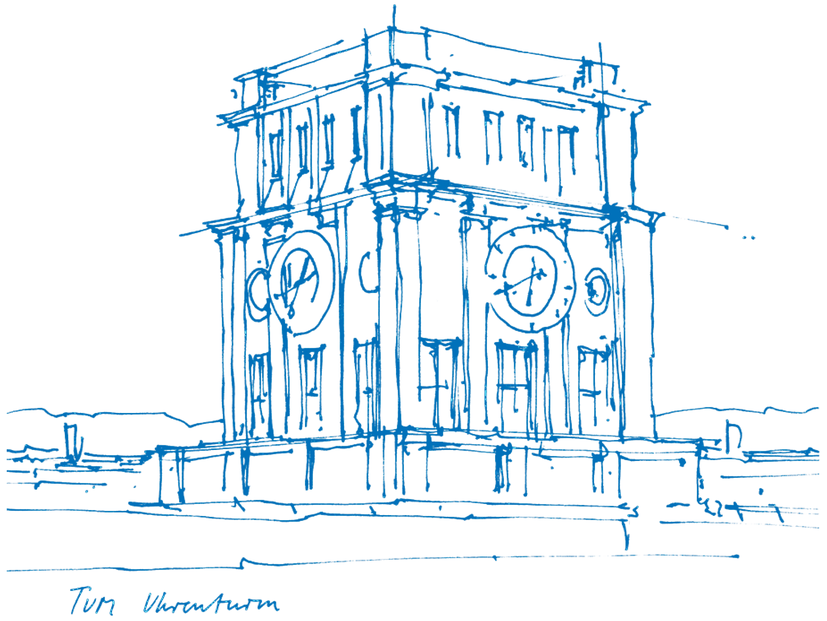
Managing Heterogeneous Topologies and Understanding Their Impact on Performance

Doctoral Showcase

Stepan Vanecek
stepan.vanecek@tum.de

Chair of Computer Architecture and Parallel Systems
Technical University of Munich



Doctoral Showcase Poster SC'25, 20th November 2025

Motivation

Motivation

1 Topologies are increasingly complex

- Heterogeneous
- Dynamic, configurable

Motivation

1 Topologies are increasingly complex

- Heterogeneous
- Dynamic, configurable

2 Memories play a crucial role

- More parallelism in compute
  - ➡ Higher demand for data
- More complex memory hierarchies
  - ➡ More difficult to understand and tune performance
- More heterogeneous systems, chips, memories
  - ➡ Portability issues

Toolchain Overview

	Topology	GPUs	Memory	Performance Analysis and Modelling	Heterogeneous Chip and Memory Architectures
sys-sage	✅	☑	✅	☑	✅
MT4G	✅	✅	✅	☑	☑
GPUscout	☑	✅	✅	✅	
CXL data transfer modelling	☑		✅	✅	✅
Mitos-MemAxes toolchain	☑		✅	✅	

✅ = core focus ; ☑ = related topic

1. sys-sage

1. sys-sage

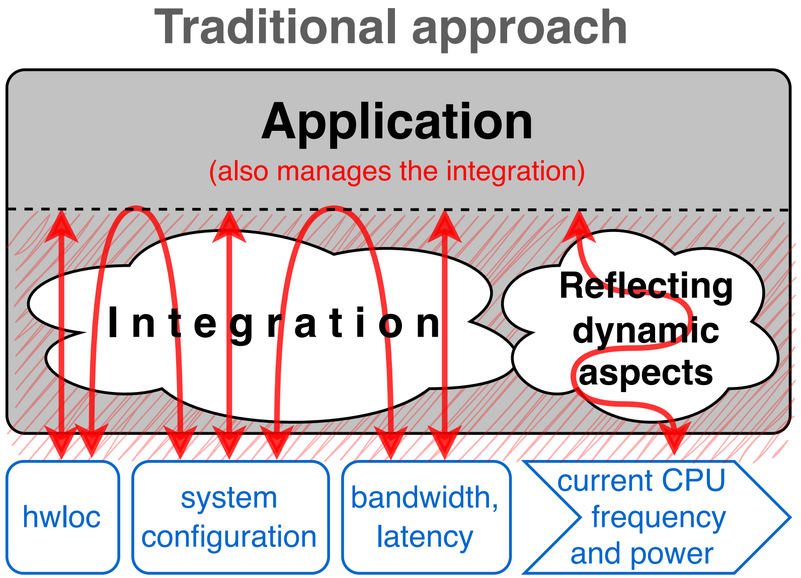
? Current landscape of topological information

- Multiple tools (such as hwloc) provide static topological snapshot
- Increased need for dynamic/runtime data (interfaces also exist)
- Need to integrate multiple static and dynamic contexts ➡ manual integration

1. sys-sage

? Current landscape of topological information

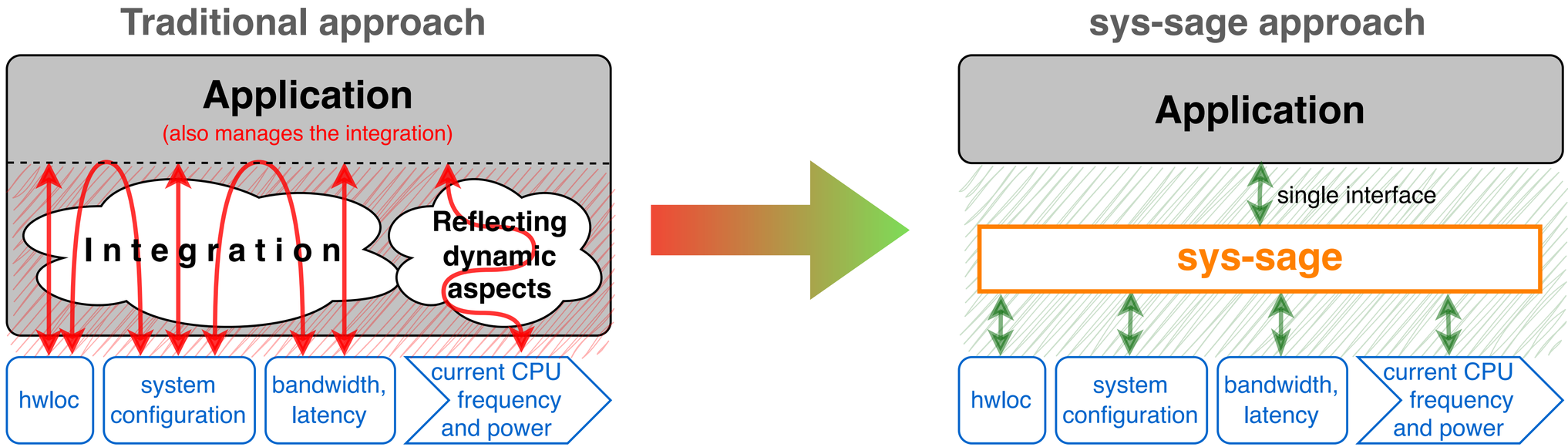
- Multiple tools (such as hwloc) provide static topological snapshot
- Increased need for dynamic/runtime data (interfaces also exist)
- Need to integrate multiple static and dynamic contexts ➡ manual integration



