
# COMPUTE-SANITIZER
# 41TH VI-HPS TUNING WORKSHOP

NICOLAS POITOUX, SYSTEM SOFTWARE ENGINEER

# COMPUTE SANITIZER

What is compute-sanitizer

- Dynamic analysis tools that reports common programming errors that could lead to undefined behavior
- Command line, non-interactive
- Included in CUDA toolkit since 11.0
  - /usr/local/cuda/bin/compute-sanitizer
  - C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\V11.X\bin\compute-sanitizer.bat
- 4 tools
  - Memcheck: report invalid memory access
  - Initcheck: reports uninitialized memory reads
  - Racecheck: reports invalid concurrent accesses to shared memory
  - Synccheck: reports invalid barrier usage
- Replaces cuda-memcheck (deprecated since CUDA 11.5)
- Supports all regular CUDA applications, and OptiX
- Supports architectures >= Maxwell on Windows, Linux (x86_64, ppc64le, aarch64, aarch64sbsa)

# COMPUTE SANITIZER

## Memcheck

- Memcheck: report invalid memory accesses
  - Out of bounds or misaligned, local, shared, global memory read/writes or atomic accesses
  - Stack overflows
  - Invalid system-scoped atomic accesses
- Reports CUDA API errors
- Hardware exceptions
- Invalid device-side malloc/free usage

# COMPUTE SANITIZER

## Memcheck

```
__device__ void writeIdx(int* buffer)
{
    buffer[threadIdx.x] = threadIdx.x;
}

__global__ void kernel(int* buffer)
{
    writeIdx(buffer);
}

int main()
{
    void* devBuf = nullptr;
    cudaMalloc(&devBuf, 31 * sizeof(int));
    kernel<<<1,32>>>(static_cast<int*>(devBuf));
    return cudaDeviceSynchronize();
}
```

- Uses debug info -g -G
- Destroys context by default,

`--destroy-on-device-error [context|kernel]`

```
$ compute-sanitizer ./memcheck-test
========= COMPUTE-SANITIZER
========= Invalid __global__ write of size 4 bytes
=========     at 0x1b0 in /home/npoitoux/vihps/memcheck.cu:3:writeIdx(int *)
=========     by thread (31,0,0) in block (0,0,0)
=========     Address 0x7f746da0007c is out of bounds
=========     and is 1 bytes after the nearest allocation at 0x7f746da00000 of size 124 bytes
=========     Device Frame:/home/npoitoux/vihps/memcheck.cu:8:kernel(int *) [0xd0]
=========     Saved host backtrace up to driver entry point at kernel launch time
=========     Host Frame: [0x21740c]
=========                 in /usr/lib/x86_64-linux-gnu/libcuda.so.1
=========     Host Frame:__cudart803 [0xeabb]
=========                 in /home/npoitoux/vihps/./memcheck-test
=========     Host Frame:cudaLaunchKernel [0x69918]
=========                 in /home/npoitoux/vihps/./memcheck-test
=========     Host Frame:/usr/local/cuda/bin/../targets/x86_64-linux/include/cuda_runtime.h:211:cudaError cudaLaunchKernel<char
>(char const*, dim3, dim3, void**, unsigned long, CUstream_st*) [0xa392]
=========                 in /home/npoitoux/vihps/./memcheck-test
=========     Host Frame:/tmp/tmpxft_00040002_00000000-6_memcheck.cudafe1.stub.c:13:__device_stub__Z6kernelPi(int*) [0xa242]
=========                 in /home/npoitoux/vihps/./memcheck-test
=========     Host Frame:/home/npoitoux/vihps/memcheck.cu:9:kernel(int*) [0xa286]
=========                 in /home/npoitoux/vihps/./memcheck-test
=========     Host Frame:/home/npoitoux/vihps/memcheck.cu:16:main [0xa0b0]
=========                 in /home/npoitoux/vihps/./memcheck-test
=========     Host Frame:__libc_start_main [0x270b3]
=========                 in /lib/x86_64-linux-gnu/libc.so.6
=========     Host Frame:_start [0x9efe]
=========                 in /home/npoitoux/vihps/./memcheck-test
=========
========= Program hit cudaErrorLaunchFailure (error 719) due to "unspecified launch failure" on CUDA API call to cudaDeviceSync
hronize.
=========     Saved host backtrace up to driver entry point at error
=========     Host Frame: [0x34fb13]
=========                 in /usr/lib/x86_64-linux-gnu/libcuda.so.1
=========     Host Frame:cudaDeviceSynchronize [0x44597]
=========                 in /home/npoitoux/vihps/./memcheck-test
=========     Host Frame:/home/npoitoux/vihps/memcheck.cu:17:main [0xa0b5]
=========                 in /home/npoitoux/vihps/./memcheck-test
=========     Host Frame:__libc_start_main [0x270b3]
=========                 in /lib/x86_64-linux-gnu/libc.so.6
=========     Host Frame:_start [0x9efe]
=========                 in /home/npoitoux/vihps/./memcheck-test
=========
========= Target application returned an error
========= ERROR SUMMARY: 2 errors
```

