CuSan: A Host-Device CUDA Data Race Sanitizer

https://github.com/tudasc/cusan

Alexander Hück*, **Tim Ziegler***, **Simon Schwitanski**+, **Joachim Jenke**+, **Christian Bischof***
*Scientific Computing, Technical University of Darmstadt and +IT Center, RWTH Aachen University
Contact: alexander.hueck@tu-darmstadt.de

CuSan detects host-device data races of C/C++ CUDA codes

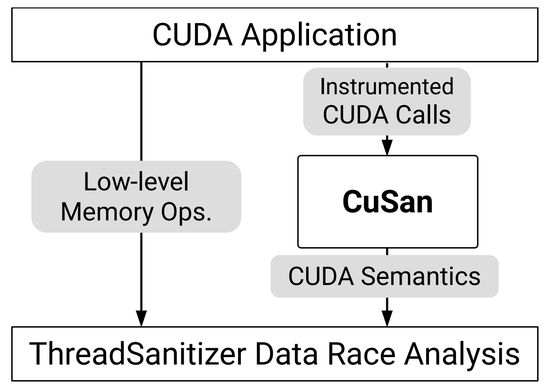
```
13 __global__ void write_kernel(int* arr, const int N)
...
46   write_kernel<<<..., stream1>>>(data, size);
47   // Missing synchronization
48   write_kernel<<<..., stream2>>>(data, size);
```

```
WARNING: ThreadSanitizer: data race (pid=2770877)
  Write of size 8 at 0x7fba99200000 by thread T5:
    #0 __device_stub__write_kernel(int*, int) cuda_nok.c:13:94
    #1 main cuda_nok.c:48:3
  Previous write of size 8 at 0x7fba99200000 by thread T4:
    #0 __device_stub__write_kernel(int*, int) cuda_nok.c:13:94
    #1 main cuda_nok.c:46:3
  Thread T5 'cuda_stream 2' (tid=0, running) created by main thread at:
    #0 cusan::runtime::Runtime::register_stream(cusan::runtime::Stream) <null>
  Thread T4 'cuda_stream 1' (tid=0, running) created by main thread at:
    #0 cusan::runtime::Runtime::register_stream(cusan::runtime::Stream) <null>
SUMMARY: ThreadSanitizer: data race cuda_nok.c:13:94 in
__device_stub__write_kernel(int*, int)
```

Overview

Detects data races between (asynchronous) CUDA and the host using ThreadSanitizer

Usage: Use compiler wrapper (e.g., `cusan-clang++`), then execute for detection



1. Basis: Use ThreadSanitizer (TSan)

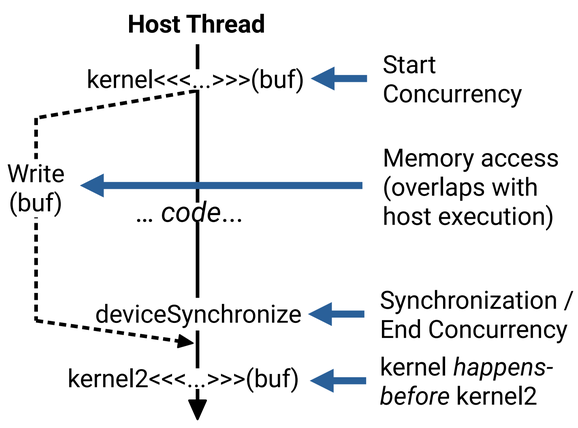
Thread-level data race detection using instrumentation and runtime library

CUDA is a black-box for TSan. Need to manually call TSan API to expose CUDA semantics:

1) Concurrency
- CUDA streams, run concurrently on GPU
  ⇒ Create *Fiber* (think virtual thread)

2) Synchronization
- Need to establish *happens-before* relationship (device, stream, event sync.)
  ⇒ Call AnnotateHappens*Before*/*After*

3) Memory Accesses
- Host-side managed memory access already instrumented
- Does kernel r/w memory?
  ⇒ Call memory annotations from Fiber

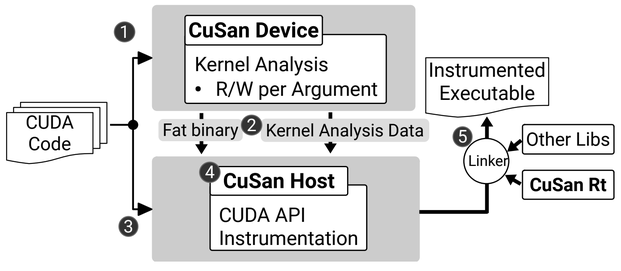


2. CuSan: Exposing CUDA (Host) Semantics to TSan

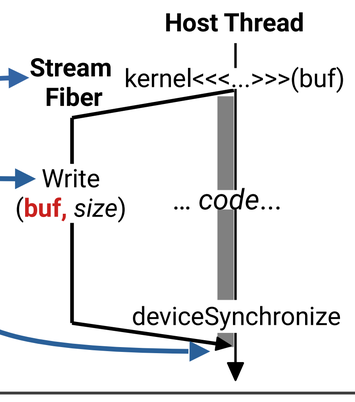
CuSan is a LLVM compiler plugin and runtime that works in conjunction with TSan