1. sys-sage

Current landscape of topological information

- Multiple tools (such as hwloc) provide static topological snapshot
- Increased need for dynamic/runtime data (interfaces also exist)
- Need to integrate multiple static and dynamic contexts ➡ manual integration



1. sys-sage

1. sys-sage

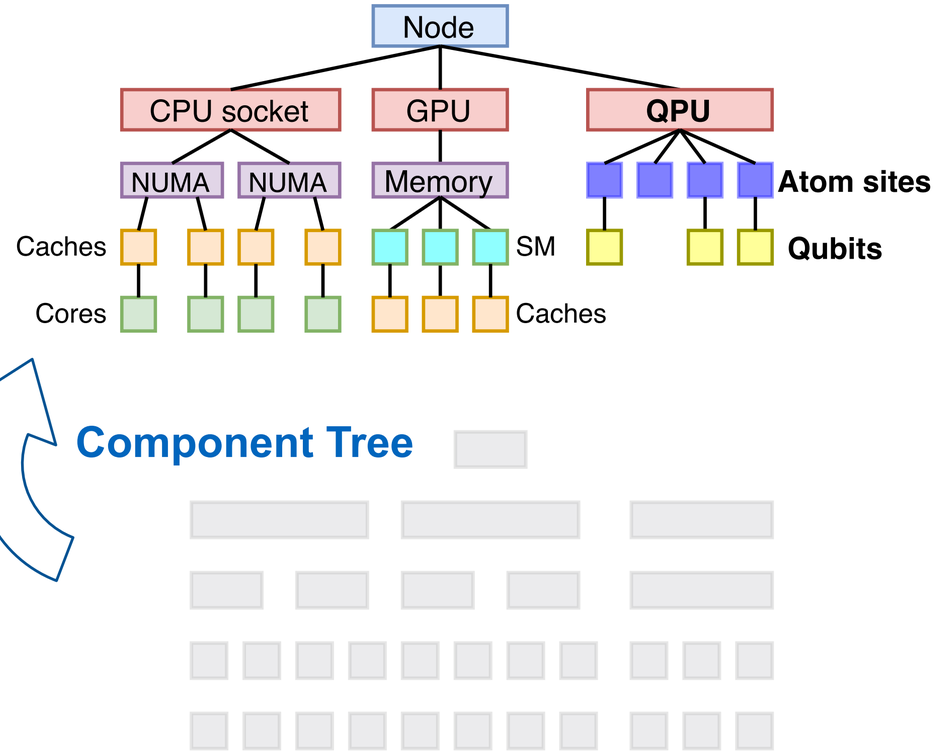
Internal Data Representation

1. sys-sage

Internal Data Representation

1. Components

- Hierarchical representation (hwloc-like)
- Simple to understand, Mandatory
- Examples: Static CPU/GPU/QPU HW topologies



1. sys-sage

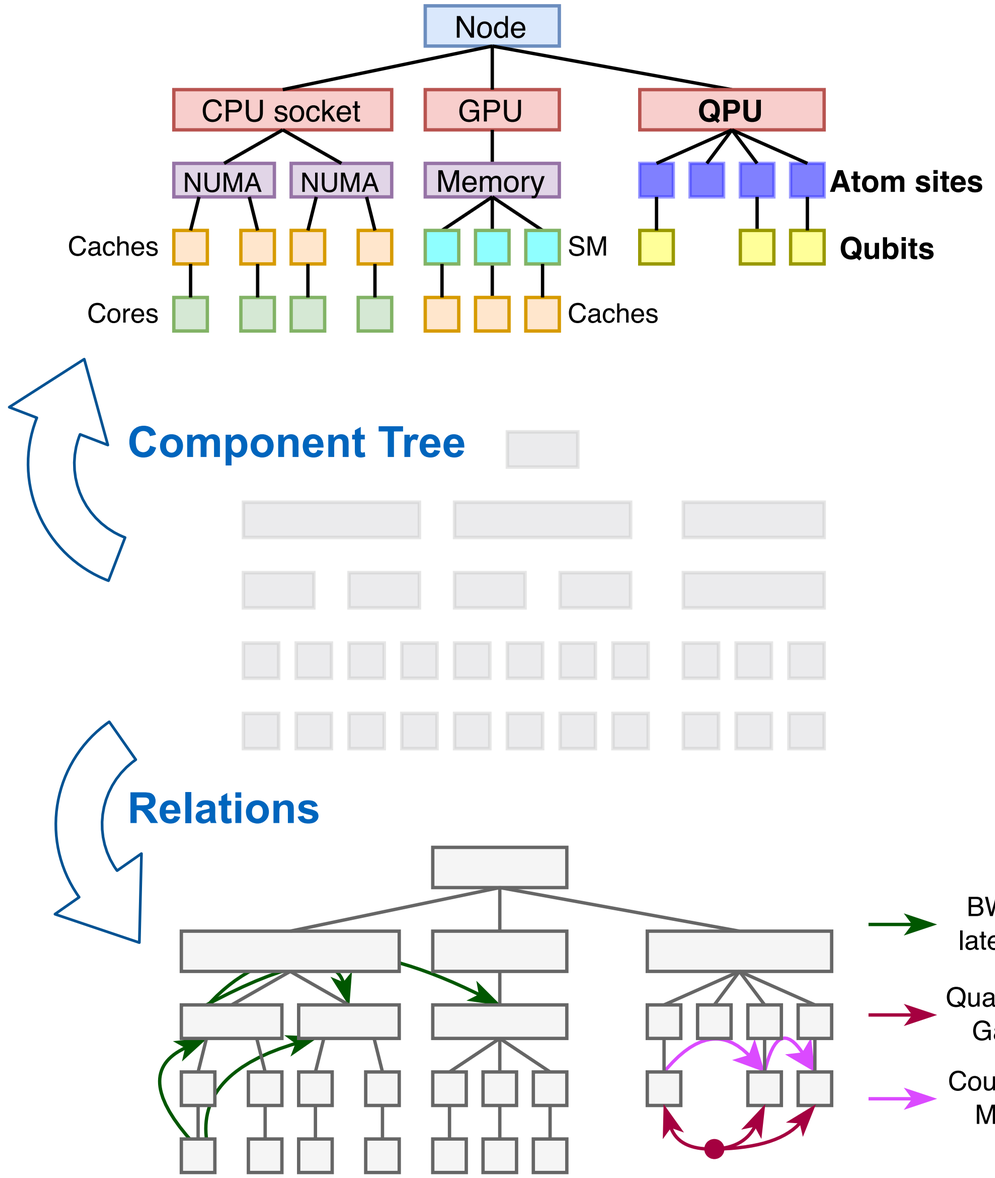
Internal Data Representation

1. Components

- Hierarchical representation (hwloc-like)
- Simple to understand, Mandatory
- Examples: Static CPU/GPU/QPU HW topologies

2. Relations

- Any Relation of 1 or more Components
- Orthogonal to the Component Tree
- More dynamic, complex information
- Examples: data exchange rates between components, dynamic settings, HW counter readings



