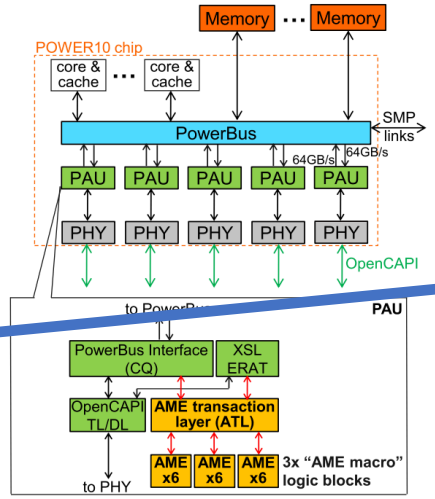
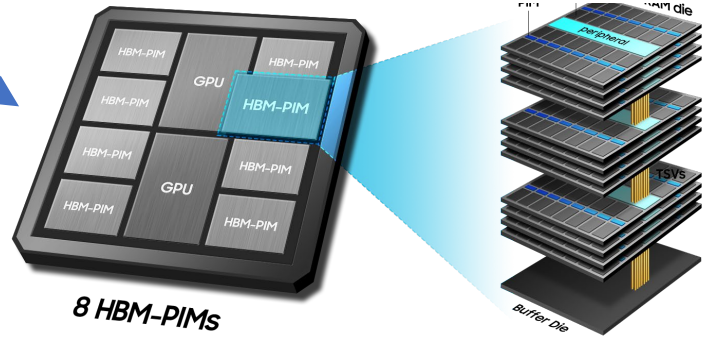
A Workflow for the Synthesis of Irregular Memory Access Microbenchmarks

Kevin Sheridan, Jered Dominguez-Trujillo, Galen Shipman (LANL)
Connor Radelja, Christopher Scott, Agustin Vaca Valverde, Jeffrey Young (GT)
Patrick Lavin (SNL)

October 2nd, 2024

Motivation





The "Cambrian Explosion" of new accelerators has led to many domain specific accelerators – not just in AI but also for data movement!

Motivation

	Memory subsystem						Floating Point		
	L1	L2	L3	DRAM	DRAM BW	Mem Latency	DP FLOPs	Vectorization	Non-FP
Flag 3D Ale							2.50%	7.10%	97.50%
PartiSN 42 groups							26.20%	90.40%	73.80%
Jayenne DDMC Hohlraum							14.30%	0.20%	85.70%
xRAGE Shaped Charge							6.50%	14.00%	93.50%
Application 1							7.80%	19.20%	92.20%
Application 2							8.10%	17.60%	91.90%

Hardware Bottlenecks for LANL HPC codes [1]

However, we still have little understanding of how future memory accelerators might affect codes of interest because:

- Finding appropriate regions of interest (ROI) for the memory system is challenging to infer even with tools like SimPoints and LoopPoint
- Some applications that we might want to benchmark can't be fully shared to extract meaningful traces or ROIs

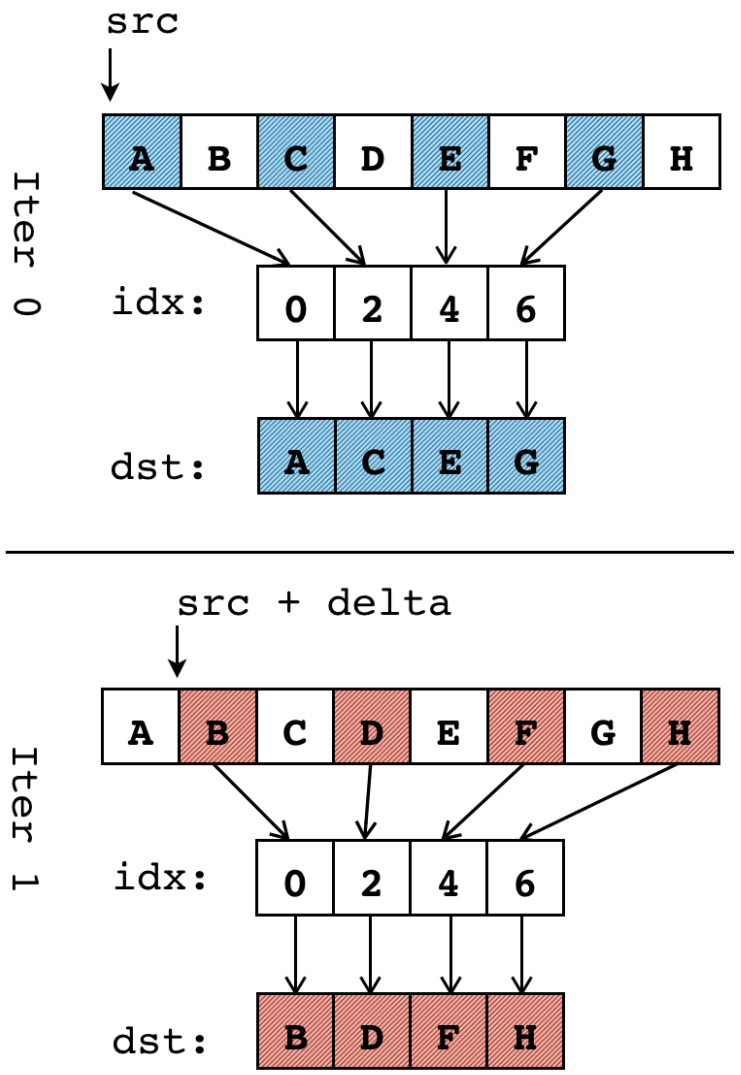
Our ideal workflow would allow us to 1) capture relevant memory accesses from full applications, 2) benchmark them on real systems, and 3) simulate new hardware models

[1] G. Shipman, et al., Assessing the Memory Wall in Complex Codes, MCHPC 2022 doi: 10.1109/MCHPC56545.2022.00009

Spatter - Version 1.0

The basis of Spatter is two kernels; one for gather and another for scatter.

```
Gather kernel:
for i in 0..N:
    for j in len(idx):
        dst[j] = src[delta*i + idx[j]]
Scatter kernel:
for i in 0..N:
    for j in len(idx):
        dst[delta*i + idx[j]] = src[j]
```



The delta and the pattern in idx specify the memory *access pattern*.

