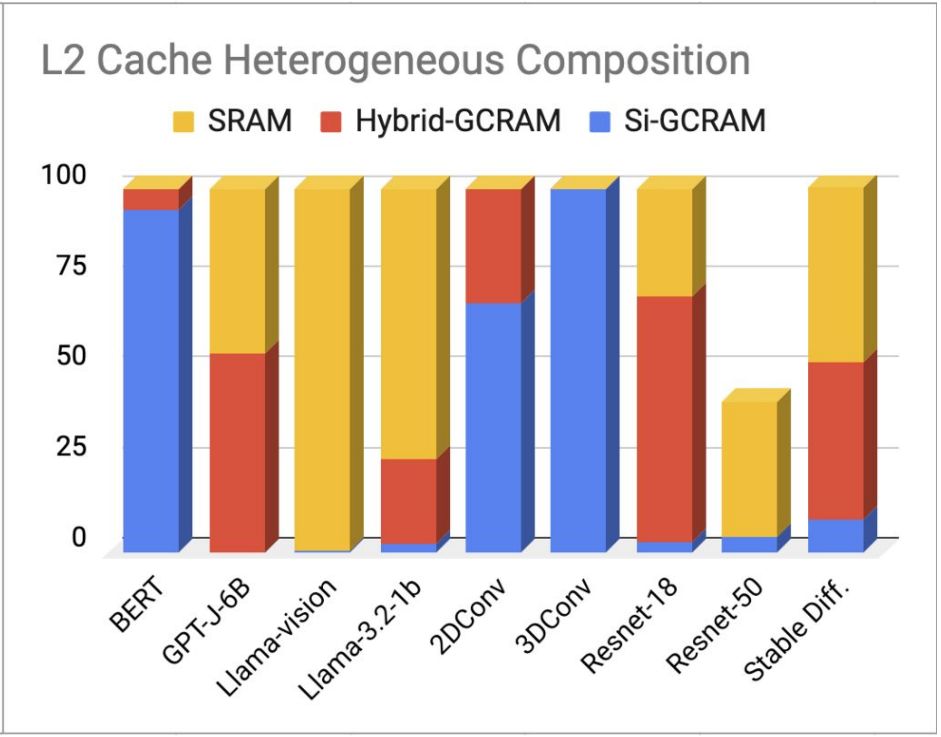
Llama 3.2 1b #930: writing into sys mem

BERT #86: GEMM kernel on SIMD core

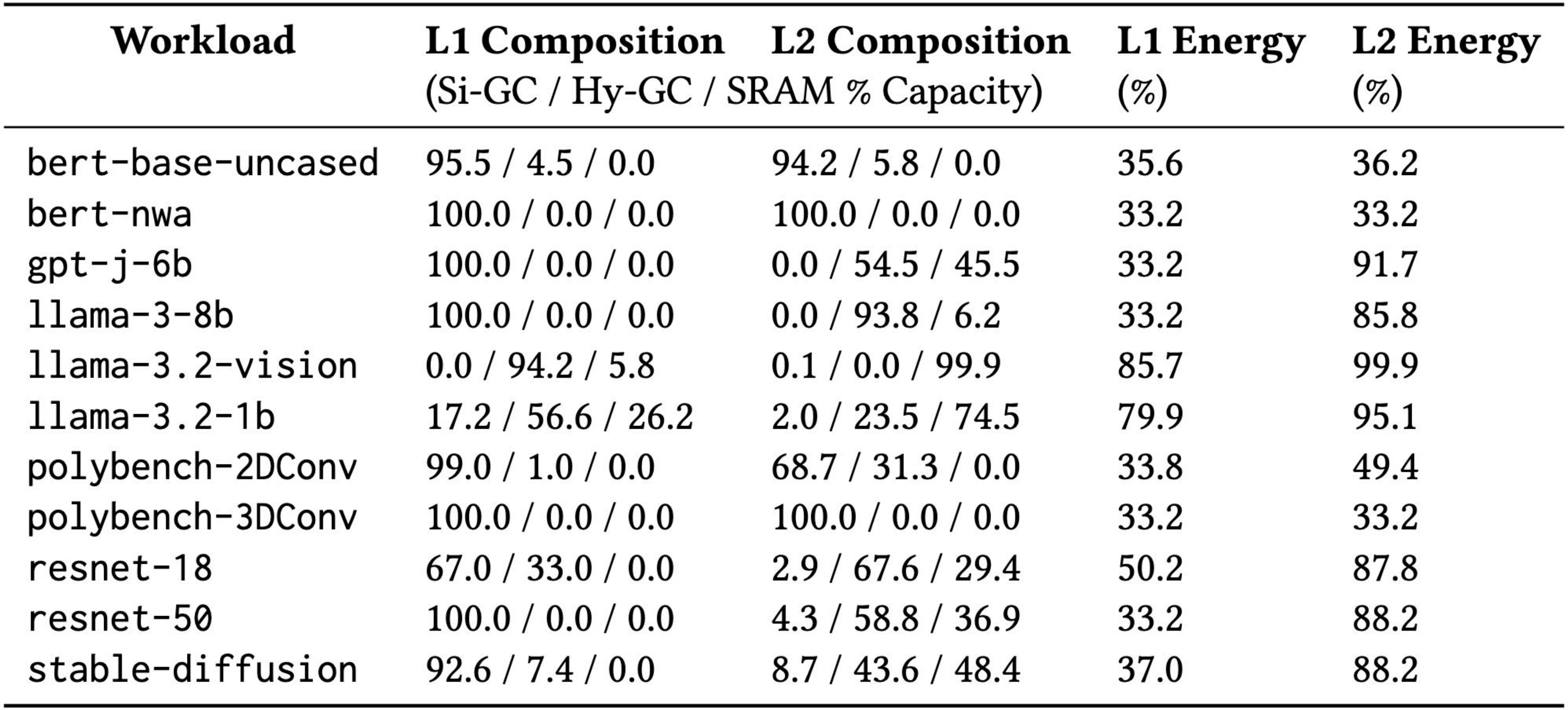
Stable Diff. #450: Writing into sys mem

Case Study 1: Heterogeneous Mem Composition





Case Study 1: Heterogeneous Mem Composition



Workload	L1 Composition	L2 Composition	L1 Energy	L2 Energy
	(Si-GC / Hy-GC / SRAM % Capacity)		(%)	(%)
bert-base-uncased	95.5 / 4.5 / 0.0	94.2 / 5.8 / 0.0	35.6	36.2
bert-nwa	100.0 / 0.0 / 0.0	100.0 / 0.0 / 0.0	33.2	33.2
gpt-j-6b	100.0 / 0.0 / 0.0	0.0 / 54.5 / 45.5	33.2	91.7
llama-3-8b	100.0 / 0.0 / 0.0	0.0 / 93.8 / 6.2	33.2	85.8
llama-3.2-vision	0.0 / 94.2 / 5.8	0.1 / 0.0 / 99.9	85.7	99.9
llama-3.2-1b	17.2 / 56.6 / 26.2	2.0 / 23.5 / 74.5	79.9	95.1
polybench-2DConv	99.0 / 1.0 / 0.0	68.7 / 31.3 / 0.0	33.8	49.4
polybench-3DConv	100.0 / 0.0 / 0.0	100.0 / 0.0 / 0.0	33.2	33.2
resnet-18	67.0 / 33.0 / 0.0	2.9 / 67.6 / 29.4	50.2	87.8
resnet-50	100.0 / 0.0 / 0.0	4.3 / 58.8 / 36.9	33.2	88.2
stable-diffusion	92.6 / 7.4 / 0.0	8.7 / 43.6 / 48.4	37.0	88.2