1. sys-sage

1. sys-sage

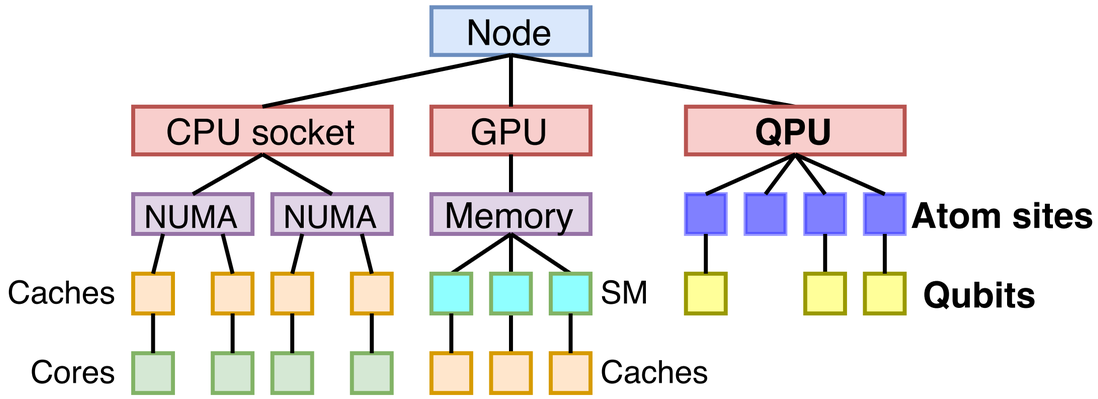
Quantum Extension

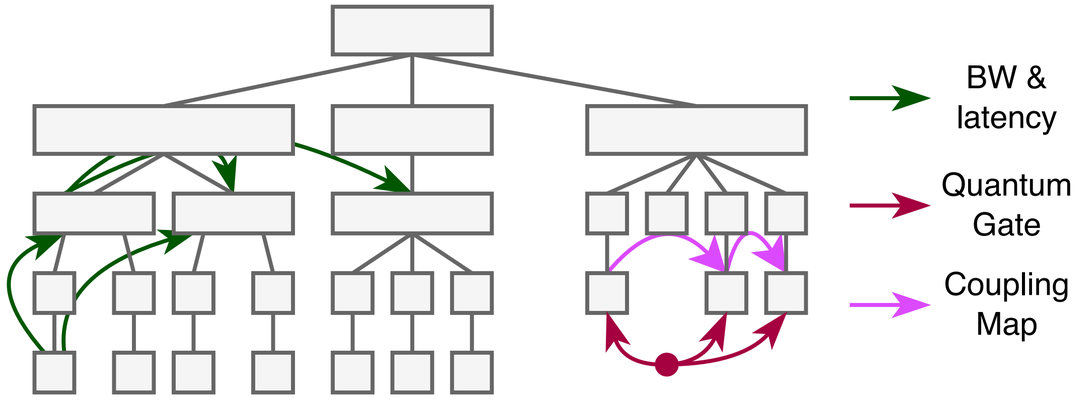
- Quantum Processing Units (QPUs) begin to accompany CPUs and GPUs in HPC systems
- sys-sage was extended to support QPUs alongside CPUs/GPUs
- Possible to retrieve information from existing APIs, such as QDMI or Qiskit runtime

1. sys-sage

Quantum Extension

- Quantum Processing Units (QPUs) begin to accompany CPUs and GPUs in HPC systems
- sys-sage was extended to support QPUs alongside CPUs/GPUs
- Possible to retrieve information from existing APIs, such as QDMI or Qiskit runtime

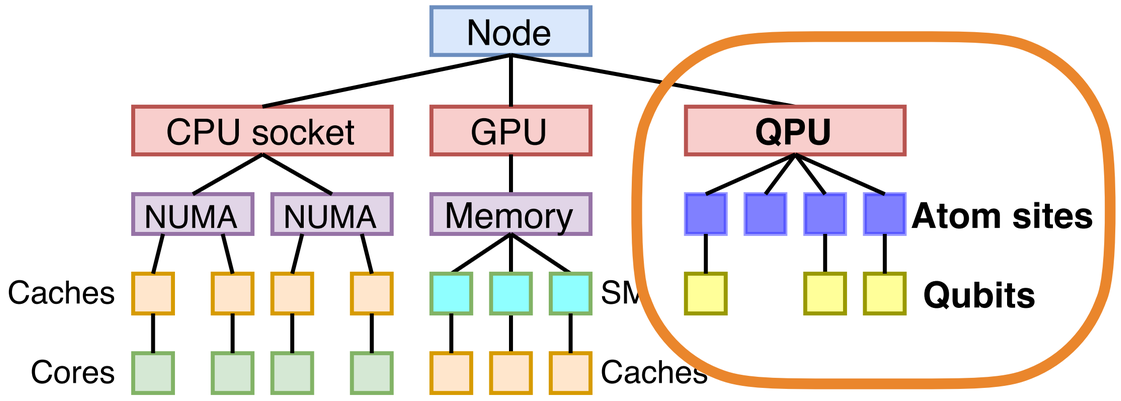


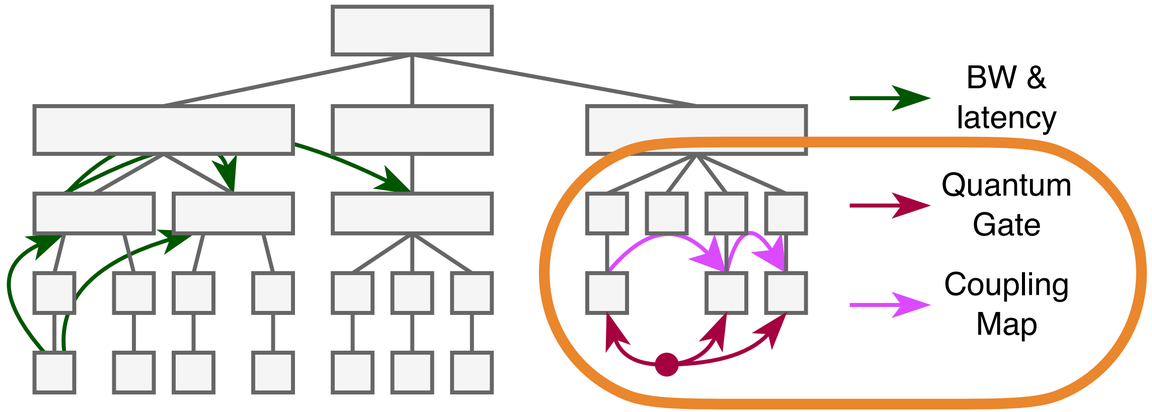


1. sys-sage

Quantum Extension

- Quantum Processing Units (QPUs) begin to accompany CPUs and GPUs in HPC systems
- sys-sage was extended to support QPUs alongside CPUs/GPUs
- Possible to retrieve information from existing APIs, such as QDMI or Qiskit runtime