The Need for an Open Source Workflow

1. Trace G/S Instructions - pulled from a proprietary simulator

```
Gather  0x0040, [0, 2, 4, 6]
Gather  0x0044, [0, 2, 4, 6]
Gather  0x0048, [0, 2, 4, 6]
Scatter 0x004C, [1, 1, 5, 5]
```

2. Change base address to delta

```
Gather  _, [0, 2, 4, 6]
Gather  4, [0, 2, 4, 6]
Gather  4, [0, 2, 4, 6]
Scatter 4, [1, 1, 5, 5]
```

3. Aggregate Counts

```
Gather  4, [0, 2, 4, 6], 2
Scatter 4, [1, 1, 5, 5], 1
```

Spatter's Format

Previous work focused just on the offsets found within individual instructions. We want to capture application level gather and scatter behavior.

GS Patterns

Sliding window approach is used to track non-trivial memory accesses within an application trace or region of interest (ROI)

$$\textbf{ScatterWindow} = \begin{array}{c} \begin{array}{cccc} \text{maddr1} & \text{maddr2} & \ldots & \text{maddrN} \end{array} \\ \begin{bmatrix} 0x0480 & 0x0488 & \ldots & 0x1488 \\ 0x1780 & 0x1788 & \ldots & 0x0783 \\ \ldots & \ldots & \ldots & \ldots \\ 0x9580 & 0x5588 & \ldots & 0x3581 \end{bmatrix} \end{array} \begin{array}{c} iaddr \\ 0x0001 \\ 0x0003 \\ \ldots \\ 0x1019 \end{array}$$

Multiple filters are used to keep the most relevant access patterns for the final pattern output

Filter #	Description
1	Index distances are only -1, 0, and/or 1
2	No symbol
3	Not in top 10 window appearance counts
4	Less than 1024 instances
5	Less than 6 unique index distances and less than 50% out of bounds distances*

*Filter rules apply to each full memory access sequence. Sub-sequences are not removed. *Default out of bounds distances in $(-\infty, -513]$ or $[513, \infty)$.*

GS Patterns codebase at https://github.com/lanl/gs_patterns and https://github.com/hpcgarage/gs_patterns

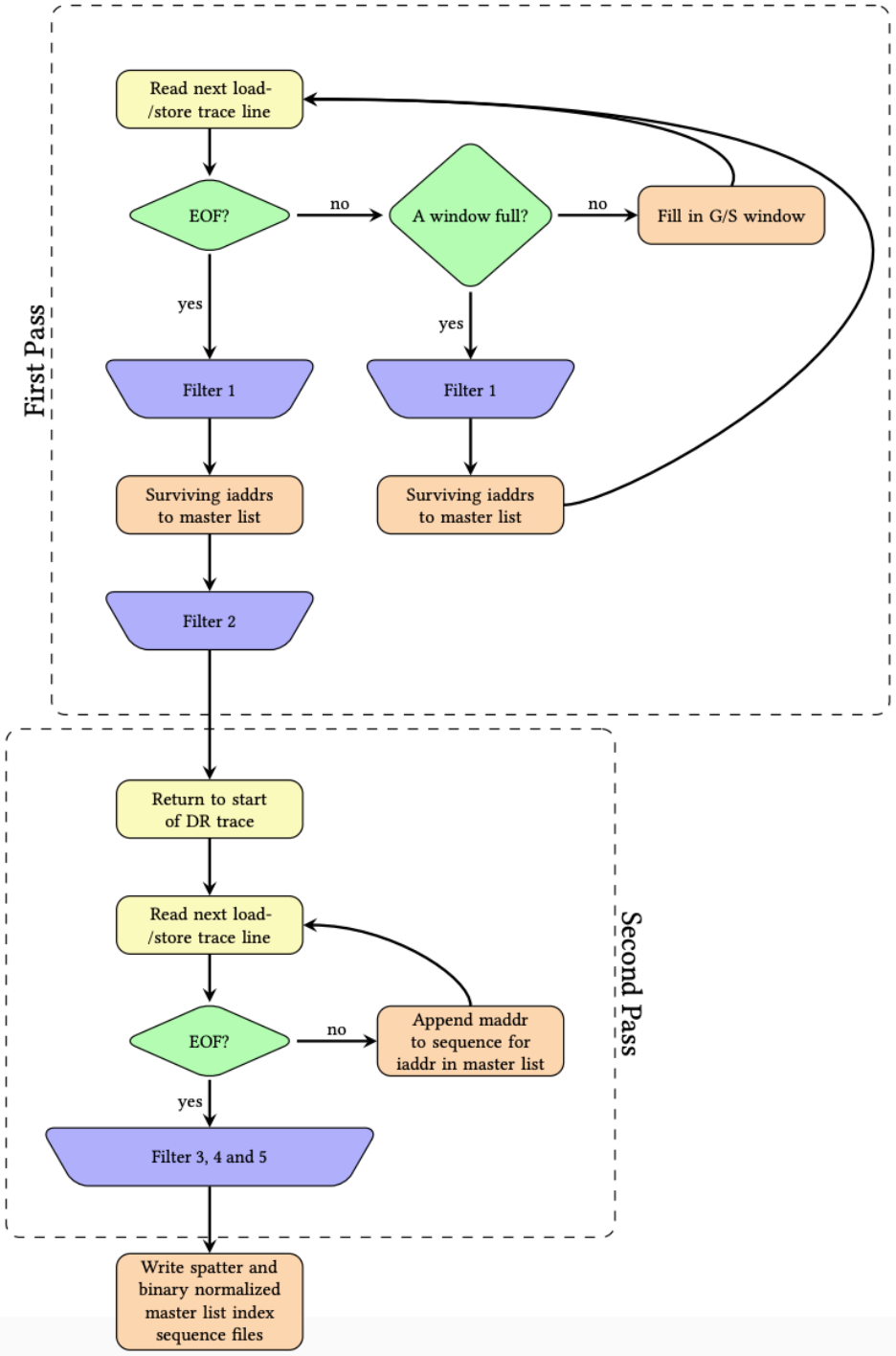
GS Patterns

Two passes are used to apply all filters and output the patterns of interest

GS Patterns output: A pattern for each instruction address that makes it all the way through the filters