# COMPUTE SANITIZER

## Memcheck

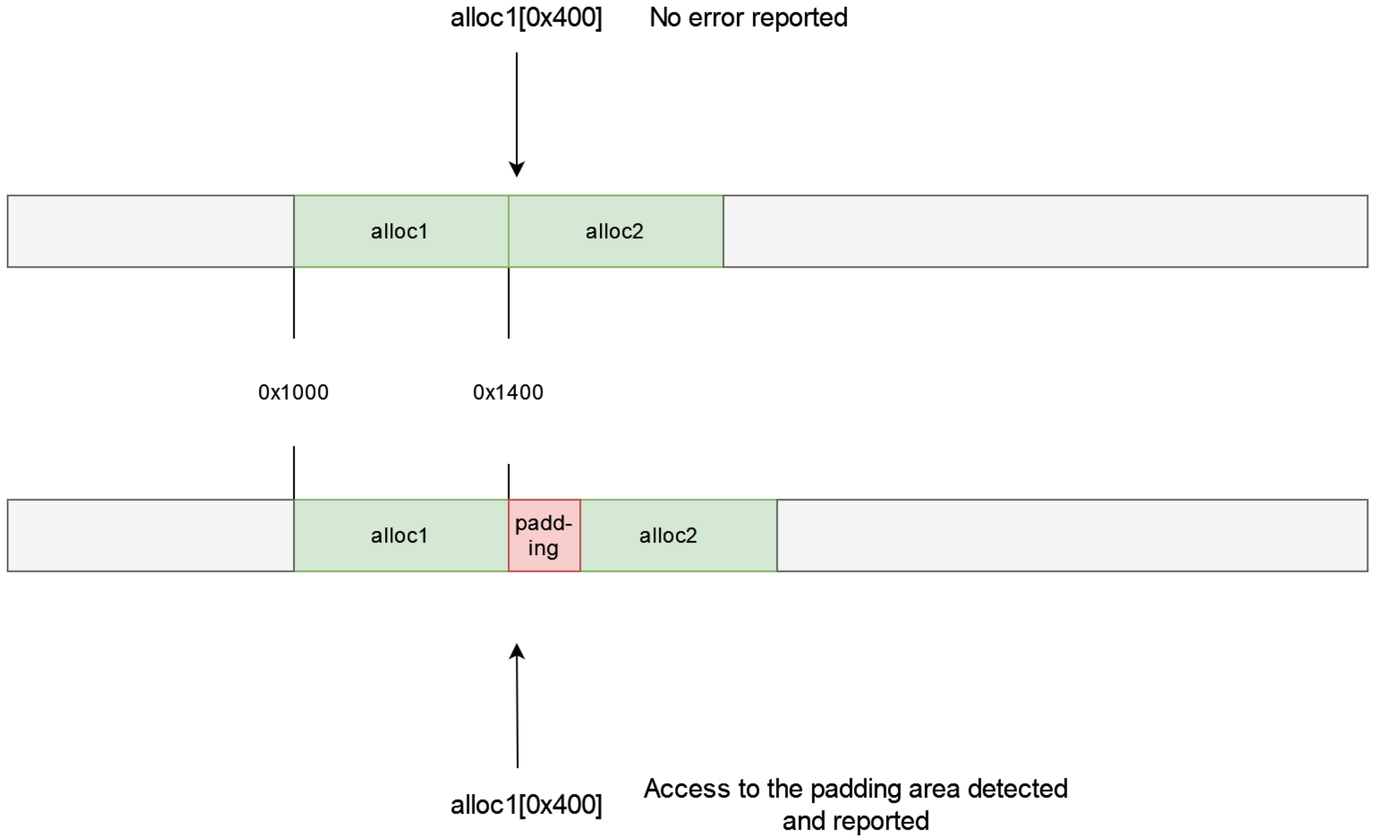
- Reports device memory leaks with `--leak-check=full`

```
$ compute-sanitizer --leak-check=full ./memcheck-test
========= COMPUTE-SANITIZER
========= Leaked 128 bytes at 0x7fd0d5a00000
=========     Saved host backtrace up to driver entry point at cudaMalloc time
=========     Host Frame: [0x251b87]
=========                 in /usr/lib/x86_64-linux-gnu/libcuda.so.1
=========     Host Frame:__cudart612 [0x412fe]
=========                 in /home/npoitoux/vihps/./memcheck-test
=========     Host Frame:__cudart618 [0xf5fb]
=========                 in /home/npoitoux/vihps/./memcheck-test
=========     Host Frame:cudaMalloc [0x4e78f]
=========                 in /home/npoitoux/vihps/./memcheck-test
=========     Host Frame:/home/npoitoux/vihps/memcheck.cu:15:main [0xa042]
=========                 in /home/npoitoux/vihps/./memcheck-test
=========     Host Frame:__libc_start_main [0x270b3]
=========                 in /lib/x86_64-linux-gnu/libc.so.6
=========     Host Frame:_start [0x9efe]
=========                 in /home/npoitoux/vihps/./memcheck-test
=========
========= LEAK SUMMARY: 128 bytes leaked in 1 allocations
========= ERROR SUMMARY: 1 error
```

# COMPUTE SANITIZER

## Memcheck

- Can pad memory to mitigate false-negative on aligned consecutive allocations
- `--padding=n`