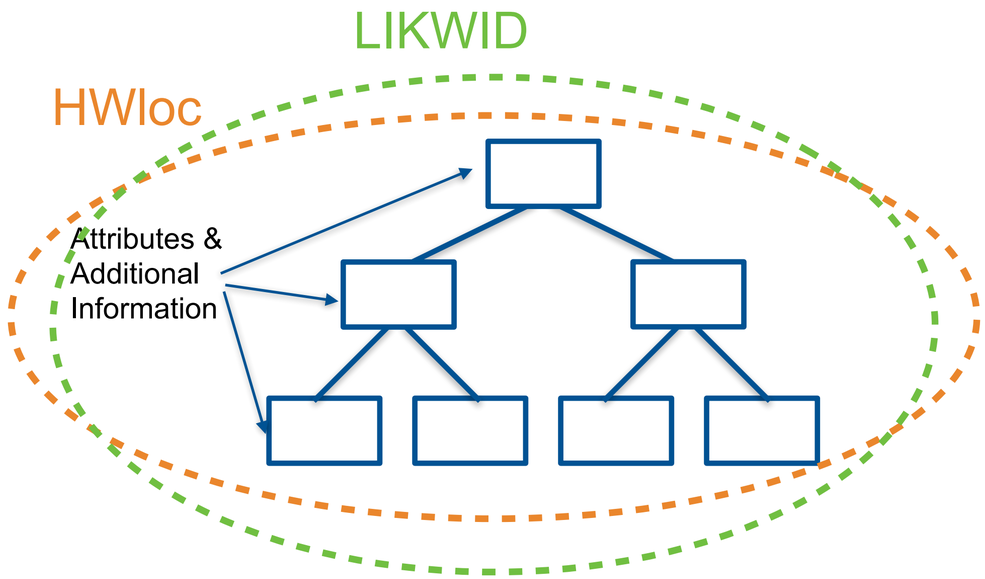


2. MT4G — Motivation

2. MT4G — Motivation

CPUs

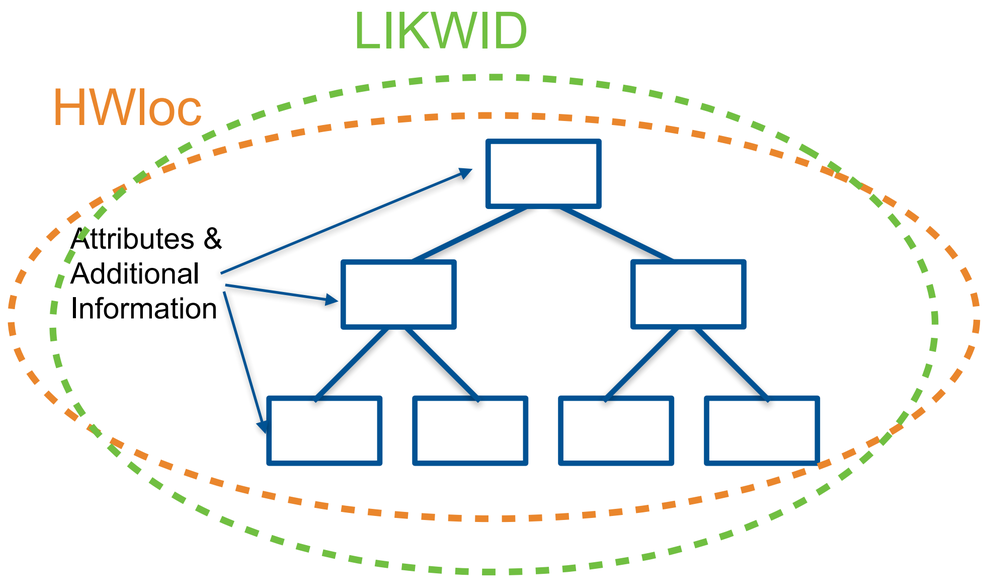
Complete, cross-vendor



2. MT4G — Motivation

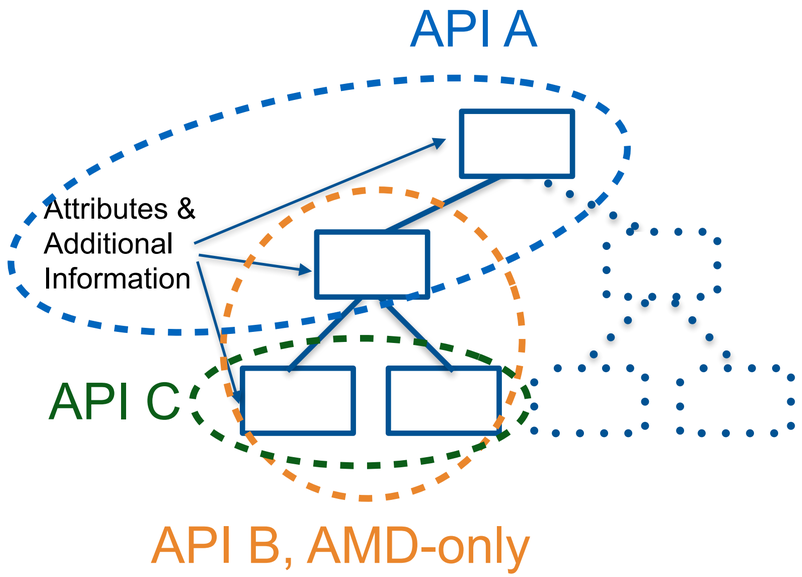
CPUs

Complete, cross-vendor



GPUs

Incomplete, scattered, vendor-specific



2. MT4G — Reverse Engineering GPU Topologies

2. MT4G — Reverse Engineering GPU Topologies

What does MT4G provide?

Report on GPU building blocks and their attributes and capabilities

- NVIDIA (Pascal and newer)
- AMD (CDNA)

2. MT4G — Reverse Engineering GPU Topologies

What does MT4G provide?

Report on GPU building blocks and their attributes and capabilities

- NVIDIA (Pascal and newer)
- AMD (CDNA)

Target Memory Elements

NVIDIA

- L1 cache
- L2 cache
- Texture & Readonly cache
- Constant L1, L1.5 cache
- Shared memory
- Device memory

AMD

- L1 cache
- L1s cache
- L2 cache
- L3 cache (partially)
- Local Data Share
- Device memory

2. MT4G — Reverse Engineering GPU Topologies

What does MT4G provide?

Report on GPU building blocks and their attributes and capabilities

- NVIDIA (Pascal and newer)
- AMD (CDNA)

Target Memory Elements

NVIDIA

- L1 cache
- L2 cache
- Texture & Readonly cache
- Constant L1, L1.5 cache
- Shared memory
- Device memory

AMD

- L1 cache
- L1s cache
- L2 cache
- L3 cache (partially)
- Local Data Share
- Device memory

Queried properties

- Size
- Cache Line Size ($-only)
- Fetch Granularity ($-only)
- Physical Layout
- Load Latency
- Read & Write Bandwidth (L2 cache and above)

\+ Compute Resource Topological Information

2. MT4G — Coverage

✅ = Available via microbenchmark; ☑️ = Available via API; ➡️SOON = Bandwidth only for higher-level elements; ❌ = not available

2. MT4G — Coverage

Memory Element	Size		LD lat		R&W BW		cache line size		fetch granularity		Amount per SM/GPU		Physical hardware sharing	
Vendor	*NV*	*AMD*	*NV*	*AMD*	*NV*	*AMD*	*NV*	*AMD*	*NV*	*AMD*	*NV*	*AMD*	*NV*	*AMD*
(v) L1 cache	✅	✅	✅	✅	SOON	SOON	✅	✅	✅	✅	✅	✅	✅	
sL1 cache		✅		✅		SOON		✅		✅				✅
L2 cache	☑	☑	✅	✅	✅	✅	✅	☑	✅	✅	✅	☑		
L3 cache		☑		❌		✅		☑		❌		☑		
Texture cache	✅		✅		SOON		✅		✅		✅		✅	
Readonly cache	✅		✅		SOON		✅		✅		✅		✅	
Constant L1 cache	✅		✅		SOON		✅		✅		✅		✅	
Constant L1.5 cache	✅		✅		SOON		❌		✅		❌			
Shared Mem. / LDS	☑	☑	✅	✅	SOON	SOON								
Device Memory	☑	☑	✅	✅	✅	✅								