Filter #	Description
1	Index distances are only -1, 0, and/or 1
2	No symbol
3	Not in top 10 window appearance counts
4	Less than 1024 instances
5	Less than 6 unique index distances and less than 50% out of bounds distances*

*Filter rules apply to each full memory access sequence. Sub-sequences are not removed. *Default out of bounds distances in $(-\infty, -513]$ or $[513, \infty)$.*

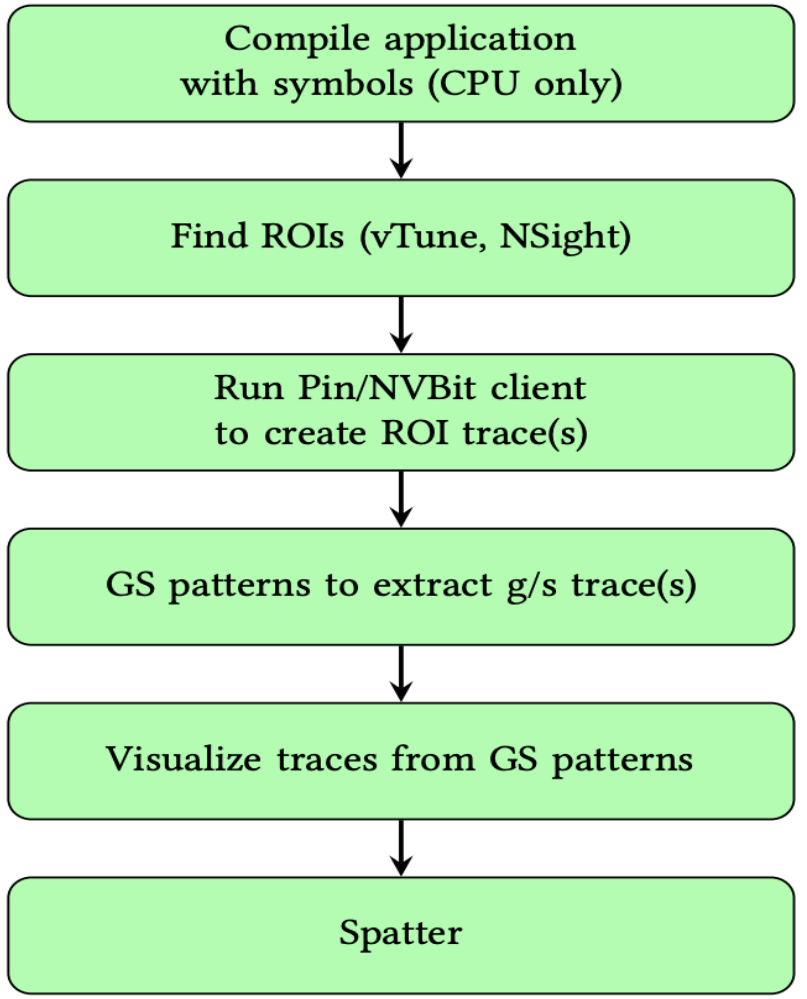


GS Patterns Workflow

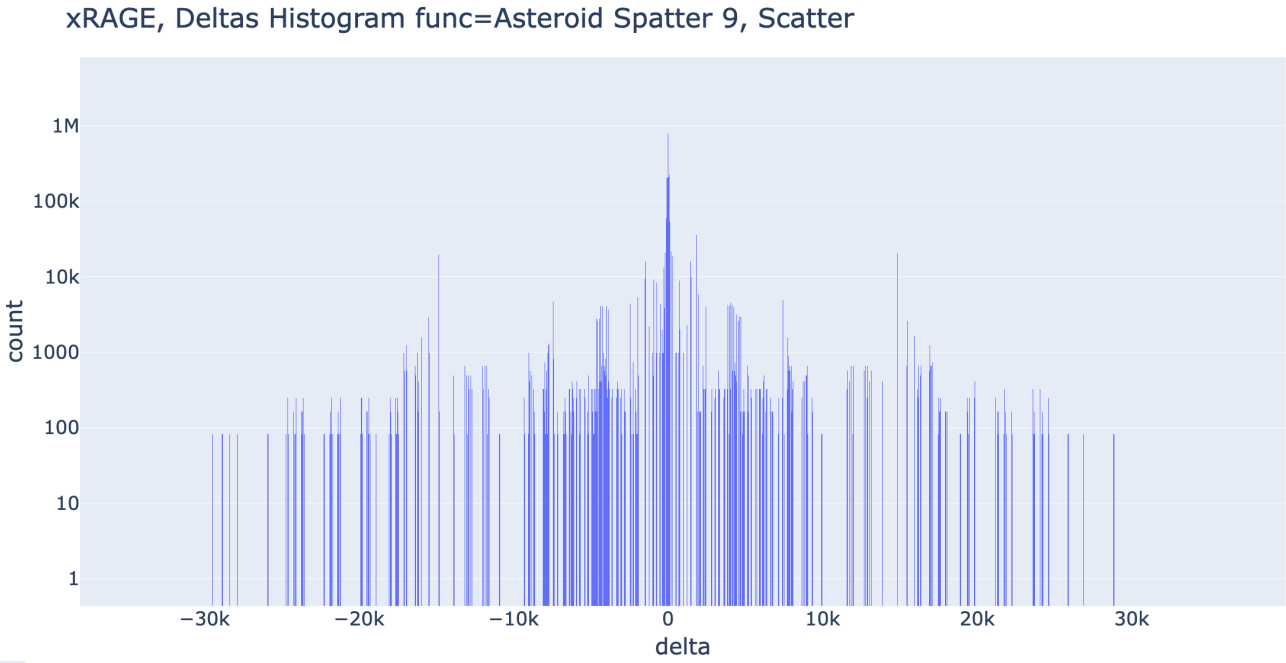
Further extensions to GS Patterns refactored code to use C++ and a plugin infrastructure for PinTool, NVBit

