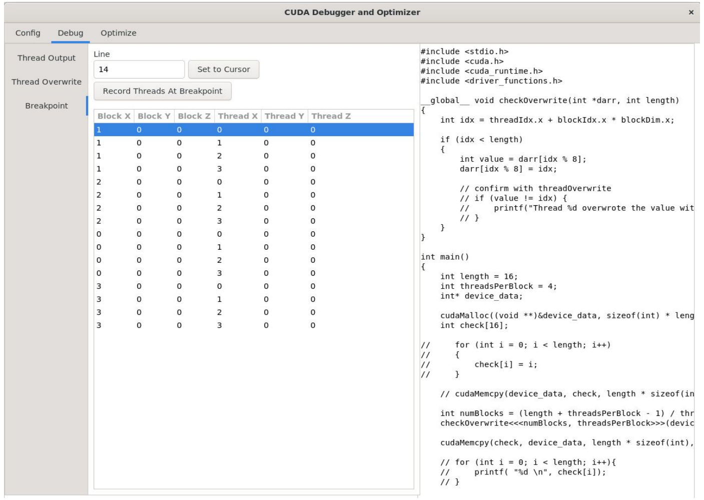
Summary & Background

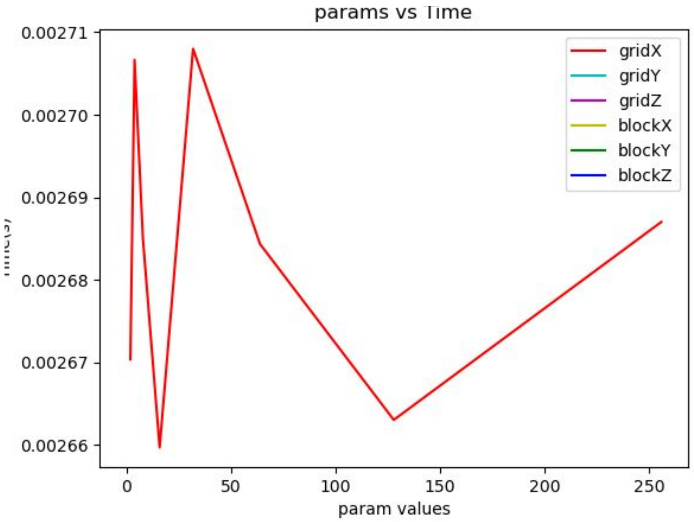
- ➔ Project: CUDA debugger/optimizer
  - ◆ thread output, thread overwriting, breakpoint
  - ◆ time/memory bottleneck, optimal macro config, speedup
- ➔ Produce values and visuals to simplify the debugging/optimizing process



