GPUscout

Locating Data Movement-related Bottlenecks on (for now NVidia) GPUs

Stepan Vanecek

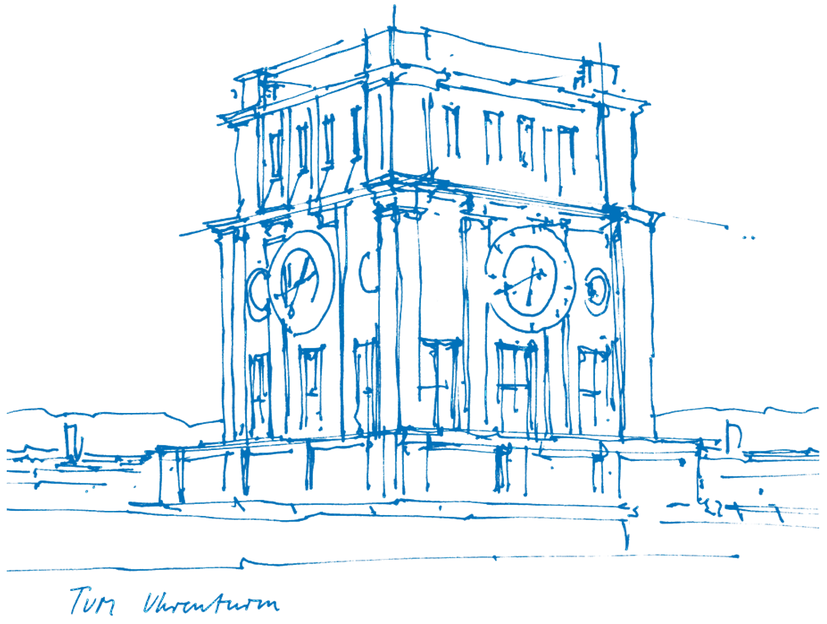
stepan.vanecek@tum.de

Research Group Martin Schulz

Chair of Computer Architecture and Parallel Systems

Technical University of Munich

Scalable Tools Workshop, 12th August 2024



GPUscout

A tool for identifying data movement-related bottlenecks in GPU kernels

- *Only data movements within a GPU*
- *Only NVidia GPUs for now*

Main objectives

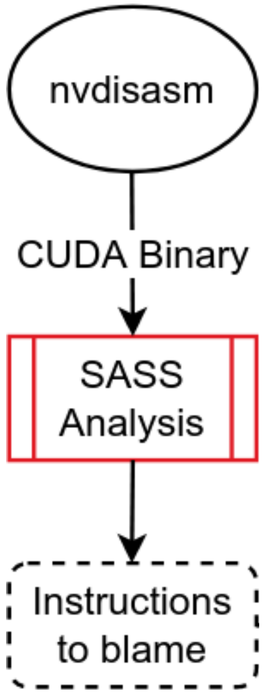
- Discovers and identifies the problematic behaviour
- Provides information about the severity and offers metrics to verify improvements
- Points the user directly to detected instruction and line of source code

What GPUscout builds on?

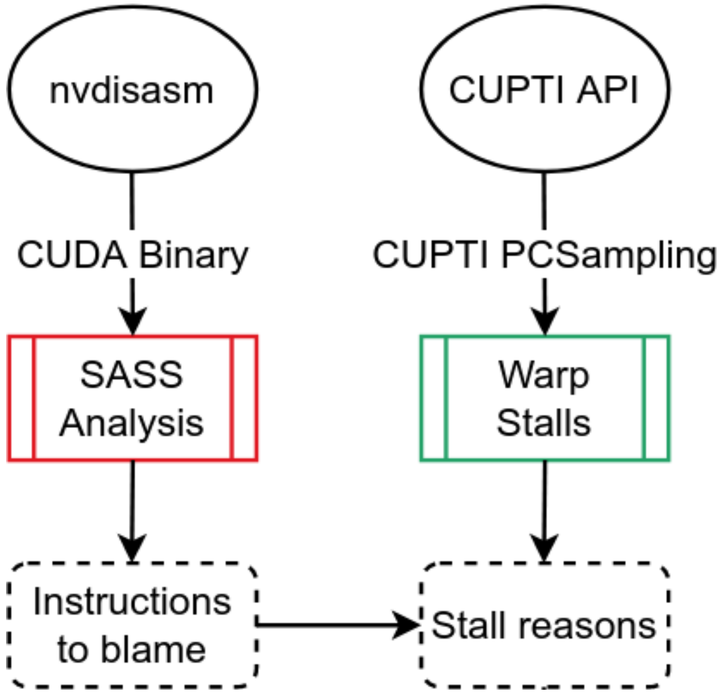
…on NVidia GPUs.