✅ = Available via microbenchmark; ☑ = Available via API; SOON = Bandwidth only for higher-level elements; ❌ = not available

3. GPUscout — Locating memory bottlenecks on GPUs

3. GPUscout — Locating memory bottlenecks on GPUs

? Understanding and optimizing data movements on GPUs is important

- Complex memory subsystem
- Massive parallelism
  - ➡ potential stalls on many cores
- Less transparent scheduling

3. GPUscout — Locating memory bottlenecks on GPUs

? Understanding and optimizing data movements on GPUs is important

- Complex memory subsystem
- Massive parallelism
  - ➡ potential stalls on many cores
- Less transparent scheduling

➡ GPUscout combines three views on the data

1. Static assembly analysis
2. Sampling warp stalls
3. Collecting performance metrics

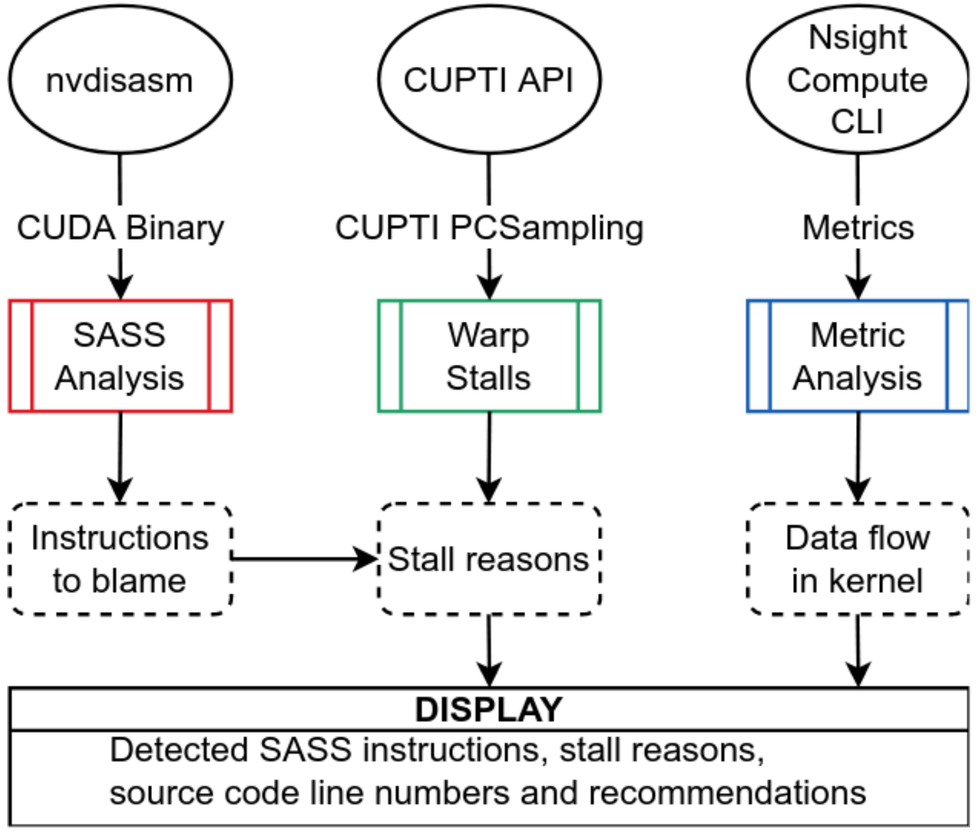
3. GPUscout — Locating memory bottlenecks on GPUs

? Understanding and optimizing data movements on GPUs is important

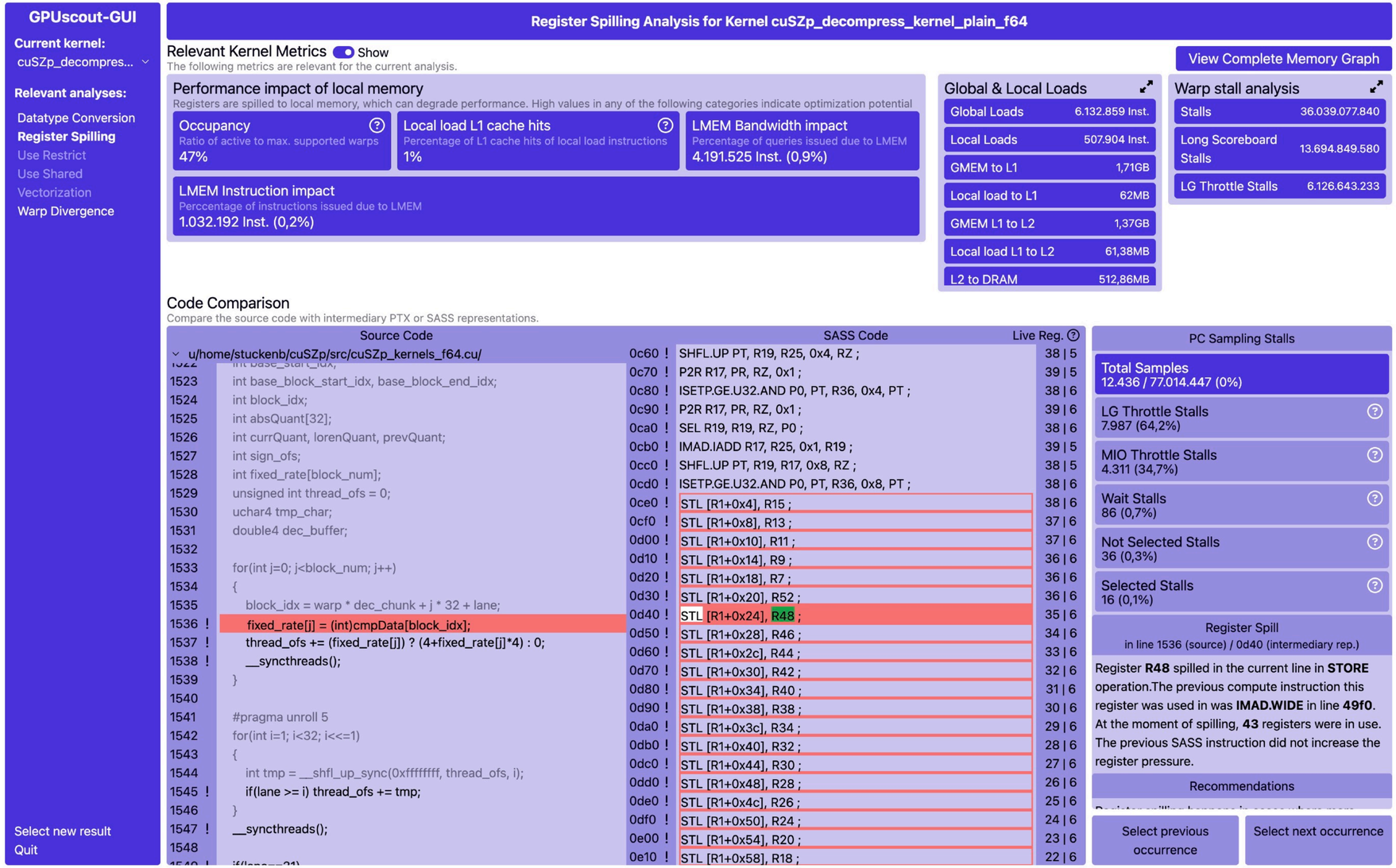
- Complex memory subsystem
- Massive parallelism
  - ➡ potential stalls on many cores
- Less transparent scheduling