Line: 14

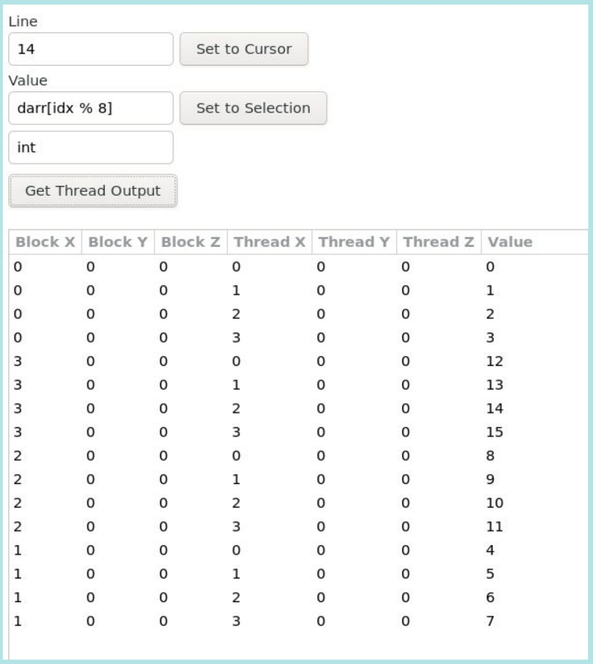
Block X	Block Y	Block Z	Thread X	Thread Y	Thread Z
1	0	0	0	0	0
1	0	0	1	0	0
1	0	0	2	0	0
1	0	0	3	0	0
2	0	0	0	0	0
2	0	0	1	0	0
2	0	0	2	0	0
2	0	0	3	0	0
0	0	0	0	0	0
0	0	0	1	0	0
0	0	0	2	0	0
0	0	0	3	0	0
3	0	0	0	0	0
3	0	0	1	0	0
3	0	0	2	0	0
3	0	0	3	0	0

```
#include <stdio.h>
#include <cuda.h>
#include <cuda_runtime.h>
#include <driver_functions.h>

__global__ void checkOverwrite(int *darr, int length)
{
    int idx = threadIdx.x + blockIdx.x * blockDim.x;

    if (idx < length)
    {
        int value = darr[idx % 8];
        darr[idx % 8] = idx;

        // confirm with threadOverwrite
        // if (value != idx) {
        //     printf("Thread %d overwrote the value wit
        // }
    }
}

int main()
{
    int length = 16;
    int threadsPerBlock = 4;
    int* device_data;

    cudaMalloc((void **)&device_data, sizeof(int) * leng
    int check[16];

//     for (int i = 0; i < length; i++)
//     {
//         check[i] = i;
//     }

    // cudaMemcpy(device_data, check, length * sizeof(in

    int numBlocks = (length + threadsPerBlock - 1) / thr
    checkOverwrite<<<numBlocks, threadsPerBlock>>>(devic

    cudaMemcpy(check, device_data, length * sizeof(int),

    // for (int i = 0; i < length; i++){
    //     printf( "%d \n", check[i]);
    // }
```



threadOutput & threadOverwrite

Output: blockIdx, threadIdx, and updated variable value



Line

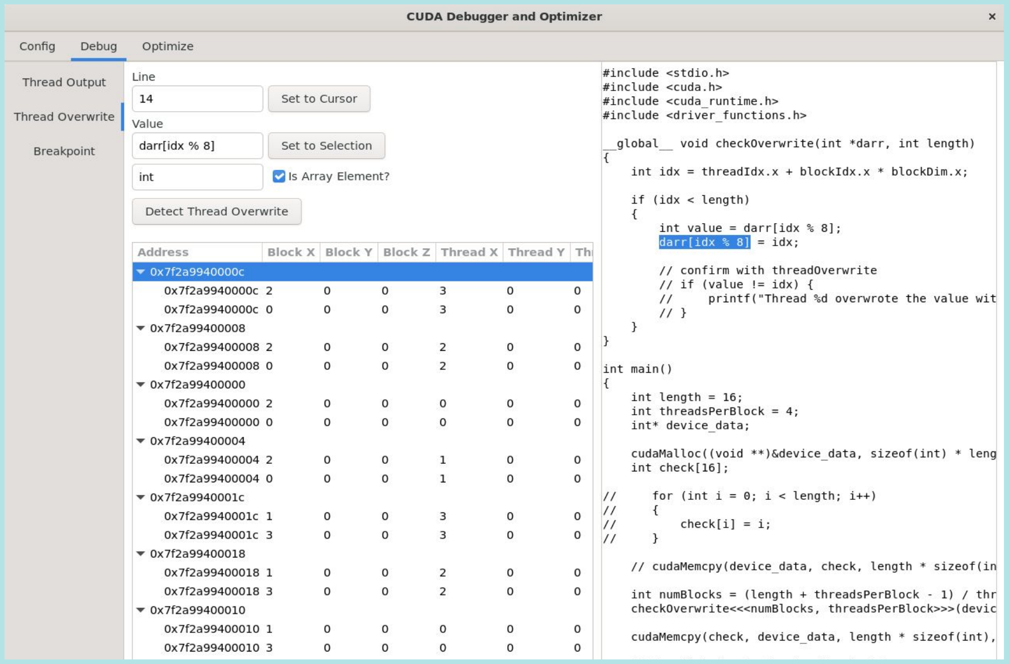
14

Value

darr[idx % 8]