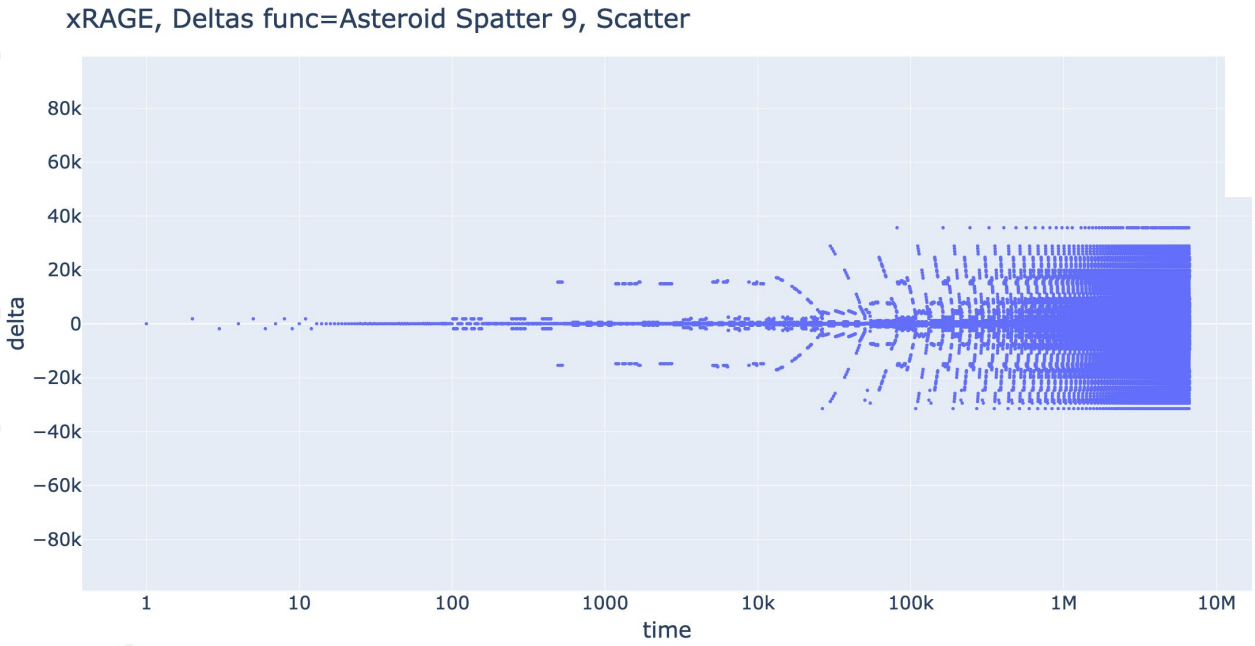
Currently ROI analysis speeds up the overall GS Patterns workflow but is a somewhat manual process to run and annotate codes

Check out collected public patterns at https://github.com/hpcgarage/spatter-patterns



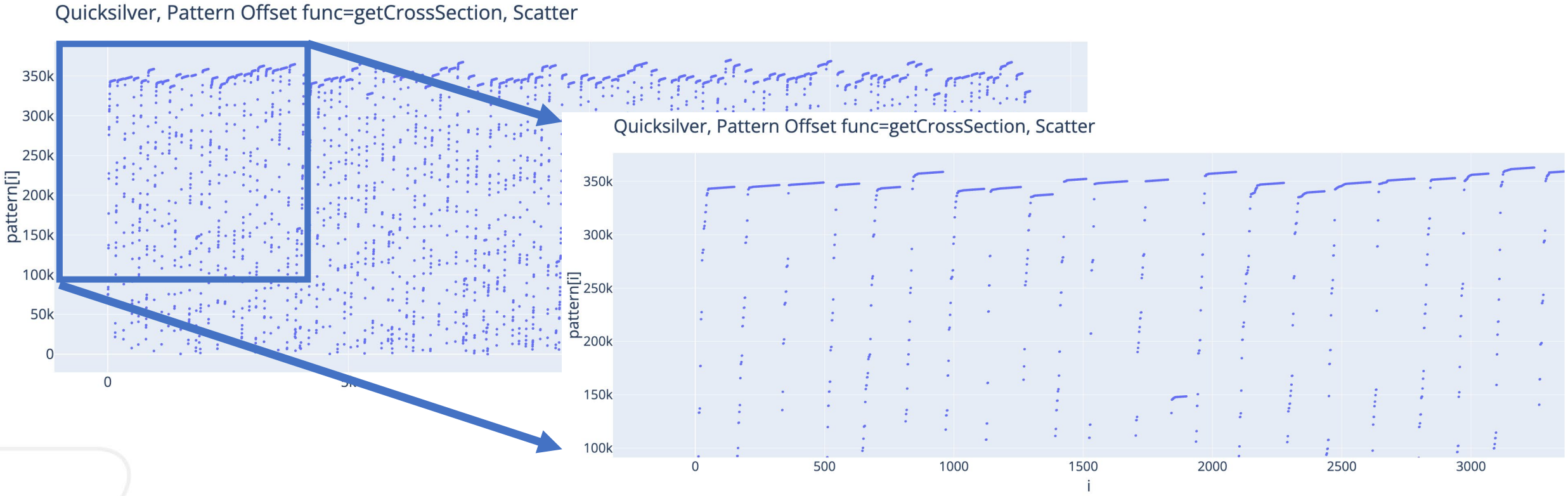
Pattern Visualization Tools





The GS Patterns JSON output can be easily visualized in a variety of different ways…