➡ GPUscout combines three views on the data

1. Static assembly analysis
2. Sampling warp stalls
3. Collecting performance metrics

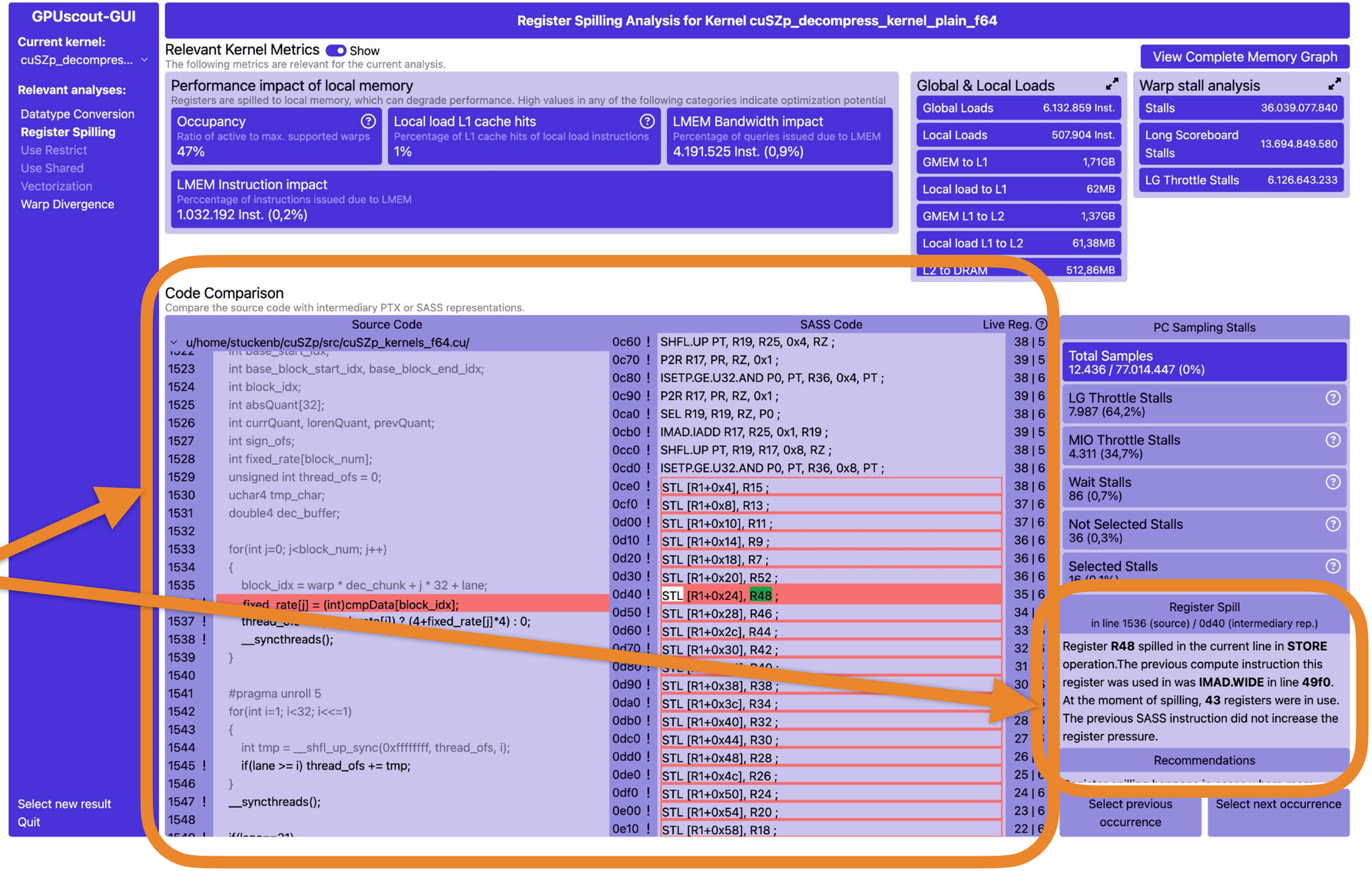


3. GPUscout — GUI



3. GPUscout — GUI

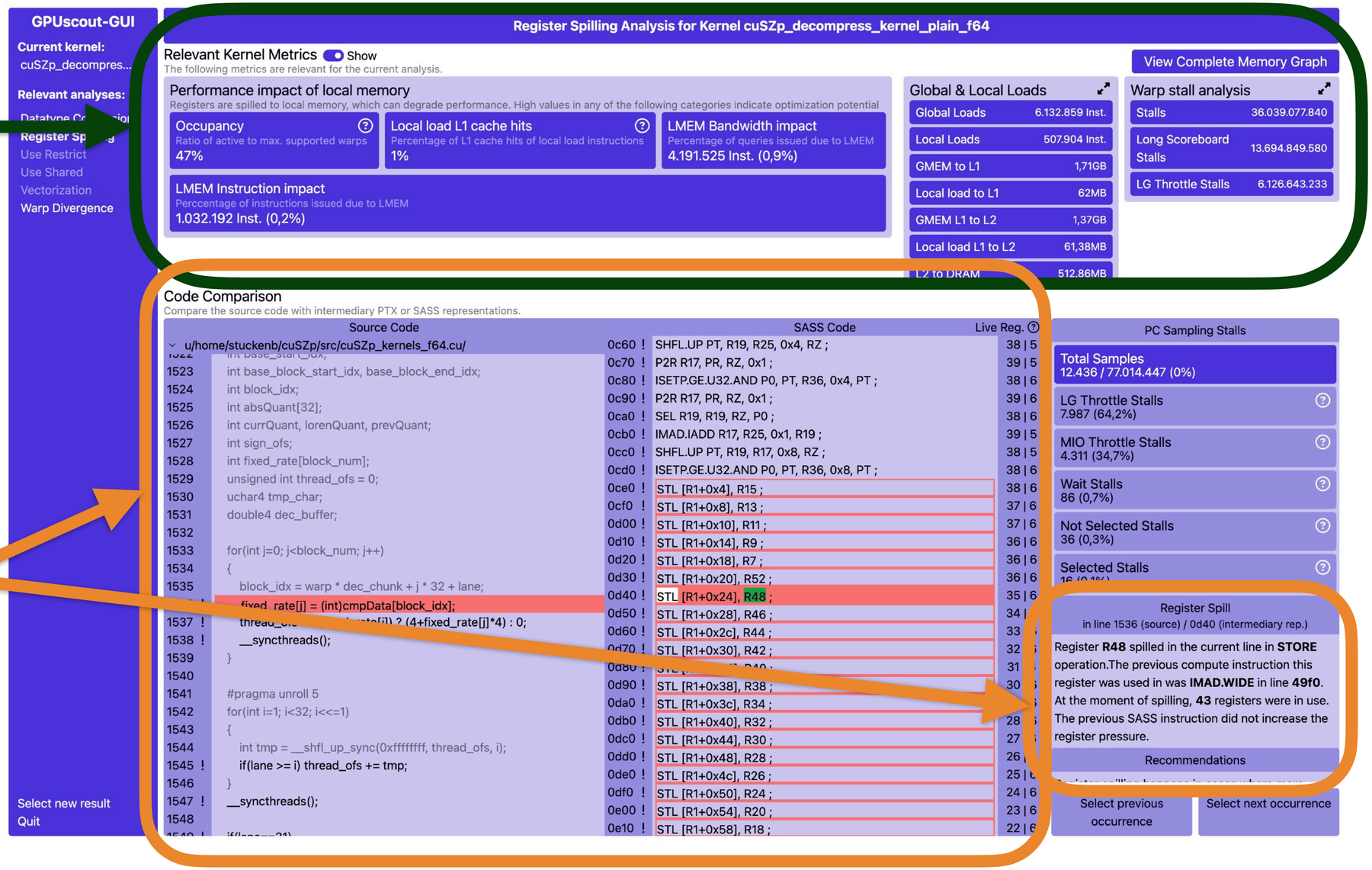
Static Code Analysis



3. GPUscout — GUI

Kernel Metrics

Static Code Analysis

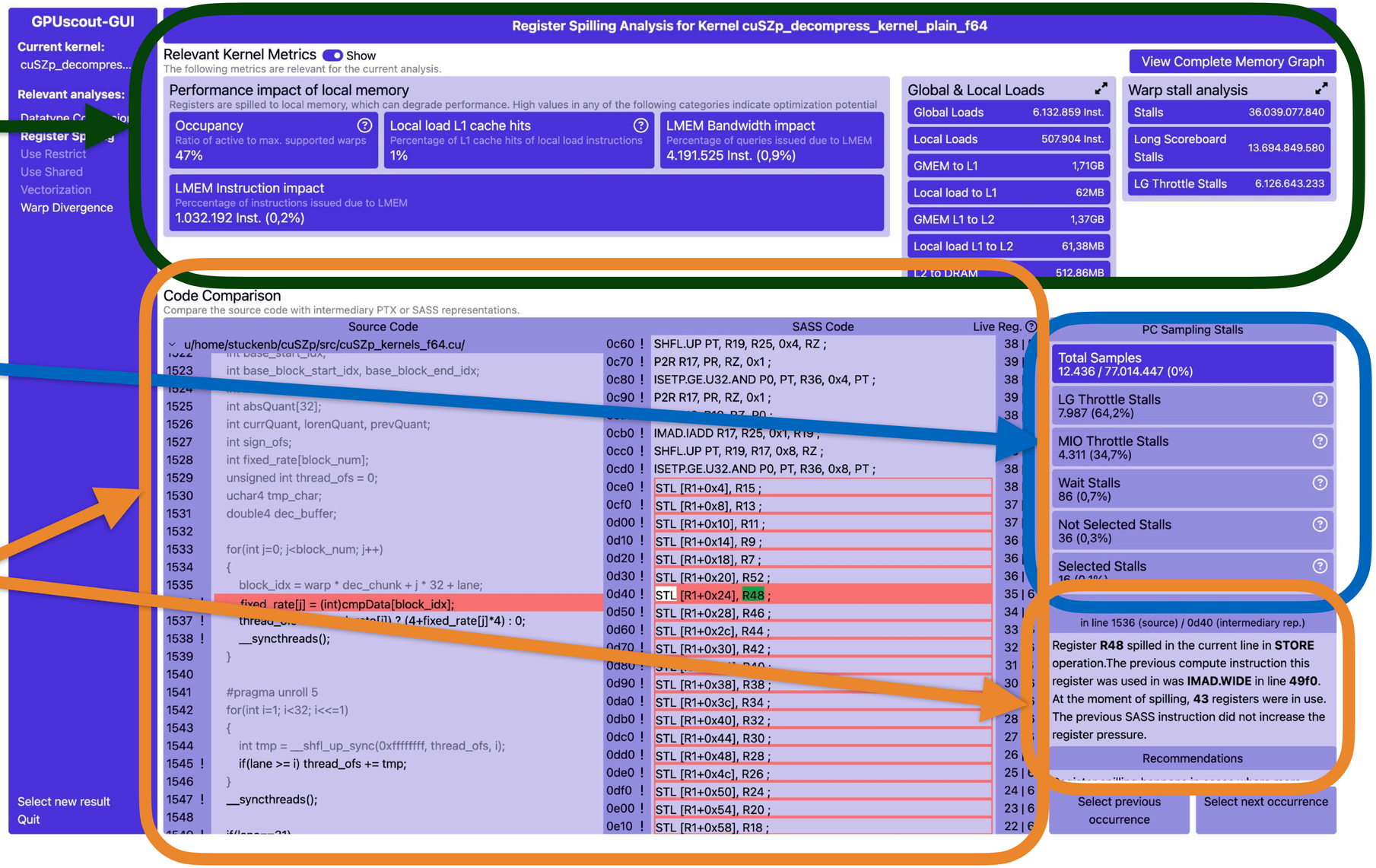


3. GPUscout — GUI

Kernel Metrics

Warp Stalls

Static Code Analysis



4. Modeling Cross-node Communication via CXL.mem

4. Modeling Cross-node Communication via CXL.mem

CXL.mem 3.0+ : Multi-node shared memory pools

? Which MPI buffers should be replaced by CXL.mem?
? What is the priority and what is the expected performance gain?