int

Block X	Block Y	Block Z	Thread X	Thread Y	Thread Z	Value
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	0	2	0	0	2
0	0	0	3	0	0	3
3	0	0	0	0	0	12
3	0	0	1	0	0	13
3	0	0	2	0	0	14
3	0	0	3	0	0	15
2	0	0	0	0	0	8
2	0	0	1	0	0	9
2	0	0	2	0	0	10
2	0	0	3	0	0	11
1	0	0	0	0	0	4
1	0	0	1	0	0	5
1	0	0	2	0	0	6
1	0	0	3	0	0	7



Line

14

Value

darr[idx % 8]

int

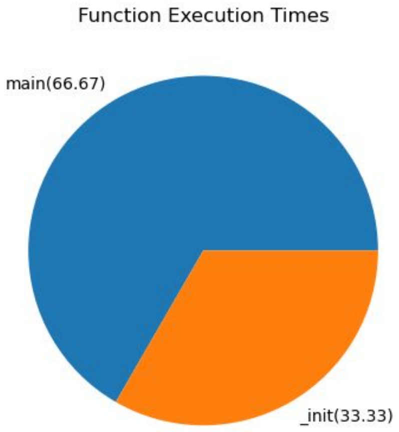
☑ Is Array Element?

Address	Block X	Block Y	Block Z	Thread X	Thread Y	Th
0x7f2a9940000c						
0x7f2a9940000c	2	0	0	3	0	0
0x7f2a9940000c	0	0	0	3	0	0
0x7f2a99400008						
0x7f2a99400008	2	0	0	2	0	0
0x7f2a99400008	0	0	0	2	0	0
0x7f2a99400000						
0x7f2a99400000	2	0	0	0	0	0
0x7f2a99400000	0	0	0	0	0	0
0x7f2a99400004						
0x7f2a99400004	2	0	0	1	0	0
0x7f2a99400004	0	0	0	1	0	0
0x7f2a9940001c						
0x7f2a9940001c	1	0	0	3	0	0
0x7f2a9940001c	3	0	0	3	0	0
0x7f2a99400018						
0x7f2a99400018	1	0	0	2	0	0
0x7f2a99400018	3	0	0	2	0	0
0x7f2a99400010						
0x7f2a99400010	1	0	0	0	0	0
0x7f2a99400010	3	0	0	0	0	0

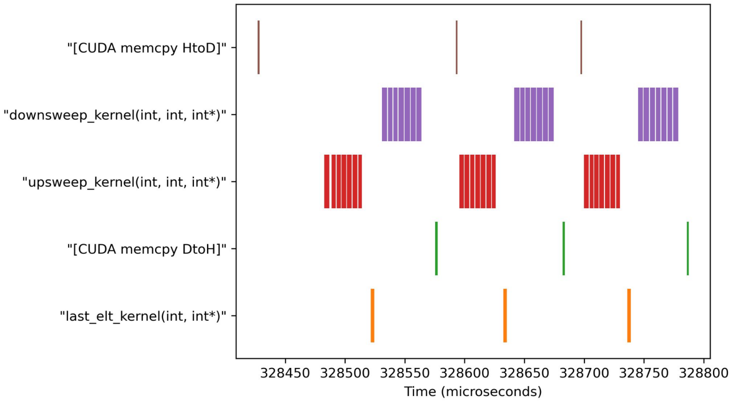
```
#include <stdio.h>
#include <cuda.h>
#include <cuda_runtime.h>
#include <driver_functions.h>

__global__ void checkOverwrite(int *darr, int length)
{
    int idx = threadIdx.x + blockIdx.x * blockDim.x;

    if (idx < length)
    {
        int value = darr[idx % 8];
        darr[idx % 8] = idx;

        // confirm with threadOverwrite
        // if (value != idx) {
        //     printf("Thread %d overwrote the value wit
        // }
    }
}

int main()
{
    int length = 16;
    int threadsPerBlock = 4;
    int* device_data;

    cudaMalloc((void **)&device_data, sizeof(int) * leng
    int check[16];

//     for (int i = 0; i < length; i++)
//     {
//         check[i] = i;
//     }

    // cudaMemcpy(device_data, check, length * sizeof(in

    int numBlocks = (length + threadsPerBlock - 1) / thr
    checkOverwrite<<<numBlocks, threadsPerBlock>>>(devic

    cudaMemcpy(check, device_data, length * sizeof(int),
```