1. **SASS** (and PTX)
    - Assmeblies in CUDA
2. CUPTI
    - Provides data for profiling and tracing tools
    - GPUscout uses the **PC Sampling API** of CUPTI (Warp stalls)
        - Provides: stall reasons, source code line
3. Nsight Compute CLI (**ncu**)
    - Kernel-wide performance counters

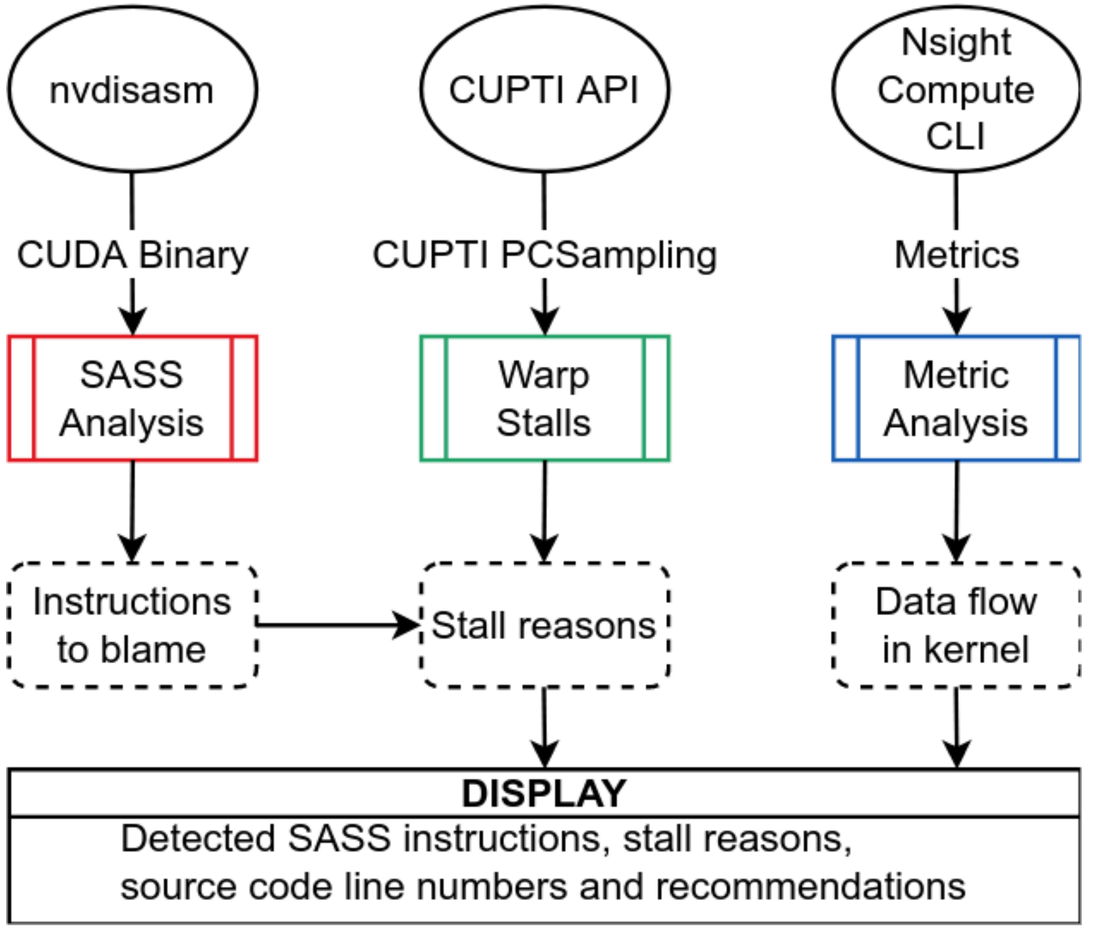
Architecture of GPUscout



Architecture of GPUscout



Architecture of GPUscout



Bottlenecks Analysis

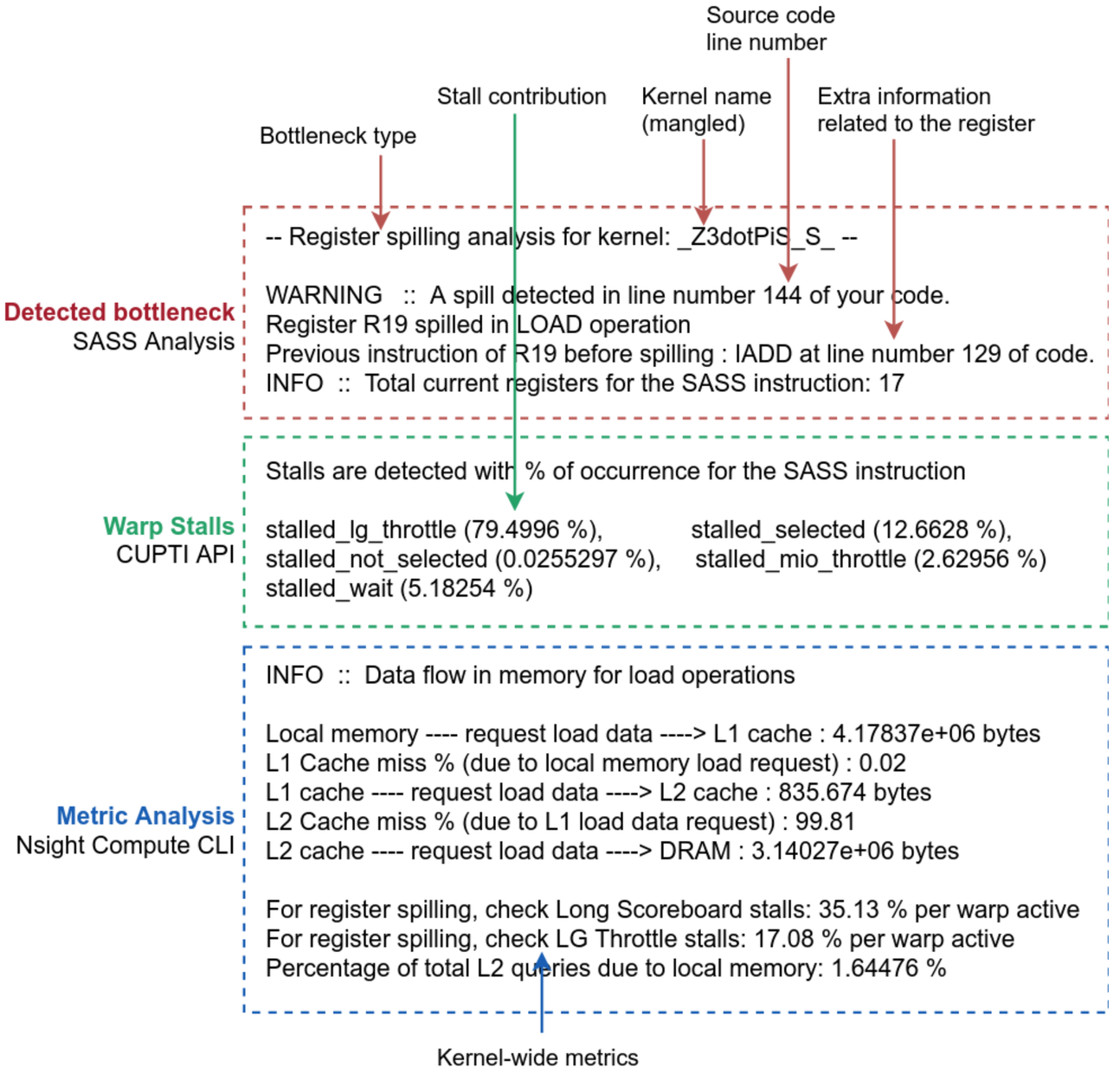
- **SASS analysis** at heart of GPUscout
  - Searching for specific code patterns
- **Warp stalls** for identified code line
- Kernel-wide **metrics** provide overview of data movements