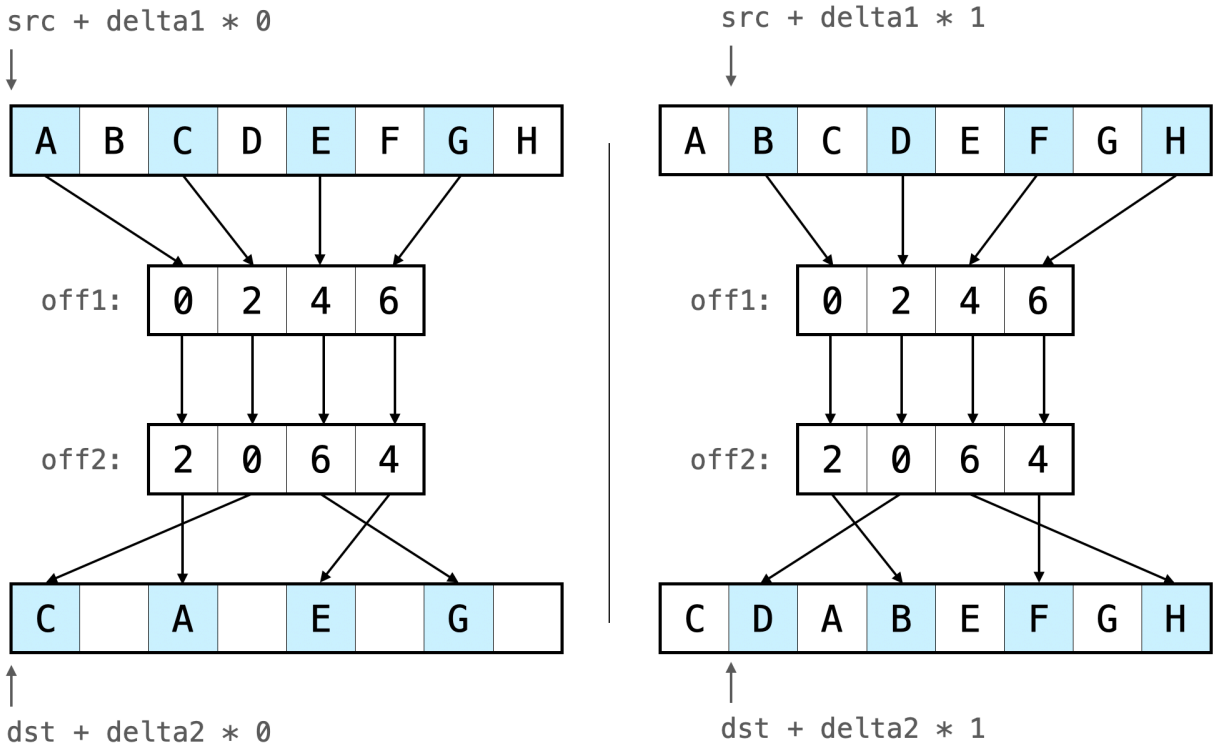
Pattern Visualization Tools



But we likely need to do more detailed statistical analysis to make more sense of these patterns..

Spatter 2.0

- Complete refactor of argument parsing, build system, and movement towards C++ design
- Addition of new kernels to better represent multiple levels of indirection - GatherScatter, MultiGather, MultiScatter
- Support for longer offset buffer lengths for improved application pattern representation
- MPI support for weak/strong scaling
- Support for atomics with scatter operations



GatherScatter Kernel Representation

https://github.com/hpcgarage/spatter

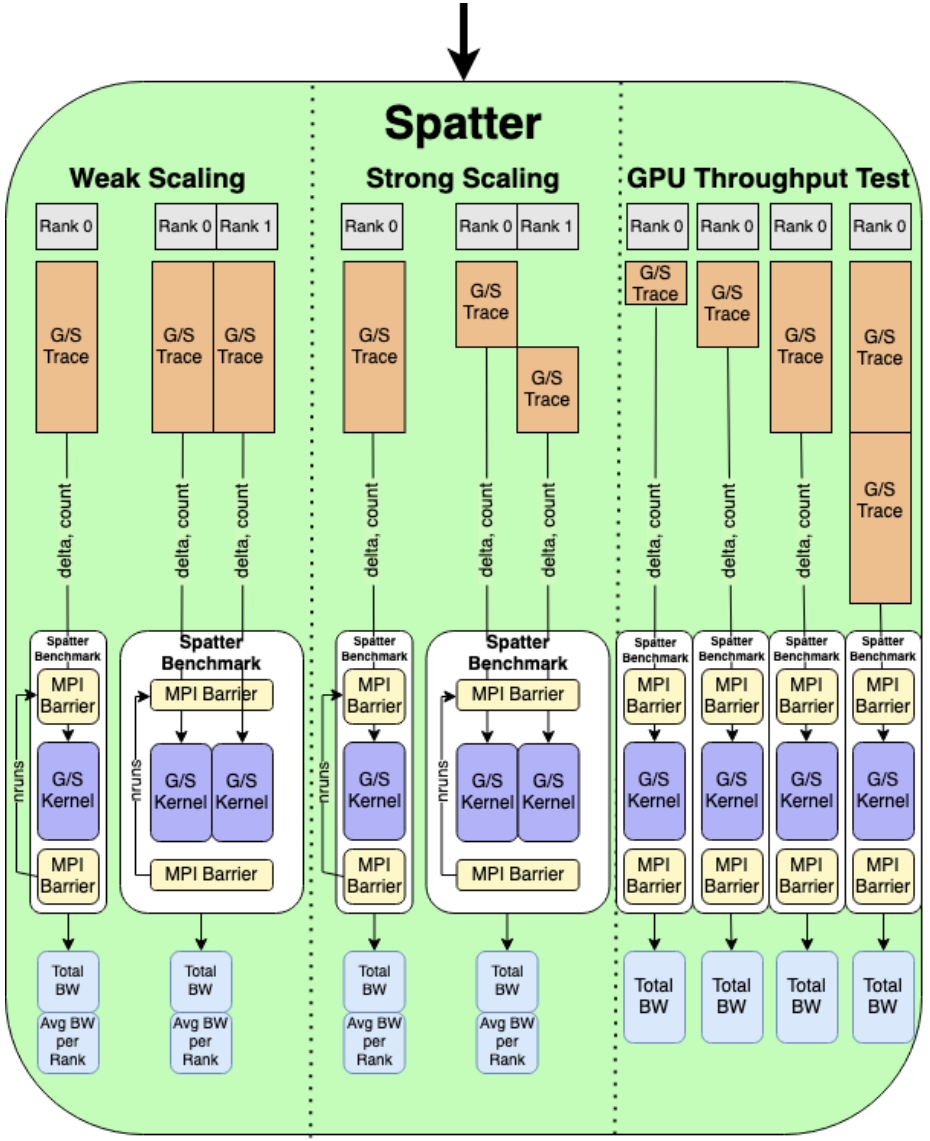
Sandia National Laboratories Georgia Tech Los Alamos NATIONAL LABORATORY