? What system attributes are needed to efficiently replace MPI with CXL?

4. Modeling Cross-node Communication via CXL.mem

CXL.mem 3.0+ : Multi-node shared memory pools

? Which MPI buffers should be replaced by CXL.mem?
? What is the priority and what is the expected performance gain?
? What system attributes are needed to efficiently replace MPI with CXL?

Model performance of MPI vs. CXL.mem communication

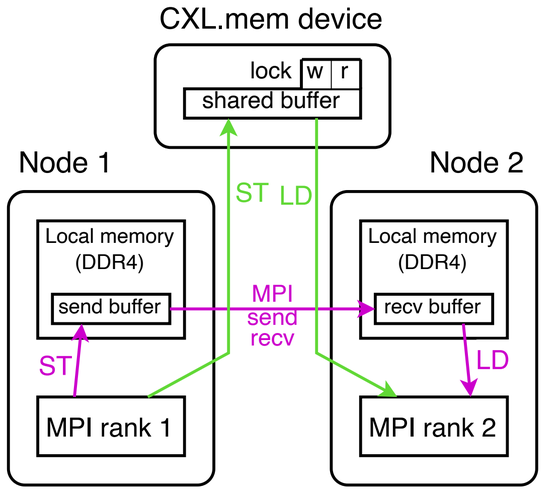
4. Modeling Cross-node Communication via CXL.mem

CXL.mem 3.0+ : Multi-node shared memory pools

? Which MPI buffers should be replaced by CXL.mem?
? What is the priority and what is the expected performance gain?
? What system attributes are needed to efficiently replace MPI with CXL?