Case Study 1 Ablation Results

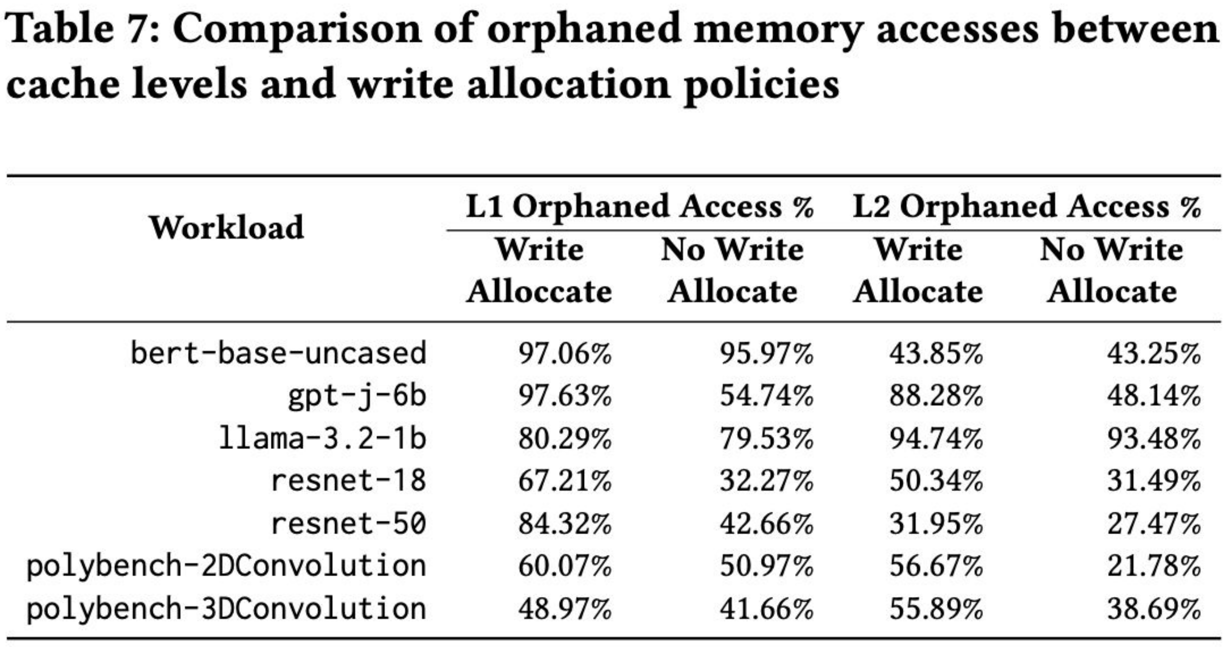
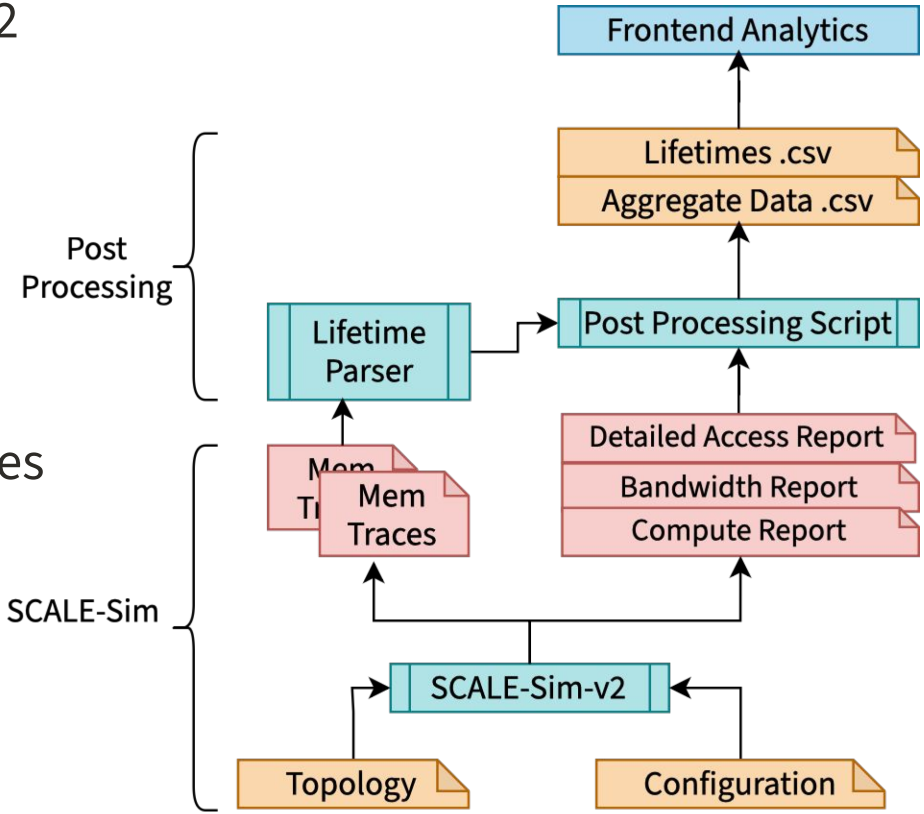


Table 7: Comparison of orphaned memory accesses between cache levels and write allocation policies