# COMPUTE SANITIZER

## Memcheck

```
$ compute-sanitizer ./padding
========= COMPUTE-SANITIZER
========= ERROR SUMMARY: 0 errors
$ compute-sanitizer --padding=1 ./padding
========= COMPUTE-SANITIZER
========= Invalid __global__ write of size 1 bytes
=========     at 0x160 in /home/npoitoux/vihps/padding.cu:3:kernel(char *)
=========     by thread (0,0,0) in block (0,0,0)
=========     Address 0x7f42cda00200 is out of bounds
=========     and is 1 bytes after the nearest allocation at 0x7f42cda00000 of size 512 bytes
=========     Saved host backtrace up to driver entry point at kernel launch time
=========     Host Frame: [0x21740c]
=========                 in /usr/lib/x86_64-linux-gnu/libcuda.so.1
=========     Host Frame:__cudart803 [0xeaab]
=========                 in /home/npoitoux/vihps/./padding
=========     Host Frame:cudaLaunchKernel [0x69908]
=========                 in /home/npoitoux/vihps/./padding
=========     Host Frame:/usr/local/cuda/bin/../targets/x86_64-linux/include/cuda_runtime.h:211:cudaError cudaLaunchKernel<char
>(char const*, dim3, dim3, void**, unsigned long, CUstream_st*) [0xa37a]
=========                 in /home/npoitoux/vihps/./padding
=========     Host Frame:/tmp/tmpxft_0003b58b_00000000-6_padding.cudafe1.stub.c:13:__device_stub__Z6kernelPc(char*) [0xa22a]
=========                 in /home/npoitoux/vihps/./padding
=========     Host Frame:/home/npoitoux/vihps/padding.cu:4:kernel(char*) [0xa26e]
=========                 in /home/npoitoux/vihps/./padding
=========     Host Frame:/home/npoitoux/vihps/padding.cu:13:main [0xa098]
=========                 in /home/npoitoux/vihps/./padding
=========     Host Frame:__libc_start_main [0x270b3]
=========                 in /lib/x86_64-linux-gnu/libc.so.6
=========     Host Frame:_start [0x9eee]
=========                 in /home/npoitoux/vihps/./padding
```

```cpp
__global__ void kernel(char* buffer)
{
    buffer[512] = 93;
}

int main()
{
    void* devBuf = nullptr;
    cudaMalloc(&devBuf, 512);
    void* devBuf2 = nullptr;
    cudaMalloc(&devBuf2, 512);
    kernel<<<1,1>>>(static_cast<char*>(devBuf));
    return cudaDeviceSynchronize();
}
```

# COMPUTE SANITIZER

## Initcheck

- Detect and report uninitialized memory reads
  - Inside a kernel or from the CUDA API
- Only for global memory
- Works with peer GPUs allocations

# COMPUTE SANITIZER

## Initcheck

```
$ compute-sanitizer --tool=initcheck ./initcheck-test
========= COMPUTE-SANITIZER
========= Uninitialized __global__ memory read of size 4 bytes
=========     at 0x1a0 in /home/npoitoux/vihps/initcheck.cu:3:kernel(int *)
=========     by thread (0,0,0) in block (0,0,0)
=========     Address 0x7f84c5800000
=========     Saved host backtrace up to driver entry point at kernel launch time
=========     Host Frame: [0x21740c]
=========                 in /usr/lib/x86_64-linux-gnu/libcuda.so.1
=========     Host Frame:__cudart803 [0xea8b]
=========                 in /home/npoitoux/vihps/./initcheck-test
=========     Host Frame:cudaLaunchKernel [0x698e8]
=========                 in /home/npoitoux/vihps/./initcheck-test
=========     Host Frame:/usr/local/cuda/bin/../targets/x86_64-linux/include/cuda_runtime.h:211:cudaError cudaLaunchKernel<char
>(char const*, dim3, dim3, void**, unsigned long, CUstream_st*) [0xa361]
=========                 in /home/npoitoux/vihps/./initcheck-test
=========     Host Frame:/tmp/tmpxft_00040107_00000000-6_initcheck.cudafe1.stub.c:13:__device_stub__Z6kernelPi(int*) [0xa211]
=========                 in /home/npoitoux/vihps/./initcheck-test
=========     Host Frame:/home/npoitoux/vihps/initcheck.cu:4:kernel(int*) [0xa255]
=========                 in /home/npoitoux/vihps/./initcheck-test
=========     Host Frame:/home/npoitoux/vihps/initcheck.cu:11:main [0xa07f]
=========                 in /home/npoitoux/vihps/./initcheck-test
=========     Host Frame:__libc_start_main [0x270b3]
=========                 in /lib/x86_64-linux-gnu/libc.so.6
=========     Host Frame:_start [0x9eee]
=========                 in /home/npoitoux/vihps/./initcheck-test
=========
========= ERROR SUMMARY: 1 error
```

```cpp
__global__ void kernel(int* buffer)
{
    buffer[threadIdx.x] = buffer[threadIdx.x] + threadIdx.x;
}

int main()
{
    void* devBuf = nullptr;
    cudaMalloc(&devBuf, 32 * sizeof(int));
    kernel<<<1,1>>>(static_cast<int*>(devBuf));
    return cudaDeviceSynchronize();
}
```