Spatter 2.0 MPI Workflow

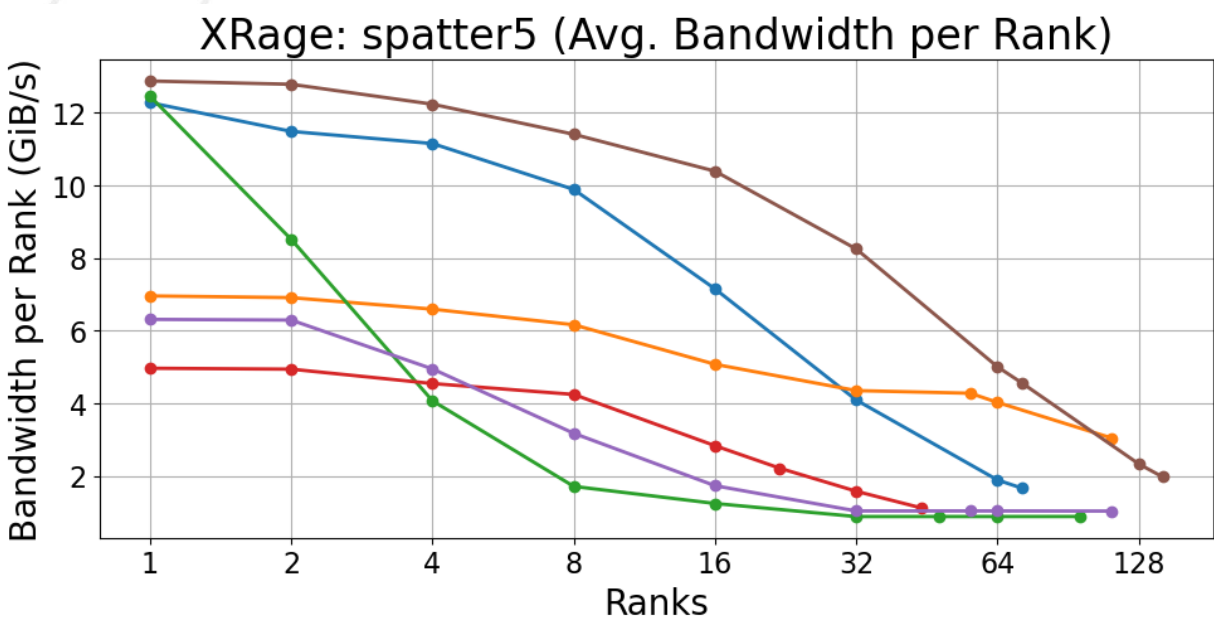
Weak Scaling – each MPI rank gets the same pattern and scaling scripts are used to sweep across N ranks

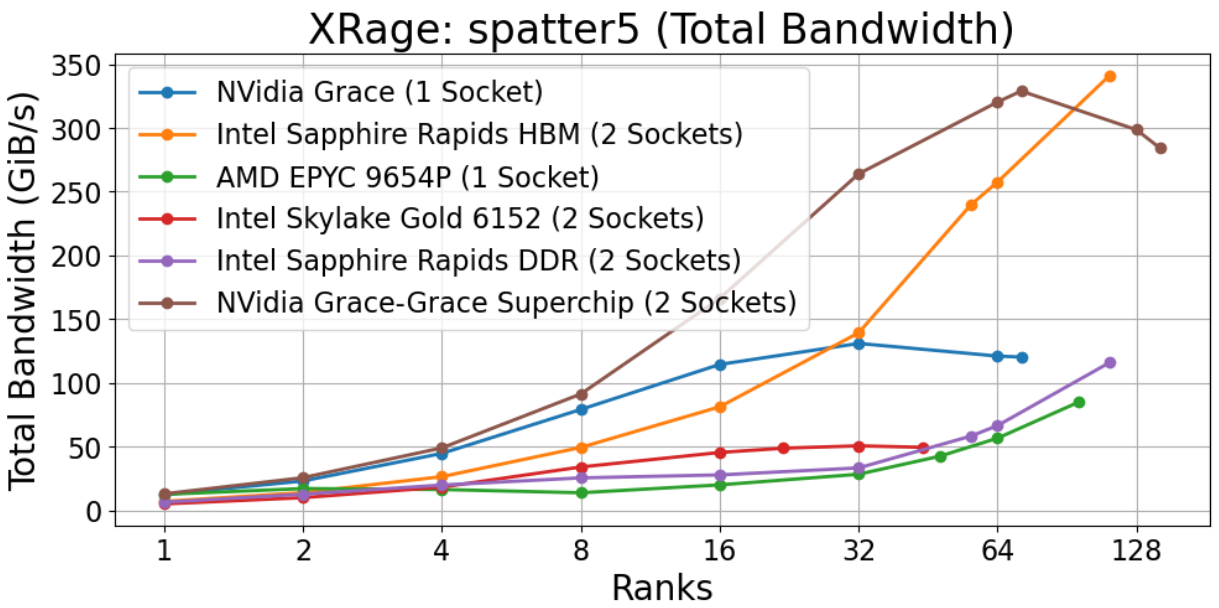
Strong Scaling – access patterns are partitioned across MPI ranks

GPU Throughput – patterns are truncated or expanded to vary amount of memory accesses and saturate GPU memory subsystem