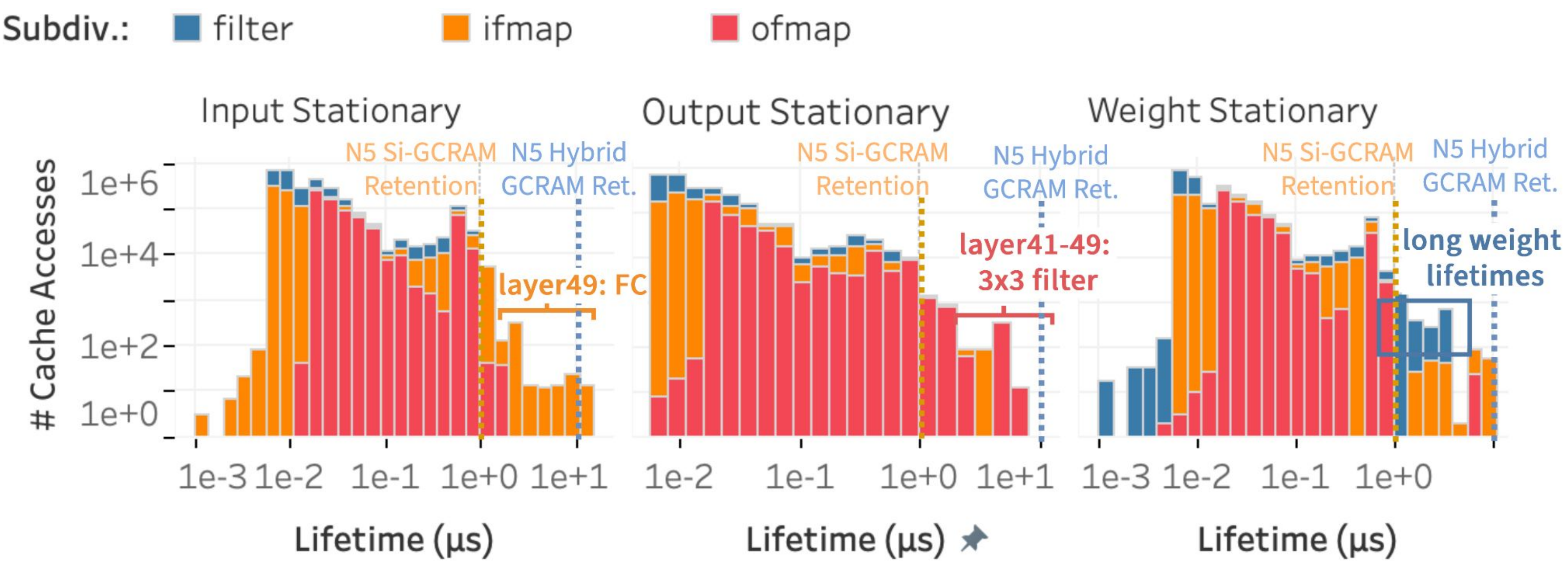
Workload	L1 Orphaned Access %		L2 Orphaned Access %	
	Write Alloccate	No Write Allocate	Write Allocate	No Write Allocate
bert-base-uncased	97.06%	95.97%	43.85%	43.25%
gpt-j-6b	97.63%	54.74%	88.28%	48.14%
llama-3.2-1b	80.29%	79.53%	94.74%	93.48%
resnet-18	67.21%	32.27%	50.34%	31.49%
resnet-50	84.32%	42.66%	31.95%	27.47%
polybench-2DConvolution	60.07%	50.97%	56.67%	21.78%
polybench-3DConvolution	48.97%	41.66%	55.89%	38.69%

Case Study 2: Systolic Array Simulation

1. 256*256 systolic array in SCALE-Sim-v2
2. Run the convolution operations of ResNet-50 through systolic array
3. Dataflow variations: input stationary, weight stationary, output stationary
4. Coarse-grained compute/bandwidth reports and fine-grained memory traces
5. Outputs same cache access logs to be processed by frontend