# COMPUTE SANITIZER

## Initcheck

- Detect unused (not written to) memory with `--track-unused-memory yes`

```
$ compute-sanitizer --tool=initcheck --track-unused-memory=yes ./initcheck-test
========= COMPUTE-SANITIZER
=========  Unused memory in allocation 0x7ff24b800000 of size 128
=========     Not written 124 bytes at 4 (0x7ff24b800004)
=========     96.875% of allocation were unused.
=========     Saved host backtrace up to driver entry point at cudaMalloc time
=========     Host Frame: [0x251b87]
=========                 in /usr/lib/x86_64-linux-gnu/libcuda.so.1
=========     Host Frame:__cudart612 [0x412ce]
=========                 in /home/npoitoux/vihps/./initcheck-test
=========     Host Frame:__cudart618 [0xf5cb]
=========                 in /home/npoitoux/vihps/./initcheck-test
=========     Host Frame:cudaMalloc [0x4e75f]
=========                 in /home/npoitoux/vihps/./initcheck-test
=========     Host Frame:/home/npoitoux/vihps/initcheck.cu:10:main [0xa011]
=========                 in /home/npoitoux/vihps/./initcheck-test
=========     Host Frame:__libc_start_main [0x270b3]
=========                 in /lib/x86_64-linux-gnu/libc.so.6
=========     Host Frame:_start [0x9eee]
=========                 in /home/npoitoux/vihps/./initcheck-test
=========
========= ERROR SUMMARY: 1 error
```

# COMPUTE SANITIZER

## Racecheck

- Detects possible data races
  - WAW, WAR, RAW accesses to a shared memory address by different threads with lack of valid synchronization (warp level, block level…)
- Only on shared memory
- Two reporting options
  - Analysis: Aggregated reports
  - Hazard: Every single detected error with details
- Multiple severity levels
  - Info: not displayed by default. e.g. same value
  - Warn: may be valid on architectures older than Volta
- Can support partial masks
- Works with cuda::barrier from libcu++

# COMPUTE SANITIZER

## Racecheck

```
$ compute-sanitizer --tool=racecheck ./racecheck-test
========= COMPUTE-SANITIZER
========= Error: Race reported between Write access at 0x250 in /home/npoitoux/vihps/racecheck.cu:5:kernel(int *)
=========     and Read access at 0x5d0 in /home/npoitoux/vihps/racecheck.cu:6:kernel(int *) [256 hazards]
=========
========= RACECHECK SUMMARY: 1 hazard displayed (1 error, 0 warnings)
```

```
$ compute-sanitizer --tool=racecheck --racecheck-report=hazard ./racecheck-test | head -n 25
========= COMPUTE-SANITIZER
========= Warning: (Warp Level Programming) Potential RAW hazard detected at __shared__ 0x84 in block (0,0,0) :
=========     Write Thread (33,0,0) at 0x250 in /home/npoitoux/vihps/racecheck.cu:5:kernel(int *)
=========     Read Thread (32,0,0) at 0x5d0 in /home/npoitoux/vihps/racecheck.cu:6:kernel(int *)
=========     Current Value : 33
=========     Saved host backtrace up to driver entry point at kernel launch time
=========     Host Frame: [0x21740c]
=========                 in /usr/lib/x86_64-linux-gnu/libcuda.so.1
=========     Host Frame:__cudart803 [0xea8b]
=========                 in /home/npoitoux/vihps/./racecheck-test
=========     Host Frame:cudaLaunchKernel [0x698e8]
=========                 in /home/npoitoux/vihps/./racecheck-test
=========     Host Frame:/usr/local/cuda/bin/../targets/x86_64-linux/include/cuda_runtime.h:211:cudaError cudaLaunchKernel<char
>(char const*, dim3, dim3, void**, unsigned long, CUstream_st*) [0xa361]
=========                 in /home/npoitoux/vihps/./racecheck-test
=========     Host Frame:/tmp/tmpxft_00040259_00000000-6_racecheck.cudafe1.stub.c:13:__device_stub__Z6kernelPi(int*) [0xa211]
=========                 in /home/npoitoux/vihps/./racecheck-test
=========     Host Frame:/home/npoitoux/vihps/racecheck.cu:7:kernel(int*) [0xa255]
=========                 in /home/npoitoux/vihps/./racecheck-test
=========     Host Frame:/home/npoitoux/vihps/racecheck.cu:14:main [0xa07f]
=========                 in /home/npoitoux/vihps/./racecheck-test
=========     Host Frame:__libc_start_main [0x270b3]
=========                 in /lib/x86_64-linux-gnu/libc.so.6
=========     Host Frame:_start [0x9eee]
=========                 in /home/npoitoux/vihps/./racecheck-test
```

```cpp
__global__ void kernel(int* buffer)
{
    __shared__ int shared[64];

    shared[threadIdx.x] = threadIdx.x;
    buffer[threadIdx.x] = shared[(threadIdx.x + 1) % 64];
}

int main()
{
    void* devBuf = nullptr;
    cudaMalloc(&devBuf, 64 * sizeof(int));
    kernel<<<1,64>>>(static_cast<int*>(devBuf));
    return cudaDeviceSynchronize();
}
```

# COMPUTE SANITIZER

## Synccheck

- Detects invalid usage of CUDA synchronization primitives
- Dependent on the architecture
  - Divergent threads in warp/block, invalid barrier arguments
- Also verifies correctness of cuda::barrier