Output: address, blockIdx, threadIdx, (if array, include index), and updated variable value

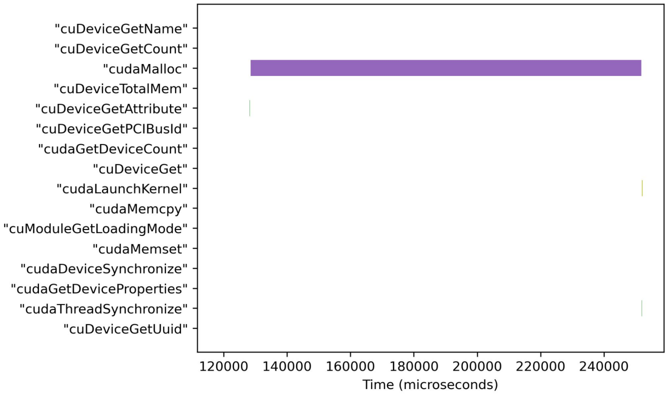
Time Bottleneck: gprof/GPU/API



gprof: function runtime as a % of total runtime



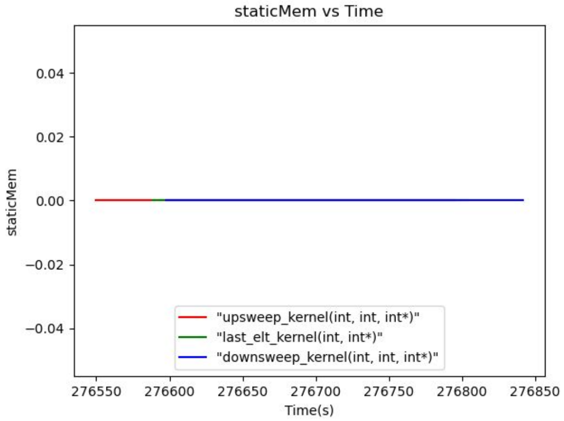
GPU

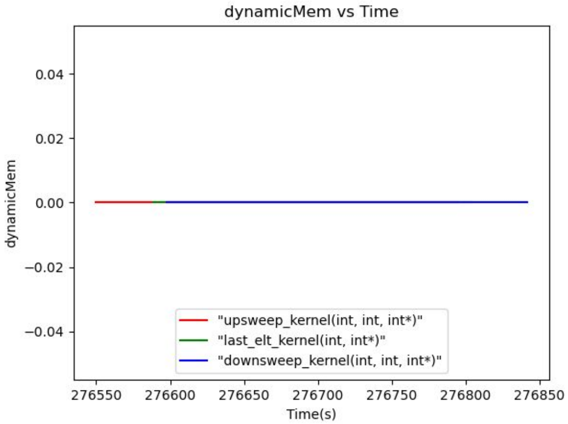


API

Memory Bottleneck: nvprof

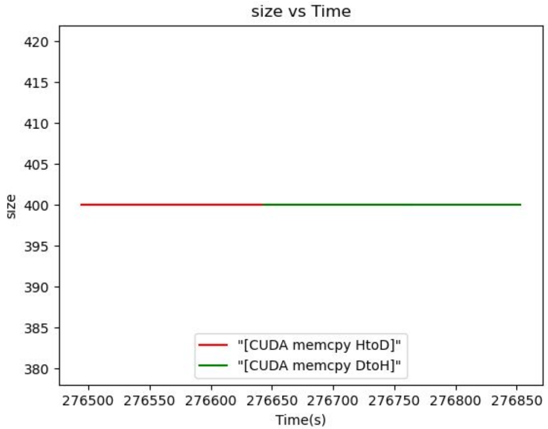
staticMem

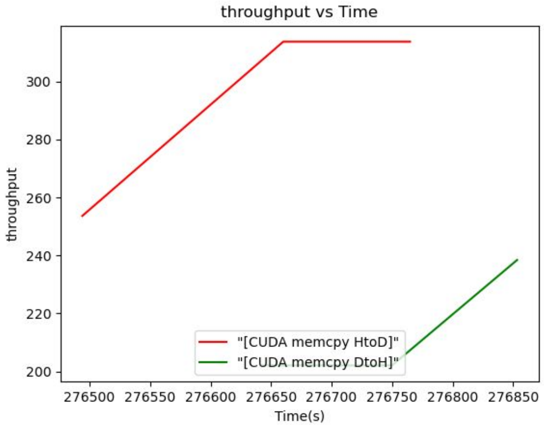




dynamicMem

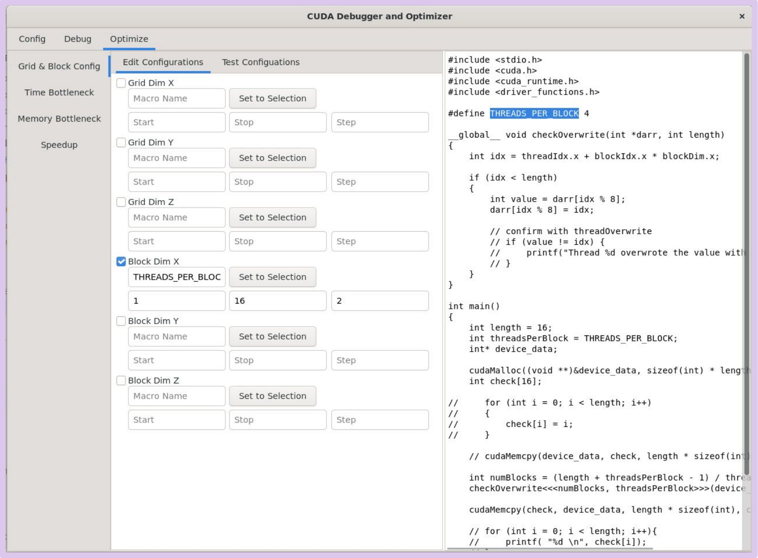
Mem size





throughput

Optimal Config



☐ Grid Dim X

☐ Grid Dim Y

☐ Grid Dim Z

☑ Block Dim X

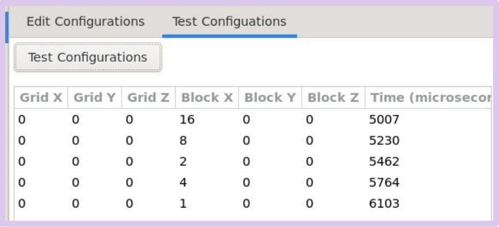
THREADS_PER_BLOC

1 | 16 | 2

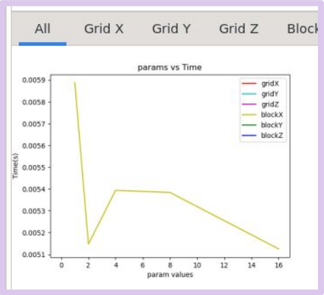
☐ Block Dim Y

☐ Block Dim Z

```
#include <stdio.h>
#include <cuda.h>
#include <cuda_runtime.h>