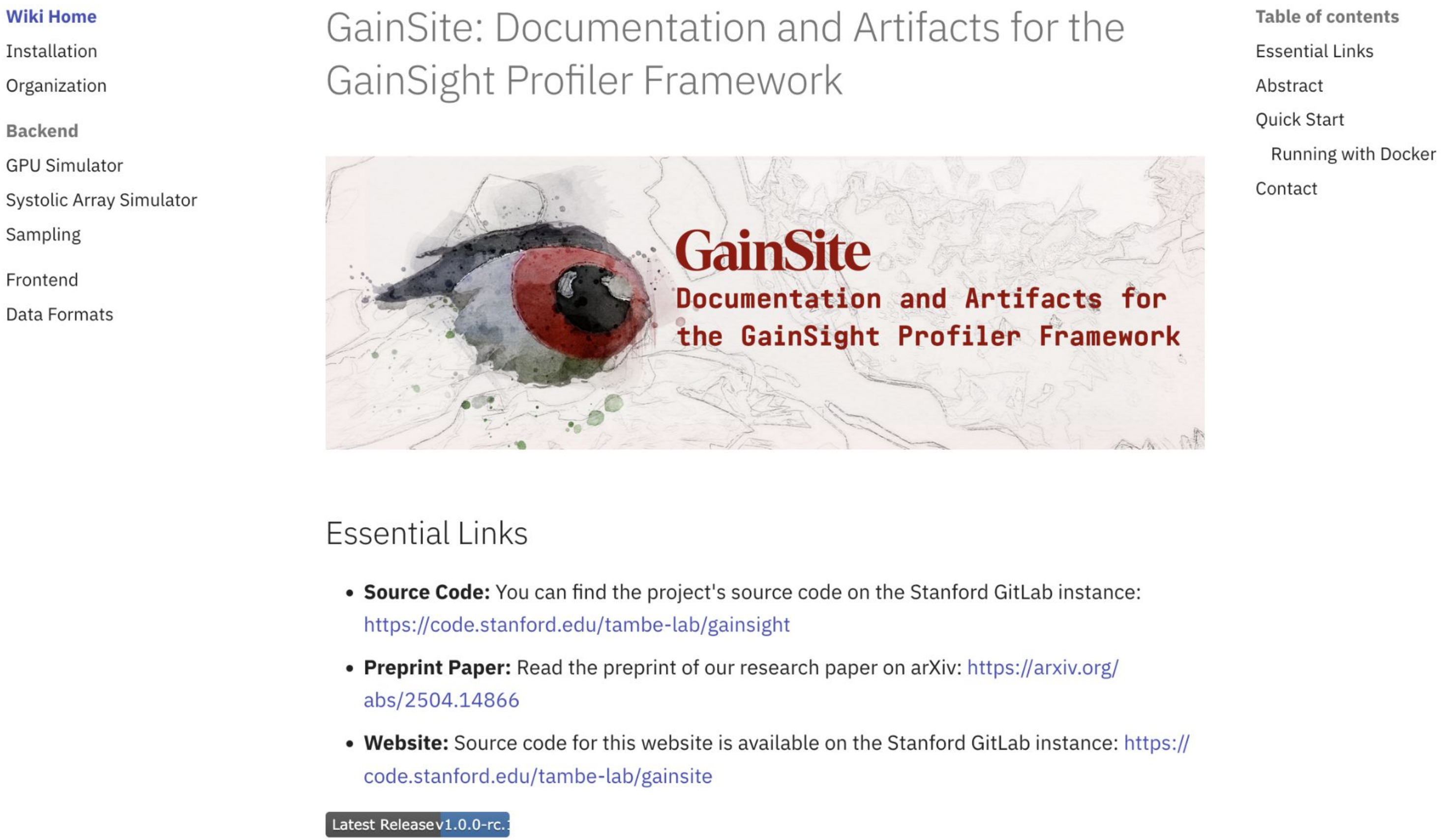


Case Study 2: Results



Discussions

GainSight is Open-Source at gainsight.stanford.edu



Wiki Home

Installation

Organization

Backend

GPU Simulator

Systolic Array Simulator

Sampling

Frontend

Data Formats

GainSite: Documentation and Artifacts for the GainSight Profiler Framework

GainSite
Documentation and Artifacts for the GainSight Profiler Framework

Essential Links

- **Source Code:** You can find the project's source code on the Stanford GitLab instance: https://code.stanford.edu/tambe-lab/gainsight
- **Preprint Paper:** Read the preprint of our research paper on arXiv: https://arxiv.org/abs/2504.14866
- **Website:** Source code for this website is available on the Stanford GitLab instance: https://code.stanford.edu/tambe-lab/gainsite

Latest Release v1.0.0-rc.