# COMPUTE SANITIZER

## Synccheck

```cpp
#include <cuda/barrier>

__global__ void kernel()
{
    __shared__ cuda::barrier<cuda::thread_scope_block> barrier;

    if (threadIdx.x == 0)
    {
        init(&barrier, blockDim.x / 2);
    }

    __syncthreads();

    auto token = barrier.arrive();
    barrier.wait(std::move(token));
}

int main()
{
    kernel<<<1,32>>>();
    return cudaDeviceSynchronize();
}
```

# COMPUTE SANITIZER

## Synccheck

```
$ compute-sanitizer --tool=synccheck ./synccheck-test
========= COMPUTE-SANITIZER
========= Barrier error detected. Barrier overflow
=========     at 0x540 in /usr/local/cuda/targets/x86_64-linux/include/cuda/std/barrier:189:cuda::__4::barrier<(cuda::std::__4:
:__detail::thread_scope)2, cuda::std::__4::__empty_completion>::arrive(long)
=========     by thread (31,0,0) in block (0,0,0)
=========     Device Frame:/home/npoitoux/vihps/synccheck.cu:14:kernel() [0x6f0]
=========     Saved host backtrace up to driver entry point at kernel launch time
=========     Host Frame: [0x21740c]
=========                 in /usr/lib/x86_64-linux-gnu/libcuda.so.1
=========     Host Frame:__cudart803 [0xea4b]
=========                 in /home/npoitoux/vihps/./synccheck-test
=========     Host Frame:cudaLaunchKernel [0x698a8]
=========                 in /home/npoitoux/vihps/./synccheck-test
=========     Host Frame:/usr/local/cuda/bin/../targets/x86_64-linux/include/cuda_runtime.h:211:cudaError cudaLaunchKernel<char
>(char const*, dim3, dim3, void**, unsigned long, CUstream_st*) [0xa31c]
=========                 in /home/npoitoux/vihps/./synccheck-test
=========     Host Frame:/tmp/tmpxft_000404ef_00000000-6_synccheck.cudafe1.stub.c:13:__device_stub__Z6kernelv() [0xa1a7]
=========                 in /home/npoitoux/vihps/./synccheck-test
=========     Host Frame:/home/npoitoux/vihps/synccheck.cu:16:kernel() [0xa210]
=========                 in /home/npoitoux/vihps/./synccheck-test
=========     Host Frame:/home/npoitoux/vihps/synccheck.cu:21:main [0xa05f]
=========                 in /home/npoitoux/vihps/./synccheck-test
=========     Host Frame:__libc_start_main [0x270b3]
=========                 in /lib/x86_64-linux-gnu/libc.so.6
=========     Host Frame:_start [0x9eee]
=========                 in /home/npoitoux/vihps/./synccheck-test
=========
========= Target application returned an error
========= ERROR SUMMARY: 1 error
```

# COMPUTE SANITIZER

Useful options

- `--target-processes=all`: track all children processes of the application
- `--kernel-regex`, `--kernel-regex-exclude`: filter kernel launches to be tracked
- `--launch-count`, `--launch-skip`: only track/ignore n kernel launches (matching the filters if specified)
- `--force-synchronization-limit`: force stream synchronization every n launches (may mitigate memory usage)
- `--xml yes`: XML output for error reports

# COMPUTE SANITIZER

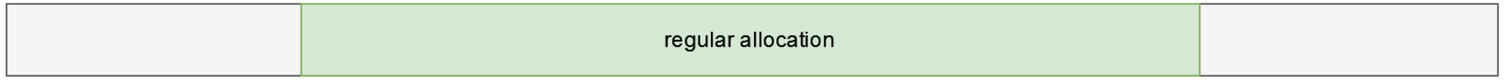
## NVTX

- Support of custom allocators through NVIDIA Tools Extension (NVTX)
- For Initcheck and Memcheck
- Permissions handling
- Naming allocations (leaks only)
- Header "nvToolsExtMem.h" found on https://github.com/NVIDIA/NVTX/tree/dev-mem-api (experimental branch)
- Requires adding `--nvtx=yes`

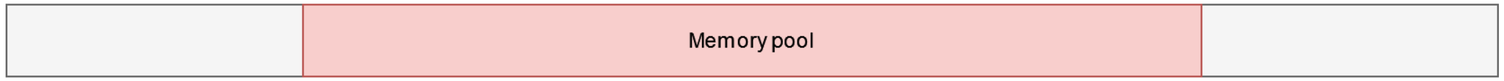
# COMPUTE SANITIZER

## NVTX

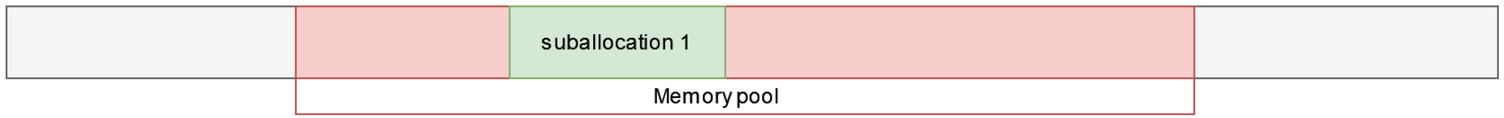
cudaMalloc(&ptr, n)