Compile time (LLVM IR)
- Analyzes kernel memory accesses (device) and instruments CUDA API (host)

Runtime
- Based on instrumentation: Calls TSan API w.r.t. synchronization and concurrency state of CUDA
- Tracks CUDA memory allocations for TSan r/w memory annotations for each kernel invocation



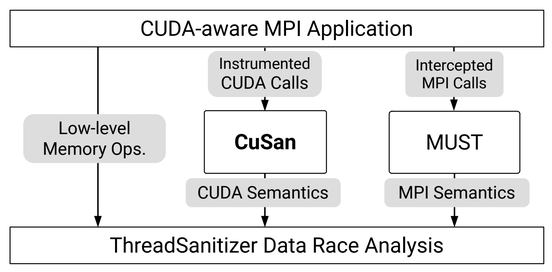

```
1 cudaMalloc(&buf, size);
2 _cusan_device_alloc(buf, size)
3 _cusan_kernel_register({buf}, {w}, stream)
4 kernel<<<...>>>(buf, ...);
5 ... code ...
6 cudaDeviceSynchronize();
7 _cusan_sync_device()
```



Concurrent Region of CUDA Kernel

3. Use Case: CUDA-Aware MPI [1]

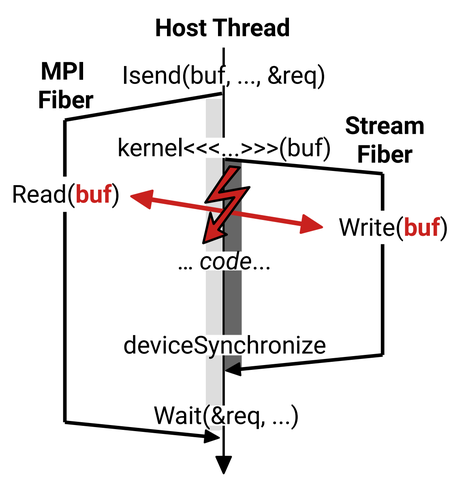
CUDA-aware MPI can **transfer device memory directly**. CUDA and MPI operations must be synchronized:
- MPI correctness checker MUST uses TSan to detect data races w.r.t. MPI
- MUST+CuSan enable data race analysis for combined semantics of CUDA and MPI



Example: If kernel writes to buffer, a data race happens
- Two conflicting memory accesses to **buf**

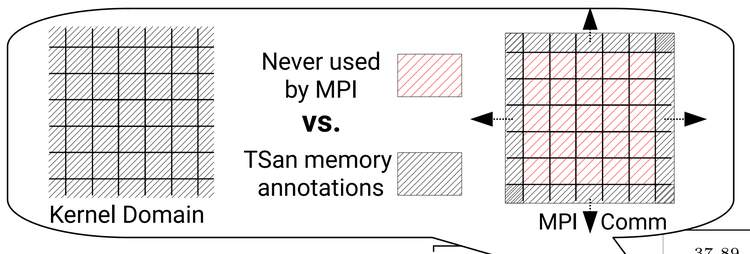
```
1 MPI_Isend(buf, ..., &req);
2   kernel<<<...>>>(buf, ...);
3   ... code ...
4   cudaDeviceSynchronize();
5 MPI_Wait(&req, ...);
```

Concurrent Region of MPI
Concurrent Region of CUDA Kernel



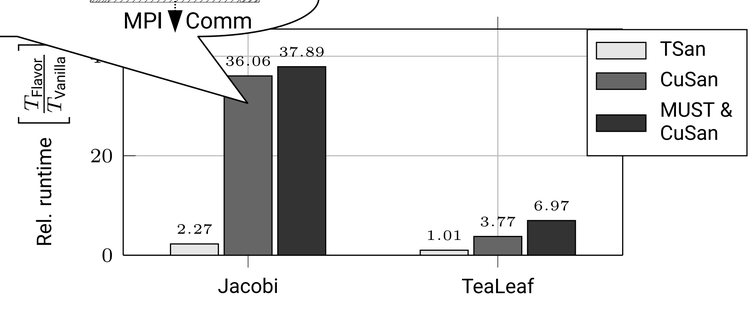
4. Performance Evaluation: CUDA-Aware MPI

Test Suite



Runtime overhead
- Depends on amount of memory tracked in TSan
- Documentation TSan: 5-15×

Setup: Two compute nodes, one MPI process each, one NVIDIA Tesla V100 each. Clang 14, OpenMPI 4.1.6 and CUDA 11.8.



CuSan marks the whole kernel buffer as r/w with TSan ↦ *expensive*
- Only buffer region that is shared between kernel(s) and user-code should be tracked

CuSan
https://github.com/tudasc/cusan
License BSD 3-Clause

MUST
https://itc.rwth-aachen.de/must
License BSD 3-Clause

[1] A. Hück, et. al., **"Compiler-Aided Correctness Checking of CUDA-Aware MPI Applications"**, DOI: 10.1109/SCW63240.2024.00032

IT Center | RWTH AACHEN UNIVERSITY

Scientific Computing