Table of contents

Discussions

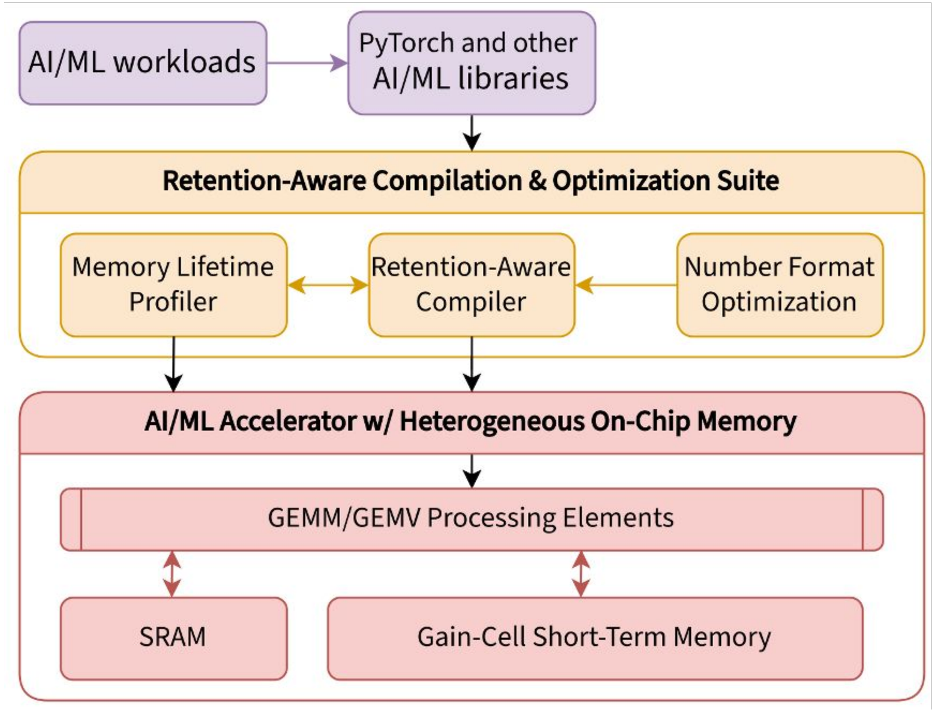
1. Heterogeneous data access patterns lead to need for heterogeneous memory and compute hardware
   a. Substantial portions of data suitable for short-term memory
2. Cache pollution and utilization in general remain ongoing issues
3. Dataflow choices in systolic arrays affect memory requirements
   a. Analytical dataflow evaluation in complex models still might be helpful
4. Circuit-level techniques to enhance retention times
   a. Tuning threshold voltage, increase clock frequency
5. Further work: use GainSight to evaluate other forms of memory devices other than GCRAM