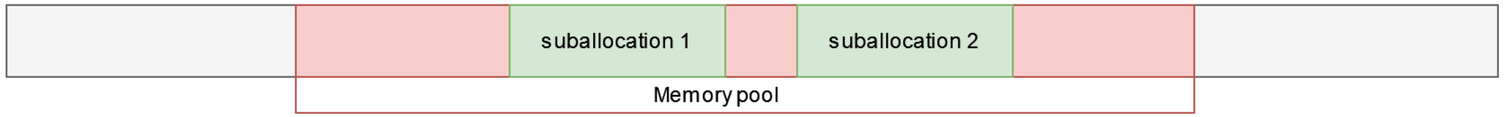

nvtxMemHeapRegister(domain, descriptor)



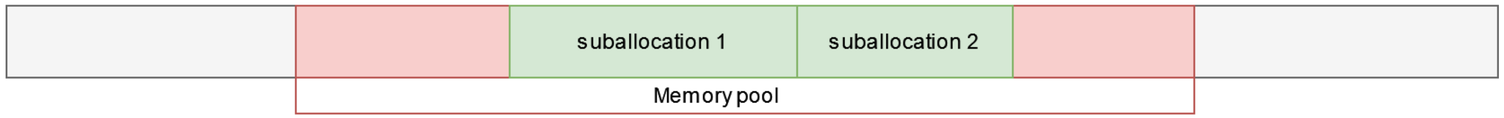

nvtxMemRegionsRegister(domain, descriptor)




nvtxMemRegionsRegister(domain, descriptor)




nvtxMemRegionsResize(domain, descriptor)



# COMPUTE SANITIZER

## NVTX

```
// header found on https://github.com/NVIDIA/NVTX/tree/dev-mem-api
#include <nvtx3/nvToolsExtMem.h>

nvtxMemHeapHandle_t registerMemoryPool(
        void* deviceAlloc, size_t poolSize, nvtxDomainHandle_t domain)
{
    nvtxMemVirtualRangeDesc_t nvtxRangeDesc = {};
    nvtxRangeDesc.size = poolSize;
    nvtxRangeDesc.ptr = deviceAlloc;

    nvtxMemHeapDesc_t nvtxHeapDesc = {};
    nvtxHeapDesc.extCompatID = NVTX_EXT_COMPATID_MEM;
    nvtxHeapDesc.structSize = sizeof(nvtxHeapDesc);
    nvtxHeapDesc.usage = NVTX_MEM_HEAP_USAGE_TYPE_SUB_ALLOCATOR;
    nvtxHeapDesc.type = NVTX_MEM_TYPE_VIRTUAL_ADDRESS;
    nvtxHeapDesc.typeSpecificDescSize = sizeof(nvtxRangeDesc);
    nvtxHeapDesc.typeSpecificDesc = &nvtxRangeDesc;

    return nvtxMemHeapRegister(domain, &nvtxHeapDesc);
}

void registerSuballoc(
        void* suballocStart, size_t allocSize, nvtxDomainHandle_t domain,
        nvtxMemHeapHandle_t pool)
{
    nvtxMemVirtualRangeDesc_t nvtxRangeDesc = {};
    nvtxRangeDesc.size = allocSize;
    nvtxRangeDesc.ptr = suballocStart;

    nvtxMemRegionsRegisterBatch_t nvtxRegionsDesc = {};
    nvtxRegionsDesc.extCompatID = NVTX_EXT_COMPATID_MEM;
    nvtxRegionsDesc.structSize = sizeof(nvtxRegionsDesc);
    nvtxRegionsDesc.regionType = NVTX_MEM_TYPE_VIRTUAL_ADDRESS;
    nvtxRegionsDesc.heap = pool;
    nvtxRegionsDesc.regionCount = 1;
    nvtxRegionsDesc.regionDescElementSize = sizeof(nvtxRangeDesc);
    nvtxRegionsDesc.regionDescElements = &nvtxRangeDesc;

    nvtxMemRegionsRegister(domain, &nvtxRegionsDesc);
}

void unregisterSuballoc(void* suballoc, nvtxDomainHandle_t domain)
{
    nvtxMemRegionRef_t nvtxRegionRef;
    nvtxRegionRef.pointer = suballoc;

    nvtxMemRegionsUnregisterBatch_t nvtxRegionsDesc = {};
    nvtxRegionsDesc.extCompatID = NVTX_EXT_COMPATID_MEM;
    nvtxRegionsDesc.structSize = sizeof(nvtxRegionsDesc);
    nvtxRegionsDesc.refType = NVTX_MEM_REGION_REF_TYPE_POINTER;
    nvtxRegionsDesc.refCount = 1;
    nvtxRegionsDesc.refElementSize = sizeof(nvtxRegionRef);
    nvtxRegionsDesc.refElements = &nvtxRegionRef;

    nvtxMemRegionsUnregister(domain, &nvtxRegionsDesc);
}
```

```
__global__ void accessMemory(char* ptr)
{
    *ptr = 1;
}

int main()
{
    // Creating a domain; required for all NVTX calls
    auto domain = nvtxDomainCreateA("my-domain");

    void *pool;
    cudaMalloc(&pool, 4096);
    auto hNvtxPool = registerMemoryPool(pool, 4096, domain);
    registerSuballoc(pool, 32, domain, hNvtxPool);

    // Valid
    accessMemory<<<1,1>>>((char*)pool);
    cudaDeviceSynchronize();

    // Out of bounds
    accessMemory<<<1,1>>>(((char*)pool) + 32);
    unregisterSuballoc(pool, domain);

    // Out of bounds
    accessMemory<<<1,1>>>((char*)pool);

    nvtxMemHeapUnregister(domain, hNvtxPool);
    cudaFree(pool);
    cudaDeviceReset();
}
```