- Additional metrics displayed for specific bottlenecks

Analyses

1. Vectorized Loads
2. Register Spilling
3. Shared Memory
4. Shared Atomics
5. Read-only Cache
6. Texture Memory
7. Datatype Conversions

…we want to add more!

GPUscout Analysis



Mixbench

Use-case 1

Mixbench

Use-case 1

- Benchmarking suite for mixed operational intensity kernels
- CUDA implementation mixbench-cuda
- Executes MAD operations

GPUscout analysis:

1. Use Shared Memory
2. Use Vectorized Loads

Mixbench

Use-case 1

- Benchmarking suite for mixed operational intensity kernels
- CUDA implementation mixbench-cuda
- Executes MAD operations

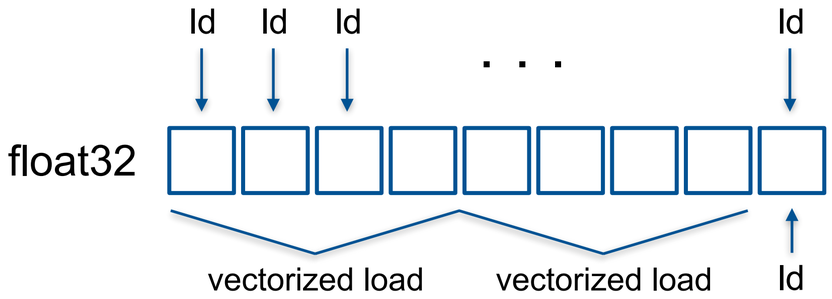
GPUscout analysis:

1. Use Shared Memory
2. Use Vectorized Loads

Mixbench

Use-case 1

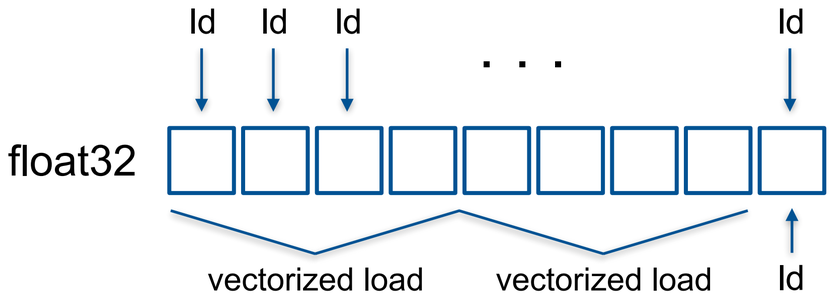
2. Use Vectorized Loads



Mixbench

Use-case 1

2. Use Vectorized Loads




```
for(int j=0; j<granularity; j++)
 tmps[j] = g_data[...];
 ...
 for(int i=0; i<compute_iterations; i++)
  tmps[j] = mad(tmps[j],tmps[j],seed);
```

```
for(int j=0; j<granularity/4; j++)
 reinterpret_cast<float4*>(tmps)[j] =
  reinterpret_cast<float4*>(g_data)[...];
 ...
 for(int i=0; i<compute_iterations; i++)
  reinterpret_cast<float4*>(tmps)[j] =
   mad(reinterpret_cast<float4*>(tmps)[j],
   reinterpret_cast<float4*>(tmps)[j],seed);
```

Mixbench

Use-case 1

2. Use Vectorized Loads

Warp stalls:

\+ Long scoreboard ↓ **62%** (originally 70%)

Metric Analysis:

– SM Occupancy ↓ **83%** (originally 92%)

➡ **Speedup of 3.77x**

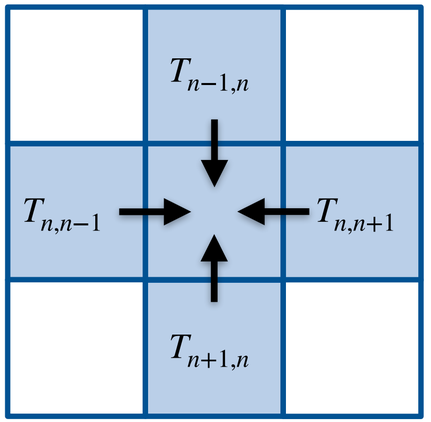
Heat Transfer Simulation

Use-case 2

Heat Transfer Simulation

Use-case 2

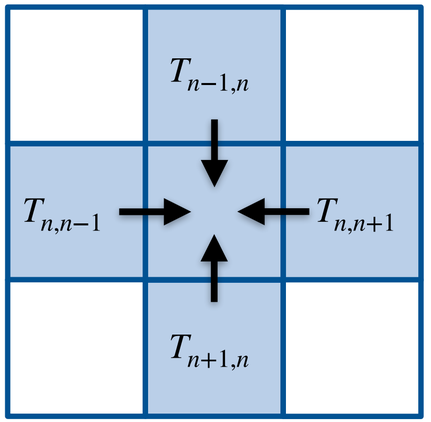
- Jacobi iterative solver for 2D
- $T_{NEW} = T_{OLD} + k * (T_{TOP} + T_{BOTTOM} + T_{LEFT} + T_{RIGHT} - 4 * T_{OLD})$
- GPUscout analysis:
  1. Use Texture Memory (or Use Shared memory)
  2. Use Vectorized Loads
  3. Using __restrict__ keyword
  4. Minimizing Datatype Conversions



Heat Transfer Simulation

Use-case 2

- Jacobi iterative solver for 2D
- $T_{NEW} = T_{OLD} + k * (T_{TOP} + T_{BOTTOM} + T_{LEFT} + T_{RIGHT} - 4 * T_{OLD})$
- GPUscout analysis:
  1. Use Texture Memory (or Use Shared memory)
  2. Use Vectorized Loads
  3. Using __restrict__ keyword
  4. Minimizing Datatype Conversions



Heat Transfer Simulation

Use-case 2

1. Use Texture Memory

```
== texture memory analysis for kernel: 2D-stencil-naive  ==
WARNING  ::  Use texture memory for register number (written-to): R4 at line
↪  number 6 of your code. The data is read from register number: R4
No spatial locality found for the register data
Stalls are detected with % of occurence for the SASS instruction
stalled_wait (66.6667 %), stalled_selected (33.3333 %)

WARNING  ::  Use texture memory for register number (written-to): R28 at line
↪  number 6 of your code. The data is read from register number: R2
Spatial locality found for the register data
Stalls are detected with % of occurence for the SASS instruction
stalled_wait (14.2857 %), stalled_lg_throttle (85.7143 %)

INFO  ::  Check data flow in texture memory, if you modify your code to use
↪  textures
Kernel ---- request load data ----> Texture Memory 0 instructions
Texture memory ---- request load data ----> L1 cache 0 bytes
L1 Cache miss % (due to texture memory load request) 100
L1 cache ---- request load data ----> L2 cache (due to texture memory load
↪  request) 0 bytes
L2 Cache miss % (due to L1 load data request) 23.89
L2 cache ---- request load data ----> DRAM 0 bytes
If using texture memory, check Tex Throttle: 0 %
If using texture memory, check Long Scoreboard: 37.79 %
```