Summary

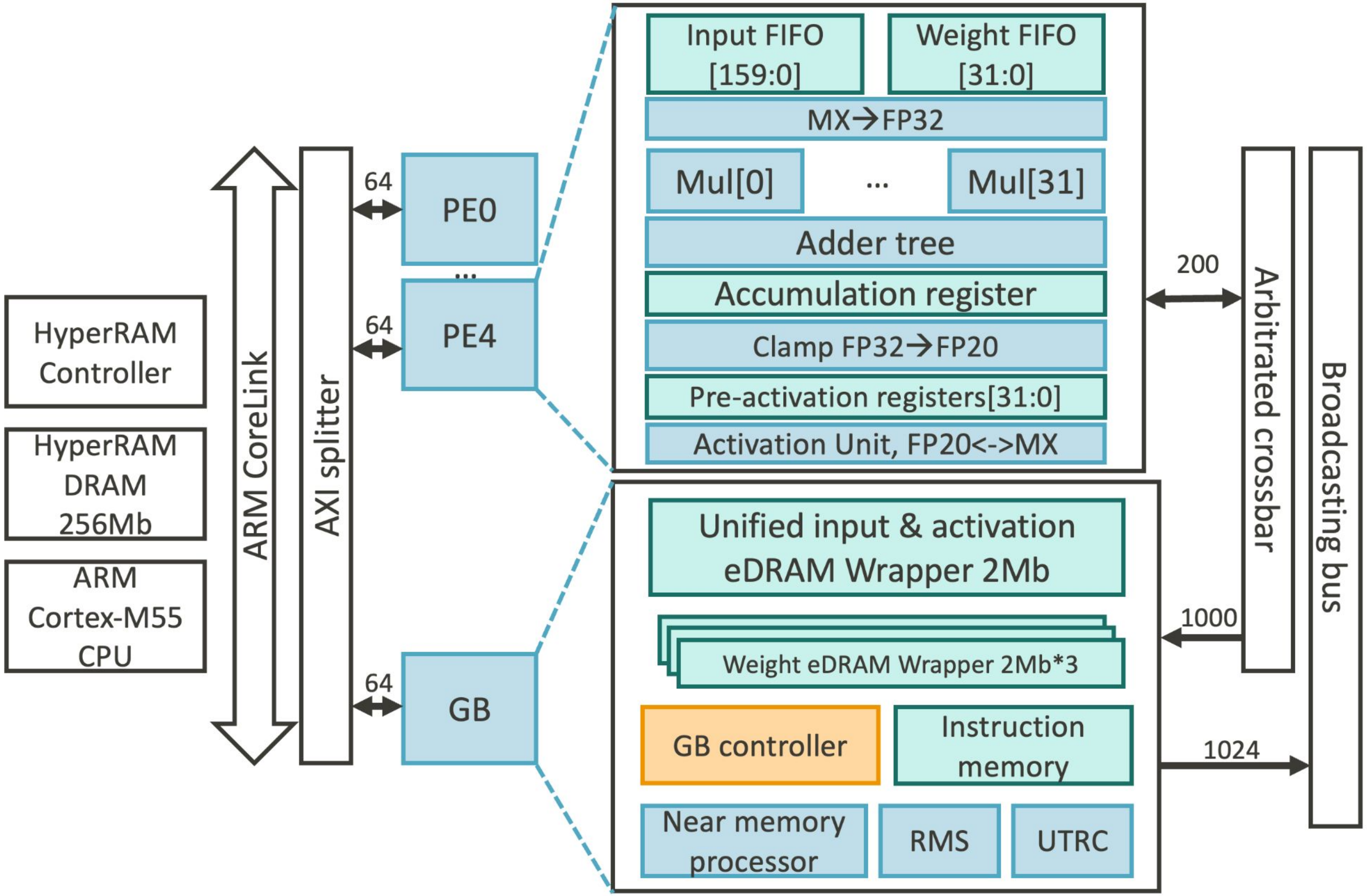
- SRAM scaling is ending.
- Huge opportunity to tame the memory wall by aligning application memory access patterns and lifetime metrics to differentiated memory systems.
- https://gainsight.stanford.edu is live and report fine-grained on-chip lifetime metrics along with PPA benefit projections from matching differentiated memories (Si-GC, Hybrid-GC, NVM, SRAM).
  - Current HW support includes NVIDIA GPUs, systolic arrays, and a guide for bring-your-own-HW. More HW support coming online soon!
  - Initial release to show results on MLPerf Inference and PolyBench workloads.