# COMPUTE SANITIZER

## NVTX

```
$ nvcc -g -G -INVTX/c/include/ nvtx.cu -o nvtx-test
$ compute-sanitizer ./nvtx-test
========= COMPUTE-SANITIZER
========= ERROR SUMMARY: 0 errors
$ compute-sanitizer --destroy-on-device-error=kernel --nvtx=yes --show-backtrace=no ./nvtx-test
========= COMPUTE-SANITIZER
========= Invalid __global__ write of size 1 bytes
=========     at 0x140 in /home/npoitoux/vihps/nvtx.cu:60:accessMemory(char *)
=========     by thread (0,0,0) in block (0,0,0)
=========     Address 0x7f82cfa00020 is out of bounds
=========     and is inside the nearest allocation at 0x7f82cfa00000 of size 4096 bytes
=========
========= Invalid __global__ write of size 1 bytes
=========     at 0x140 in /home/npoitoux/vihps/nvtx.cu:60:accessMemory(char *)
=========     by thread (0,0,0) in block (0,0,0)
=========     Address 0x7f82cfa00000 is out of bounds
=========     and is inside the nearest allocation at 0x7f82cfa00000 of size 4096 bytes
=========
========= ERROR SUMMARY: 2 errors
```

# COMPUTE SANITIZER

## Coredump

- Save program state when an error is encountered
- `--generate-coredump=yes`
- Aborts the program at the first error
- Use cuda-gdb to load the coredump and inspect state
- Doesn’t support racecheck

# COMPUTE SANITIZER

## Coredump

```
$ compute-sanitizer --generate-coredump=yes ./memcheck-test
========= COMPUTE-SANITIZER
========= Invalid __global__ write of size 4 bytes
=========     at 0x1b0 in /home/npoitoux/vihps/memcheck.cu:3:writeIdx(int *)
=========     by thread (31,0,0) in block (0,0,0)
=========     Address 0x7f7295a0007c is out of bounds
=========     and is 1 bytes after the nearest allocation at 0x7f7295a00000 of size 124 bytes
=========     Device Frame:/home/npoitoux/vihps/memcheck.cu:8:kernel(int *) [0xd0]
=========     Saved host backtrace up to driver entry point at kernel launch time
=========     Host Frame: [0x21740c]
=========                 in /usr/lib/x86_64-linux-gnu/libcuda.so.1
=========     Host Frame:__cudart803 [0xeabb]
=========                 in /home/npoitoux/vihps/./memcheck-test
=========     Host Frame:cudaLaunchKernel [0x69918]
=========                 in /home/npoitoux/vihps/./memcheck-test
=========     Host Frame:/usr/local/cuda/bin/../targets/x86_64-linux/include/cuda_runtime.h:211:cudaError cudaLaunchKernel<char
>(char const*, dim3, dim3, void**, unsigned long, CUstream_st*) [0xa392]
=========                 in /home/npoitoux/vihps/./memcheck-test
=========     Host Frame:/tmp/tmpxft_000405c8_00000000-6_memcheck.cudafe1.stub.c:13:__device_stub__Z6kernelPi(int*) [0xa242]
=========                 in /home/npoitoux/vihps/./memcheck-test
=========     Host Frame:/home/npoitoux/vihps/memcheck.cu:9:kernel(int*) [0xa286]
=========                 in /home/npoitoux/vihps/./memcheck-test
=========     Host Frame:/home/npoitoux/vihps/memcheck.cu:16:main [0xa0b0]
=========                 in /home/npoitoux/vihps/./memcheck-test
=========     Host Frame:__libc_start_main [0x270b3]
=========                 in /lib/x86_64-linux-gnu/libc.so.6
=========     Host Frame:_start [0x9efe]
=========                 in /home/npoitoux/vihps/./memcheck-test
=========
========= Error: process didn't terminate successfully
========= Target application returned an error
========= ERROR SUMMARY: 1 error
$ ls *.nvcudmp
core_1644171832_npoitoux-linux_263666.nvcudmp
```

# COMPUTE SANITIZER

## Coredump

```
$ cuda-gdb
NVIDIA (R) CUDA Debugger
11.6 release
Portions Copyright (C) 2007-2021 NVIDIA Corporation
GNU gdb (GDB) 10.2
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word".
(cuda-gdb) target cudacore core_1644171832_npoitoux-linux_263666.nvcudmp
Opening GPU coredump: core_1644171832_npoitoux-linux_263666.nvcudmp
Program terminated with signal SIGTRAP, Trace/breakpoint trap.
[Current focus set to CUDA kernel 0, grid 1, block (0,0,0), thread (31,0,0), device 0, sm 0, warp 0, lane 31]
#0  0x0000556c1aa387b0 in writeIdx (buffer=0x7f7295a00000) at memcheck.cu:3
3           buffer[threadIdx.x] = threadIdx.x;
(cuda-gdb) p buffer
$1 = (@generic int * @register) 0x7f7295a00000
(cuda-gdb) list
1       __device__ void writeIdx(int* buffer)
2       {
3           buffer[threadIdx.x] = threadIdx.x;
4       }
5
6       __global__ void kernel(int* buffer)
7       {
8           writeIdx(buffer);
9       }
10
(cuda-gdb) p threadIdx.x
$2 = 31
(cuda-gdb) bt
#0  0x0000556c1aa387b0 in writeIdx (buffer=0x7f7295a00000) at memcheck.cu:3
#1  0x0000556c1aa3a360 in kernel<<<(1,1,1),(32,1,1)>>> (buffer=0x7f7295a00000) at memcheck.cu:8
(cuda-gdb)
```