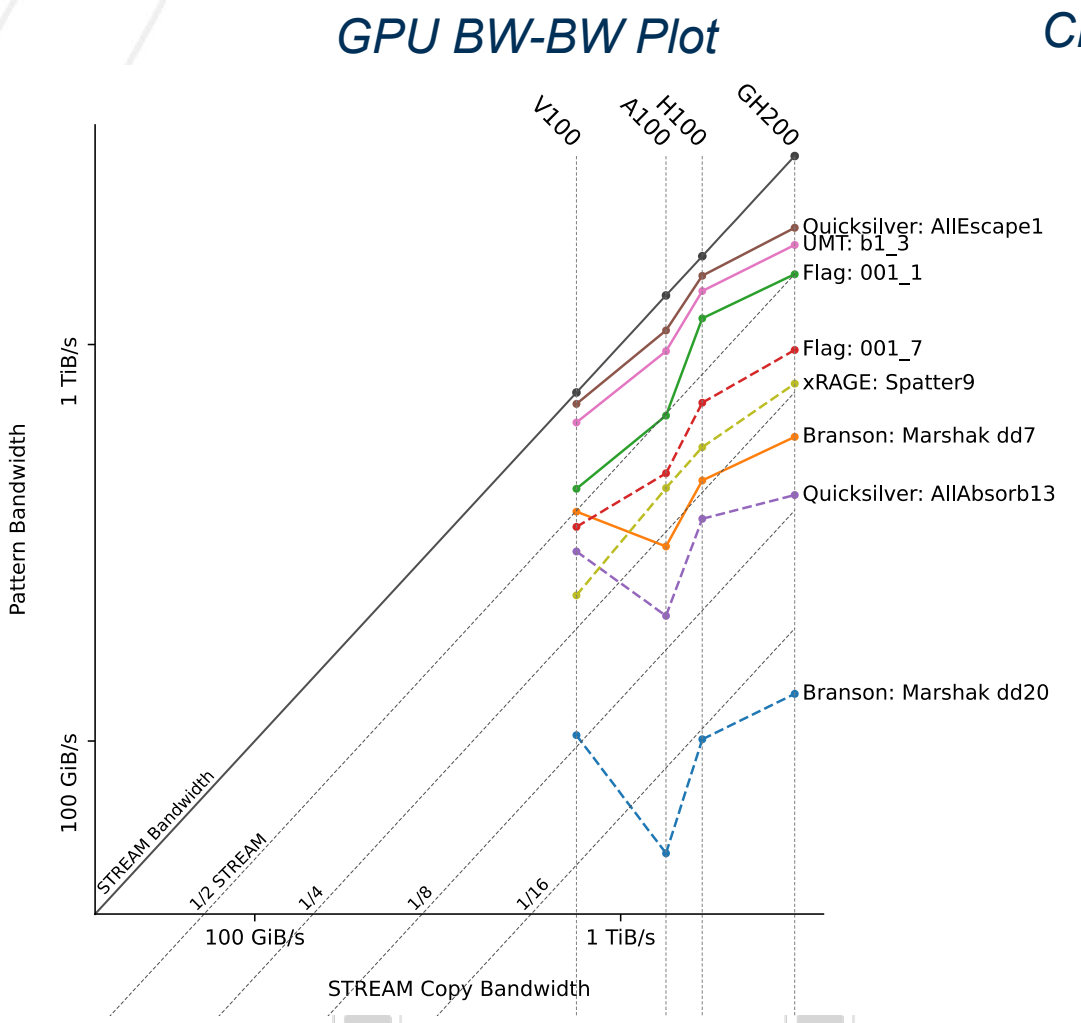
Spatter 2.0 Results

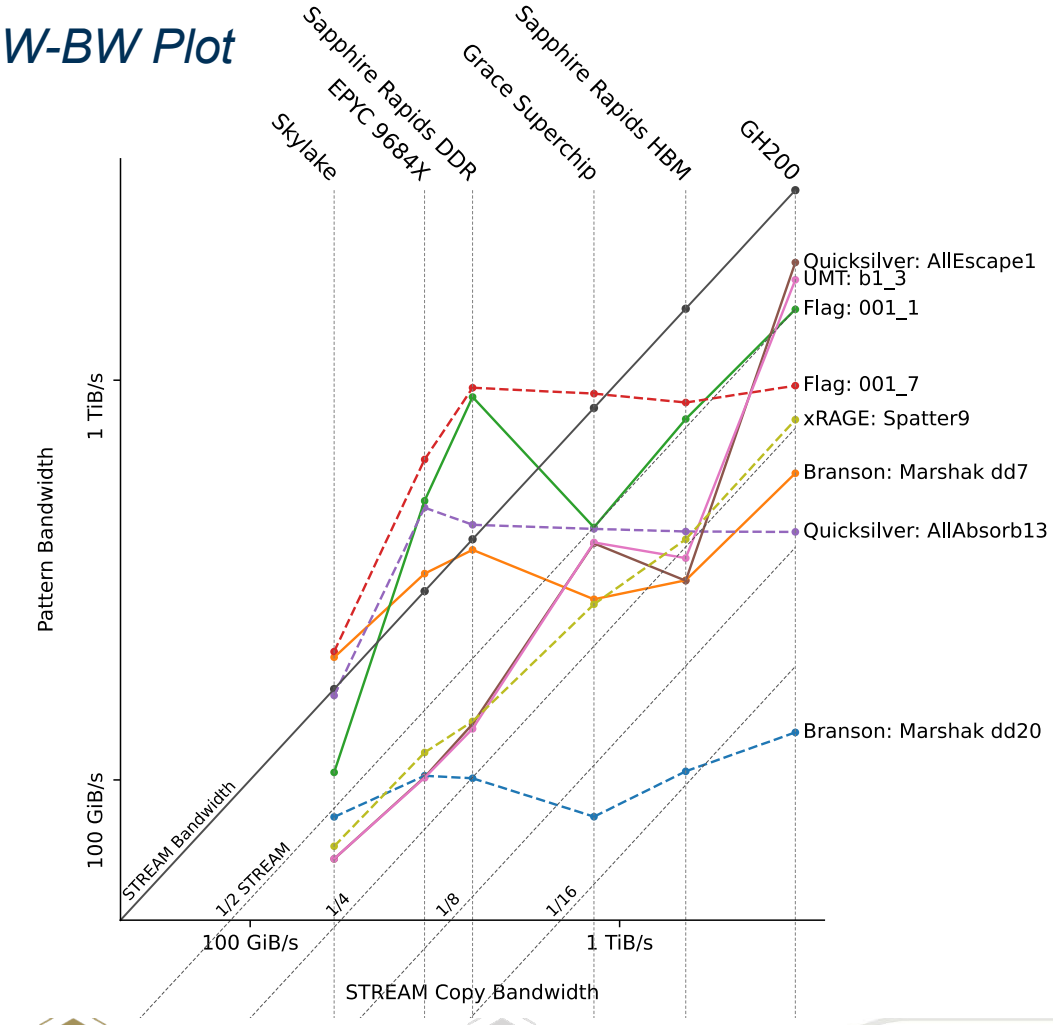




Weak Scaling MPI Tests for Xrage Gather pattern

Spatter 2.0 Results





Sandia National Laboratories Georgia Tech Los Alamos NATIONAL LABORATORY

Summary

Motivating work at LANL for ATS-5 for new memory accelerator microbenchmarks led to the release of GS Patterns and Spatter 2.0

- New capabilities allow for more complete analysis and capture of real-world application patterns

Improved workflow adds support for visualization of patterns as well as future work for simulation

- Much more work is needed to port to other ModSim tools and to do validation and analysis of patterns!

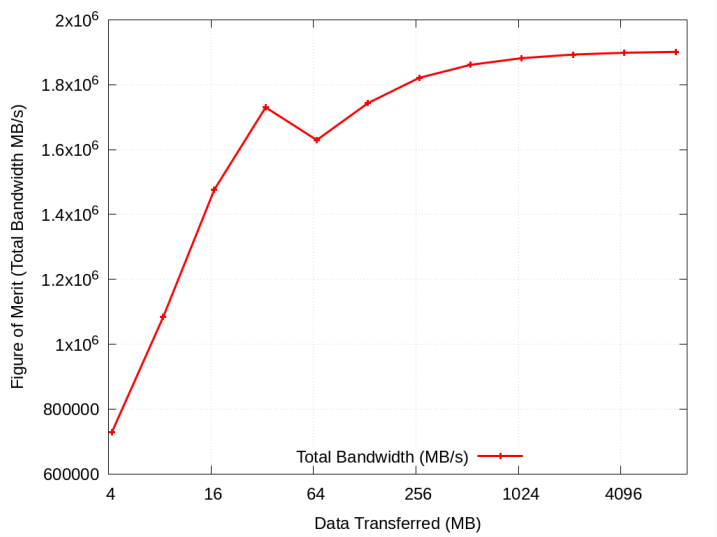


Fig. 3.14 *Spatter Throughput on H100 xRAGE Asteroid Pattern 5 on H100*

3.4.2.2. xRAGE Asteroid Spatter Pattern 9

Throughput experiment for the pattern in datafiles/xrage/asteroid/spatter9.json. Results will be found in spatter strongscaling/H100/xrage/asteroid/spatter9/ and Figures will be found in

Spatter page from https://lanl.github.io/benchmarks/

Acknowledgements

This work is supported by the NSF Rogues Gallery testbed grant (CNS-2016701)