Heat Transfer Simulation

Use-case 2

SASS Analysis

```
== texture memory analysis for kernel: 2D-stencil-naive  ==
WARNING  ::  Use texture memory for register number (written-to): R4 at line
↪  number 6 of your code. The data is read from register number: R4
No spatial locality found for the register data
Stalls are detected with % of occurence for the SASS instruction
stalled_wait (66.6667 %), stalled_selected (33.3333 %)

WARNING  ::  Use texture memory for register number (written-to): R28 at line
↪  number 6 of your code. The data is read from register number: R2
Spatial locality found for the register data
Stalls are detected with % of occurence for the SASS instruction
stalled_wait (14.2857 %), stalled_lg_throttle (85.7143 %)

INFO  ::  Check data flow in texture memory, if you modify your code to use
↪  textures
Kernel ---- request load data ----> Texture Memory 0 instructions
Texture memory ---- request load data ----> L1 cache 0 bytes
L1 Cache miss % (due to texture memory load request) 100
L1 cache ---- request load data ----> L2 cache (due to texture memory load
↪  request) 0 bytes
L2 Cache miss % (due to L1 load data request) 23.89
L2 cache ---- request load data ----> DRAM 0 bytes
If using texture memory, check Tex Throttle: 0 %
If using texture memory, check Long Scoreboard: 37.79 %
```

Heat Transfer Simulation

Use-case 2

SASS Analysis

```
== texture memory analysis for kernel: 2D-stencil-naive  ==
WARNING  ::  Use texture memory for register number (written-to): R4 at line
↪  number 6 of your code. The data is read from register number: R4
No spatial locality found for the register data
Stalls are detected with % of occurence for the SASS instruction
stalled_wait (66.6667 %), stalled_selected (33.3333 %)
```

Warp stalls

```
WARNING  ::  Use texture memory for register number (written-to): R28 at line
↪  number 6 of your code. The data is read from register number: R2
Spatial locality found for the register data
Stalls are detected with % of occurence for the SASS instruction
stalled_wait (14.2857 %), stalled_lg_throttle (85.7143 %)

INFO  ::  Check data flow in texture memory, if you modify your code to use
↪  textures
Kernel ---- request load data ----> Texture Memory 0 instructions
Texture memory ---- request load data ----> L1 cache 0 bytes
L1 Cache miss % (due to texture memory load request) 100
L1 cache ---- request load data ----> L2 cache (due to texture memory load
↪  request) 0 bytes
L2 Cache miss % (due to L1 load data request) 23.89
L2 cache ---- request load data ----> DRAM 0 bytes
If using texture memory, check Tex Throttle: 0 %
If using texture memory, check Long Scoreboard: 37.79 %
```

Heat Transfer Simulation

Use-case 2

SASS Analysis

```
== texture memory analysis for kernel: 2D-stencil-naive  ==
WARNING  ::  Use texture memory for register number (written-to): R4 at line
↪  number 6 of your code. The data is read from register number: R4
No spatial locality found for the register data
```

```
Stalls are detected with % of occurence for the SASS instruction
stalled_wait (66.6667 %), stalled_selected (33.3333 %)
```

Warp stalls

```
WARNING  ::  Use texture memory for register number (written-to): R28 at line
↪  number 6 of your code. The data is read from register number: R2
Spatial locality found for the register data
```

```
Stalls are detected with % of occurence for the SASS instruction
stalled_wait (14.2857 %), stalled_lg_throttle (85.7143 %)
```

Kernel Metrics

```
INFO  ::  Check data flow in texture memory, if you modify your code to use
↪  textures
Kernel ---- request load data ----> Texture Memory 0 instructions
Texture memory ---- request load data ----> L1 cache 0 bytes
L1 Cache miss % (due to texture memory load request) 100
L1 cache ---- request load data ----> L2 cache (due to texture memory load
↪  request) 0 bytes
L2 Cache miss % (due to L1 load data request) 23.89
L2 cache ---- request load data ----> DRAM 0 bytes
If using texture memory, check Tex Throttle: 0 %
If using texture memory, check Long Scoreboard: 37.79 %
```