Next Steps: GCRAM Integration Efforts

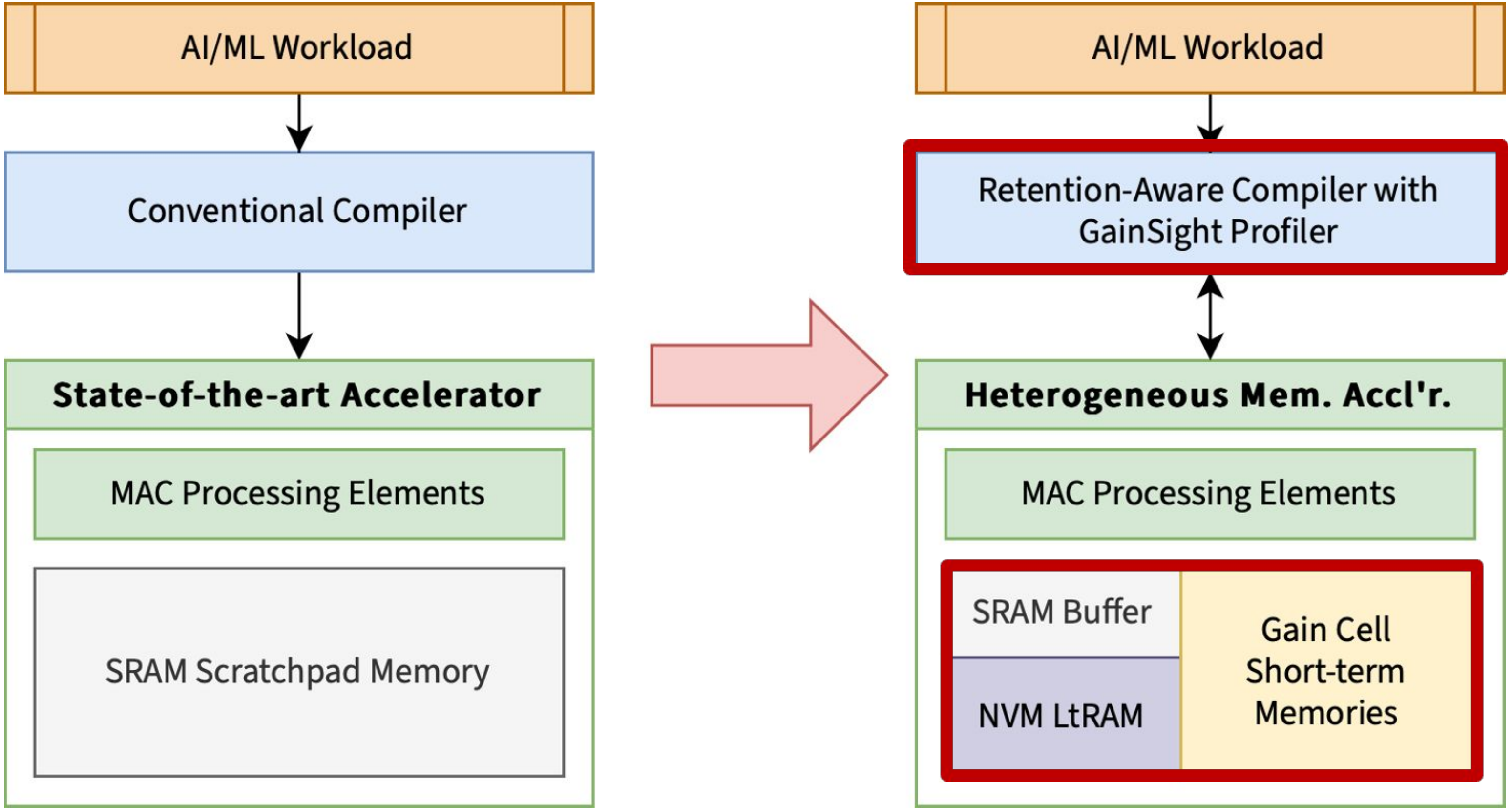
- We are building a **16nm gain-cell based AI accelerator chip**
  - SIMD-based GEMM/GEMV PEs
  - Gain-cell RAM as primary on-chip, short term memory
- First in a series of SW tools for retention-aware compiler and optimizer
- Target model: **MAMBA state-space language model**


AI/ML workloads
PyTorch and other AI/ML libraries
Retention-Aware Compilation & Optimization Suite
Memory Lifetime Profiler
Retention-Aware Compiler
Number Format Optimization
AI/ML Accelerator w/ Heterogeneous On-Chip Memory
GEMM/GEMV Processing Elements
SRAM
Gain-Cell Short-Term Memory

Rounding Out Our Vision



Rounding Out Our Vision



Acknowledgement

1. Grad student co-authors:
   Peijing Li, Matthew Hung, Yiming Tan, Konstantin Hoßfeld, Jake Cheng, Shuhan Liu, Lixian Yan, Xinxin Wang
2. Faculty advisors: Thierry Tambe, Philip Wong, Philip Levis, Subhasish Mitra
3. Sponsors and supporters

EMD Electronics Google

NSF DARPA MemoryDAX

Semiconductor Research Corporation Northwest AI HUB

Conclusion

1. GainSight provides insights into memory access patterns and data lifetimes to help with gain cell memory array design
   a. Application-guided profiling enables better heterogeneous memory design
2. Significant portion of on-chip memory accesses are short-lived:
   a. 40% of L1 and 18% of L2 GPU cache accesses
   b. 79% of systolic array scratchpad accesses
3. Si-GCRAM can reduce active energy by 11-28% compared to SRAM
4. Open-source availability
   a. Preprint paper: https://arxiv.org/abs/2504.14866
   b. Source code: https://code.stanford.edu/tambe-lab/gainsight.git
   c. Documentation and visualization site (“GainSite”): https://gainsight.stanford.edu

Thank You!