This research used resources provided by the Darwin testbed at Los Alamos National Laboratory (LANL) which is funded by the Computational Systems and Software Environments subprogram of LANL's Advanced Simulation and Computing program (NNSA/DOE).

Kevin Sheridan, Galen Shipman, and Jered Dominguez-Trujillo acknowledge support by the National Nuclear Security Administration. Los Alamos National Laboratory is operated by Triad National Security, LLC for the U.S. Department of Energy under contract 89233218CNA000001; LA-UR-24-24856.

Sandia National Laboratories Georgia Tech Los Alamos NATIONAL LABORATORY

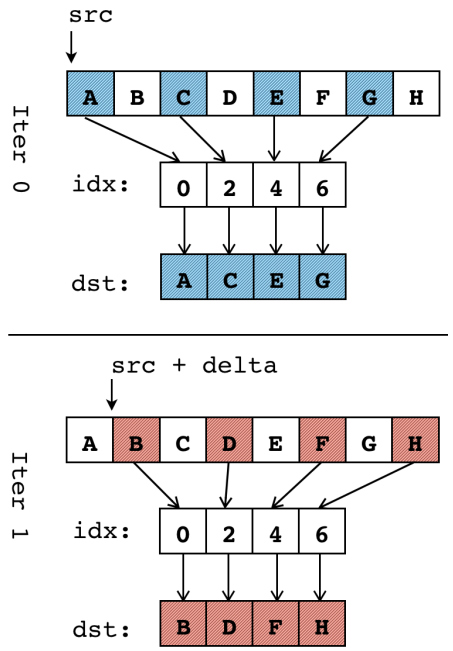
Thank You

$$\text{ScatterWindow} = \begin{array}{c} \begin{array}{cccc} \text{maddr1} & \text{maddr2} & \ldots & \text{maddrN} \end{array} \\ \begin{bmatrix} 0x0480 & 0x0488 & \ldots & 0x1488 \\ 0x1780 & 0x1788 & \ldots & 0x0783 \\ \ldots & \ldots & \ldots & \ldots \\ 0x9580 & 0x5588 & \ldots & 0x3581 \end{bmatrix} \end{array} \begin{array}{c} iaddr \\ 0x0001 \\ 0x0003 \\ \ldots \\ 0x1019 \end{array}$$

GS Patterns codebase
https://github.com/lanl/gs_patterns

Spatter
Mainline: https://github.com/hpcgarage/spatter
ATS-5: https://github.com/lanl/spatter

Spatter Patterns
https://github.com/hpcgarage/spatter-patterns



Sandia National Laboratories
Georgia Tech
Los Alamos NATIONAL LABORATORY