Heat Transfer Simulation

Use-case 2

```
== texture memory analysis for kernel: 2D-stencil-naive  ==
WARNING  ::  Use texture memory for register number (written-to): R4 at line
↪  number 6 of your code. The data is read from register number: R4
No spatial locality found for the register data
Stalls are detected with % of occurence for the SASS instruction
stalled_wait (66.6667 %), stalled_selected (33.3333 %)

WARNING  ::  Use texture memory for register number (written-to): R28 at line
↪  number 6 of your code. The data is read from register number: R2
Spatial locality found for the register data
Stalls are detected with % of occurence for the SASS instruction
stalled_wait (14.2857 %), stalled_lg_throttle (85.7143 %)
```

Heat Transfer Simulation

Use-case 2

```
== texture memory analysis for kernel: 2D-stencil-naive  ==
WARNING  ::  Use texture memory for register number (written-to): R4 at line
↪  number 6 of your code. The data is read from register number: R4
No spatial locality found for the register data
Stalls are detected with % of occurence for the SASS instruction
stalled_wait (66.6667 %), stalled_selected (33.3333 %)

WARNING  ::  Use texture memory for register number (written-to): R28 at line
↪  number 6 of your code. The data is read from register number: R2
Spatial locality found for the register data
Stalls are detected with % of occurence for the SASS instruction
stalled_wait (14.2857 %), stalled_lg_throttle (85.7143 %)
```

Heat Transfer Simulation

Use-case 2

```
== texture memory analysis for kernel: 2D-stencil-naive  ==
WARNING  ::  Use texture memory for register number (written-to): R4 at line
↪  number 6 of your code. The data is read from register number: R4
No spatial locality found for the register data
Stalls are detected with % of occurence for the SASS instruction
stalled_wait (66.6667 %), stalled_selected (33.3333 %)

WARNING  ::  Use texture memory for register number (written-to): R28 at line
↪  number 6 of your code. The data is read from register number: R2
Spatial locality found for the register data
Stalls are detected with % of occurence for the SASS instruction
stalled_wait (14.2857 %), stalled_lg_throttle (85.7143 %)
```

stalled_wait: Warp stalled waiting for a execution dependency of a fixed-latency instruction. Caused mostly because of an already highly optimized kernel.

stalled_lg_throttle: Warp stalled waiting for the L1 instruction queue for local and global (LG) memory operations. Caused mostly because of executing local or global memory instructions too frequently.

Heat Transfer Simulation

Use-case 2

```
INFO  ::  Check data flow in texture memory, if you modify your code to use
↪  textures
Kernel ---- request load data ----> Texture Memory 0 instructions
Texture memory ---- request load data ----> L1 cache 0 bytes
L1 Cache miss % (due to texture memory load request) 100
L1 cache ---- request load data ----> L2 cache (due to texture memory load
↪  request) 0 bytes
L2 Cache miss % (due to L1 load data request) 23.89
L2 cache ---- request load data ----> DRAM 0 bytes
If using texture memory, check Tex Throttle: 0 %
If using texture memory, check Long Scoreboard: 37.79 %
```

Heat Transfer Simulation

Use-case 2


```
INFO  ::  Check data flow in texture memory, if you modify your code to use
↪  textures
Kernel ---- request load data ----> Texture Memory 0 instructions
Texture memory ---- request load data ----> L1 cache 0 bytes
L1 Cache miss % (due to texture memory load request) 100
L1 cache ---- request load data ----> L2 cache (due to texture memory load
↪  request) 0 bytes
L2 Cache miss % (due to L1 load data request) 23.89
L2 cache ---- request load data ----> DRAM 0 bytes
If using texture memory, check Tex Throttle: 0 %
If using texture memory, check Long Scoreboard: 37.79 %
```

Heat Transfer Simulation

Use-case 2

- Jacobi iterative solver for 2D
- $T_{NEW} = T_{OLD} + k * (T_{TOP} + T_{BOTTOM} + T_{LEFT} + T_{RIGHT} - 4 * T_{OLD})$
- GPUscout analysis:

  1. Use Texture Memory

Heat Transfer Simulation

Use-case 2

- Jacobi iterative solver for 2D
- $T_{NEW} = T_{OLD} + k * (T_{TOP} + T_{BOTTOM} + T_{LEFT} + T_{RIGHT} - 4 * T_{OLD})$
- GPUscout analysis:

1. Use Texture Memory
   - – TEX throttle ↑ **25%** (originally 0%)
   - \+ Long Scoreboard ↓ **27%** (originally 38%)
   - \+ Throughput ↑ **61%**

➡ **Performance improvement of 39.2%**

Heat Transfer Simulation

Use-case 2

- Jacobi iterative solver for 2D
- $T_{NEW} = T_{OLD} + k * (T_{TOP} + T_{BOTTOM} + T_{LEFT} + T_{RIGHT} - 4 * T_{OLD})$
- GPUscout analysis:

3. Using __restrict__ keyword

Heat Transfer Simulation

Use-case 2

- Jacobi iterative solver for 2D
- $T_{NEW} = T_{OLD} + k * (T_{TOP} + T_{BOTTOM} + T_{LEFT} + T_{RIGHT} - 4 * T_{OLD})$
- GPUscout analysis:

3. Using __restrict__ keyword

➡ Performance improvement of only **0.3%**

Heat Transfer Simulation

Use-case 2

- Jacobi iterative solver for 2D
- $T_{NEW} = T_{OLD} + k * (T_{TOP} + T_{BOTTOM} + T_{LEFT} + T_{RIGHT} - 4 * T_{OLD})$
- GPUscout analysis:

4. Minimizing Datatype Conversions

Heat Transfer Simulation

Use-case 2

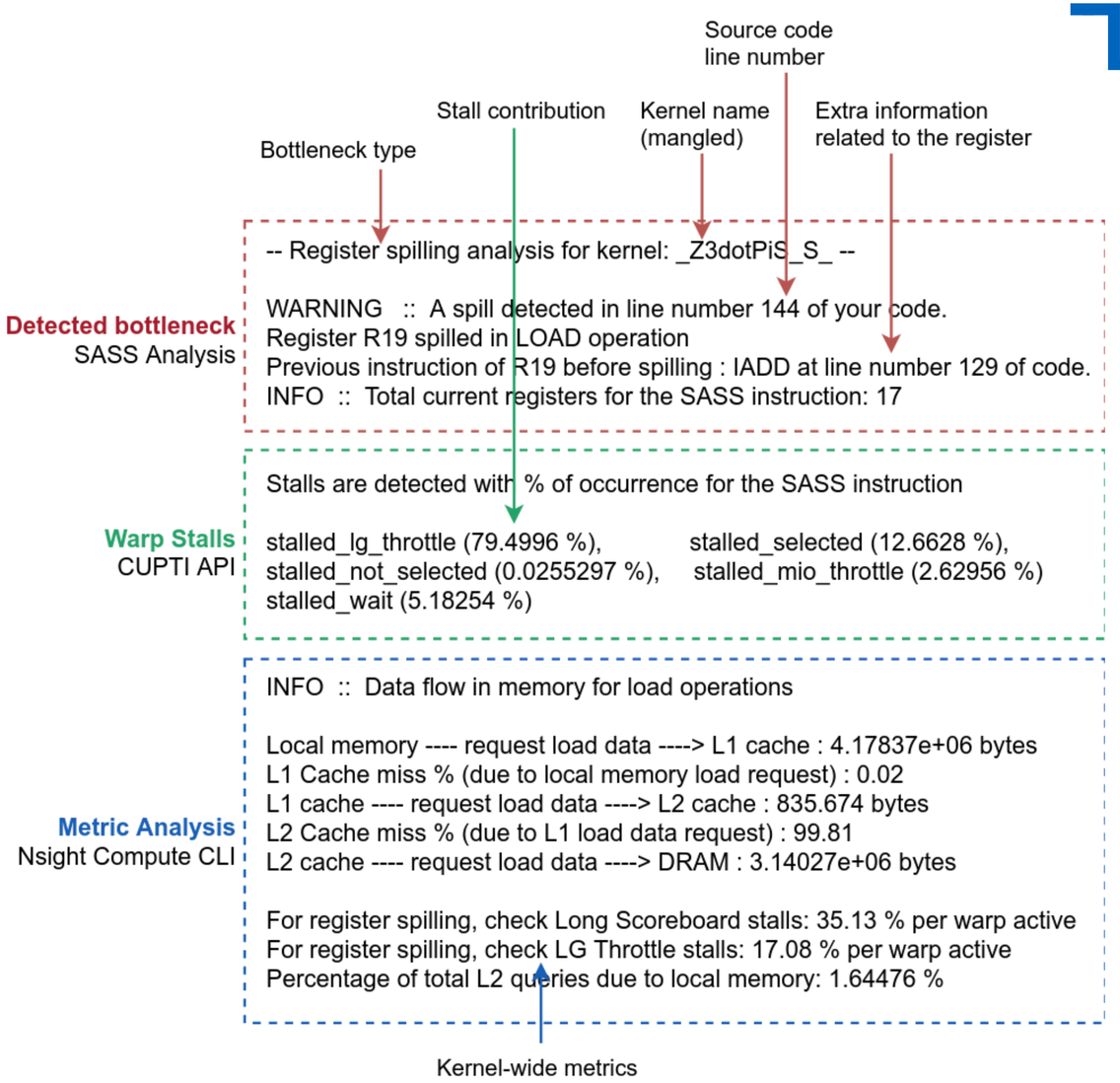
- Jacobi iterative solver for 2D
- $T_{NEW} = T_{OLD} + k * (T_{TOP} + T_{BOTTOM} + T_{LEFT} + T_{RIGHT} - 4 * T_{OLD})$
- GPUscout analysis:

4. Minimizing Datatype Conversions
    - Impossible to avoid

Next Steps

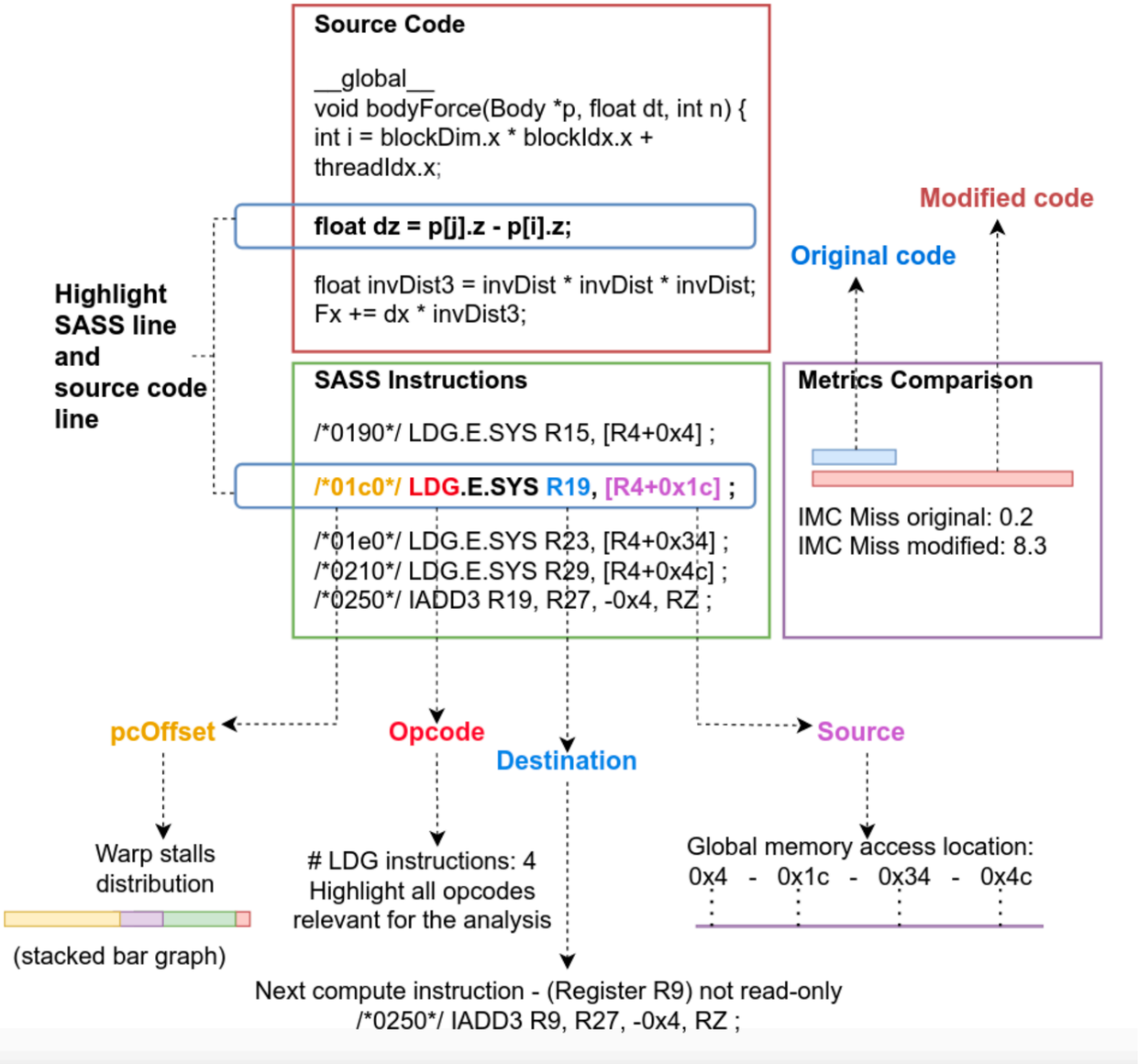
Next Steps

- **Interactive visual front-end** (work just started)



Next Steps

- **Interactive visual front-end** (work just started)



Next Steps

- **Interactive visual front-end** (work just started)
- **Add GPU memory topology information**
  - We have data available through the sys-sage library
- **Extend to AMD GPUs** (work just started)
  - *Is the SW and HW support on AMD side sufficient?*
- Add more analyses/bottlenecks
  - Coalescing, ???
- Smaller tweaks:
  - Option to analyze only a specific kernel, specific bottlenecks

GPUscout

- A tool that focuses on detecting memory-based bottlenecks on NVidia GPU kernels
- Combines static code analysis, sampling PC Stalls, and providing kernel-wide metrics
- Attributes the observed pattern directly to the related source code line
- GPUscout recommendations bring a speedup of 3.77x and 1.64x on presented kernels

Try out GPUscout and get in touch with us!

https://github.com/caps-tum/GPUscout

spack install gpuscout

stepan.vanecek@tum.de