Model performance of MPI vs. CXL.mem communication

1. Impact on cross-node communication



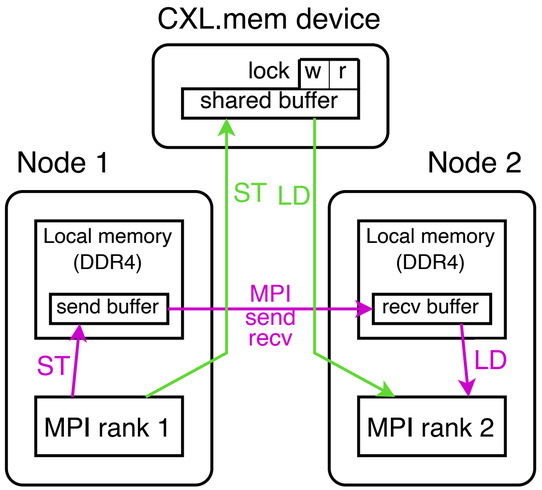
4. Modeling Cross-node Communication via CXL.mem

CXL.mem 3.0+ : Multi-node shared memory pools

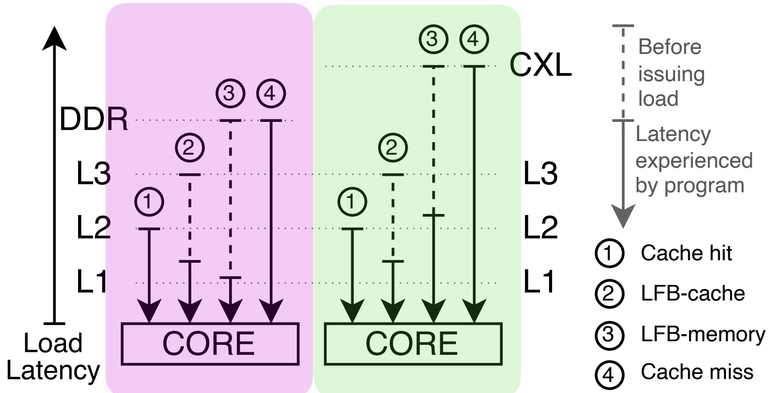
? Which MPI buffers should be replaced by CXL.mem?
? What is the priority and what is the expected performance gain?
? What system attributes are needed to efficiently replace MPI with CXL?

Model performance of MPI vs. CXL.mem communication

1. Impact on cross-node communication



2. Impact on data access patterns



Managing Heterogeneous Topologies and Understanding Their Impact on Performance

Toolchain for understanding system topologies and impact of data transfers on system performance

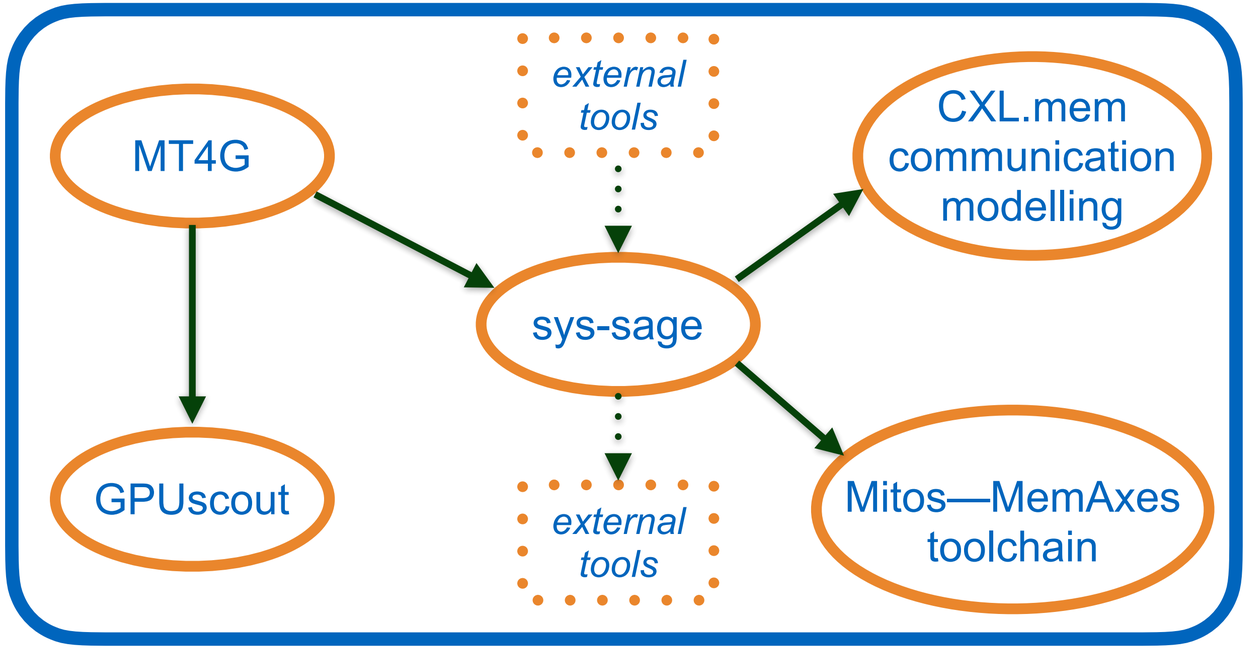
4 Main Efforts/Tools:

MT4G (GPU topology discovery)

sys-sage (system topology information management)

GPUscout (GPU memory-related bottleneck analysis)

CXL.mem communication modelling
(Modelling cross-node data transfer over CXL)



Stepan Vanecek
stepan.vanecek@tum.de

Doctoral Showcase **#141**

Technical University of Munich
Chair of Computer Architecture and Parallel Systems (CAPS)

SEANERGYS Energy Efficient Exascale
PLASMA PEPSC
DEEP-SEA
SPONSORED BY THE
Federal Ministry of Education and Research

Abstract

To solve increasingly complex problems more efficiently, modern HPC systems feature highly heterogeneous components: CPUs, GPUs, and recently QPUs (Quantum Processing Units), each with a unique, complex compute topology. The massive parallelism of GPUs, combined with emerging memory technologies on CPUs and GPUs, makes the memory topologies increasingly heterogeneous, complex, and dynamically configurable. Understanding these topological details, especially regarding available memory and its usage, is essential to operating the systems and applications efficiently.

This thesis presents a framework targeting several fundamental gaps in the currently available research and tooling: sys-sage, MT4G, GPUscout, and Mitos modeling. At the core, the sys-sage library offers a unified approach to maintaining static and dynamic topological information from different sources and APIs. Its universal architecture handles CPUs, GPUs, and QPUs alike.

MT4G provides an otherwise unavailable, vendor-agnostic, and complete report on GPU memory topologies, integrable with sys-sage. GPUs' massive parallelism amplifies the potential performance penalties of improper cache and memory usage. Therefore, GPUscout identifies root causes of frequently-occurring memory-related bottlenecks, helping users efficiently utilize the complex memory subsystem of GPUs.

Finally, to address emerging memory technologies, such as CXL.mem, this thesis presents a novel data access modeling workflow as an extension of Mitos. The model predicts the performance impact of CXL.mem-based cross-node shared-buffer data exchange as an alternative to point-to-point MPI communication.

Altogether, these tools capture topologies of HPC systems and provide missing insights into application data transfer behavior.