# COMPUTE SANITIZER

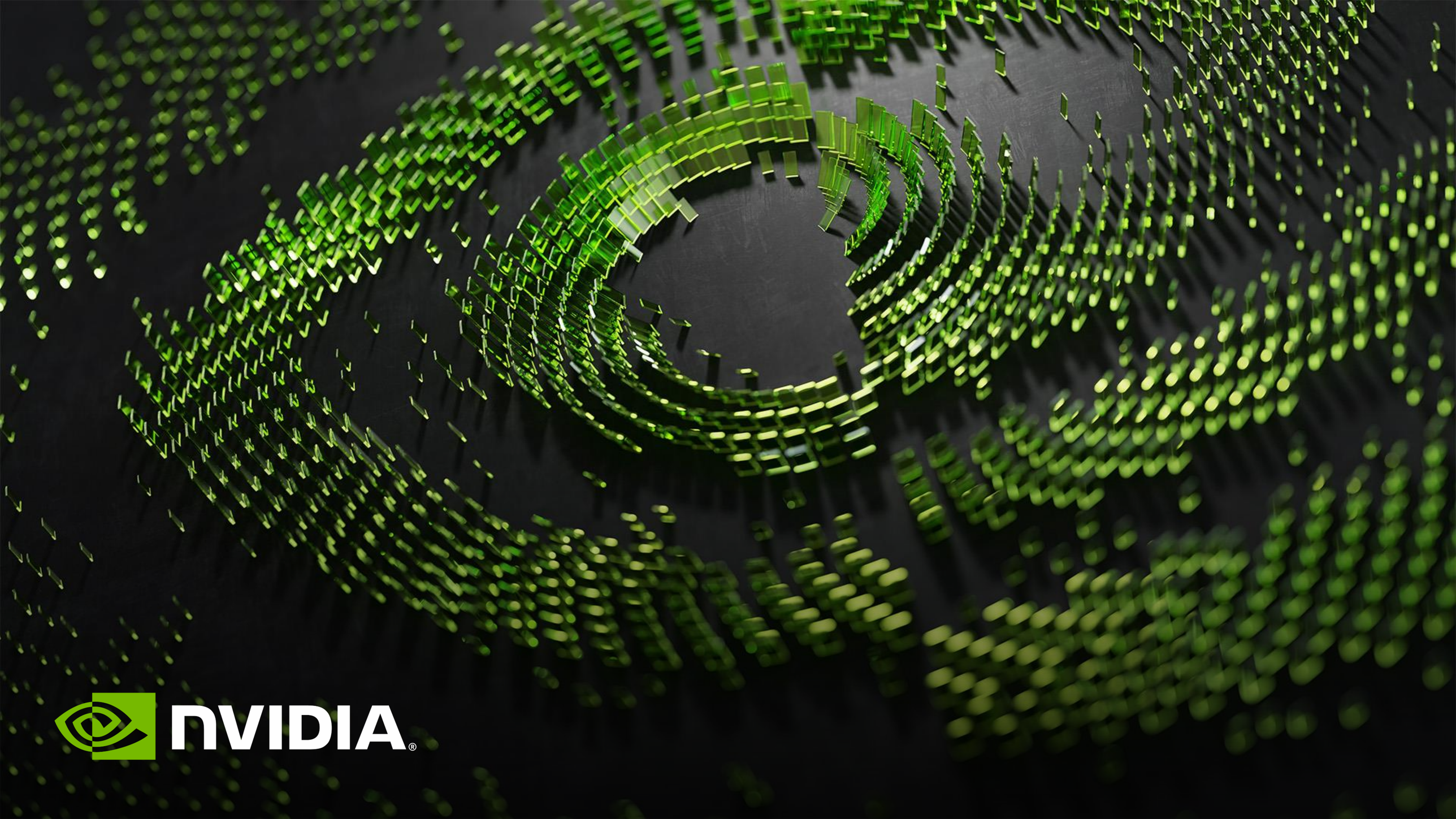
## Sanitizer API

- Compute-sanitizer built on top of the Sanitizer public API
- Allow users to build customized tools
  - Keren Zhou, Yueming Hao, John Mellor-Crummey, Xiaozhu Meng, Xu Liu. "GVPROF: A Value Profiler for GPU-Based Clusters", SC20 (2020). https://dl.acm.org/doi/10.5555/3433701.3433819
- Headers in /usr/local/cuda/compute-sanitizer/include
- Callbacks on host CUDA events
- Patching API to get callbacks on device in-kernel events

# COMPUTE SANITIZER

Links

- Documentation: https://docs.nvidia.com/compute-sanitizer/index.html
- Compute-sanitizer and public api samples: https://github.com/NVIDIA/compute-sanitizer-samples


NVIDIA