#include <driver_functions.h>

#define THREADS_PER_BLOCK 4

__global__ void checkOverwrite(int *darr, int length)
{
    int idx = threadIdx.x + blockIdx.x * blockDim.x;

    if (idx < length)
    {
        int value = darr[idx % 8];
        darr[idx % 8] = idx;

        // confirm with threadOverwrite
        // if (value != idx) {
        //     printf("Thread %d overwrote the value with
        // }
    }
}

int main()
{
    int length = 16;
    int threadsPerBlock = THREADS_PER_BLOCK;
    int* device_data;

    cudaMalloc((void **)&device_data, sizeof(int) * length
    int check[16];

//     for (int i = 0; i < length; i++)
//     {
//         check[i] = i;
//     }

    // cudaMemcpy(device_data, check, length * sizeof(int)

    int numBlocks = (length + threadsPerBlock - 1) / threa
    checkOverwrite<<<numBlocks, threadsPerBlock>>>(device_

    cudaMemcpy(check, device_data, length * sizeof(int), c

    // for (int i = 0; i < length; i++){
    //     printf( "%d \n", check[i]);
```



Grid X	Grid Y	Grid Z	Block X	Block Y	Block Z	Time (microsecor
0	0	0	16	0	0	5007
0	0	0	8	0	0	5230
0	0	0	2	0	0	5462
0	0	0	4	0	0	5764
0	0	0	1	0	0	6103



➔ User selects parameters they want to change, and gives a range to help generate all possible configs

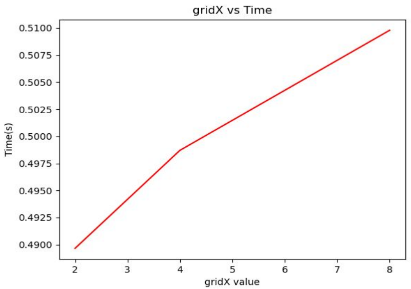
➔ Run code with different params

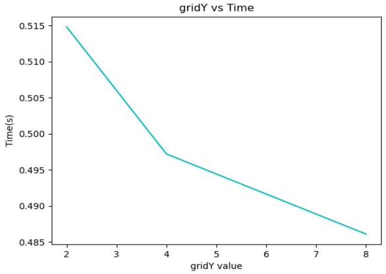
➔ Add runtime of code to params line

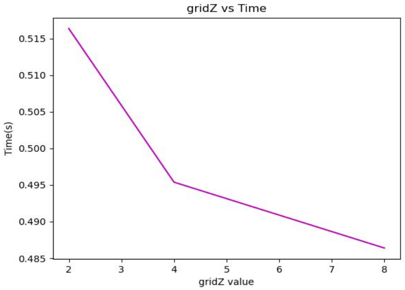
➔ Note: Output of optimal config is used to generate parameter graphs

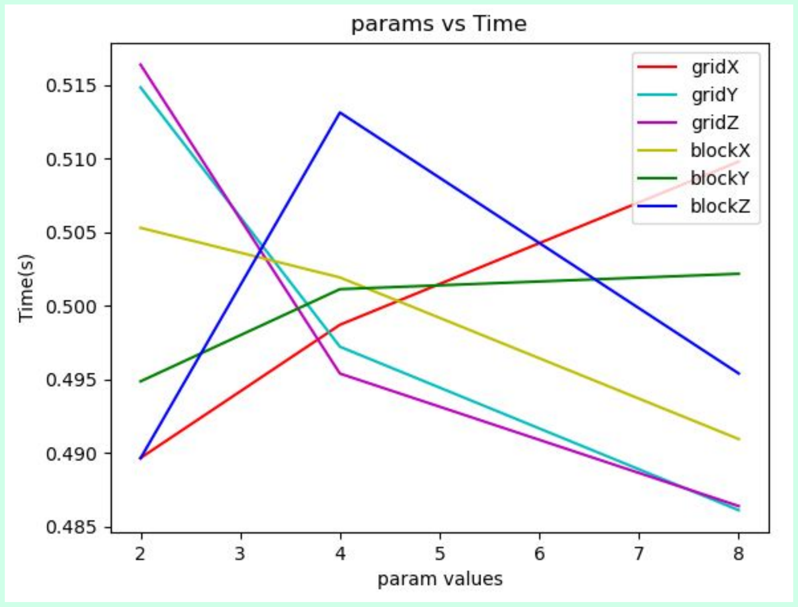
➔ Requires user to have definitions of macros in the code

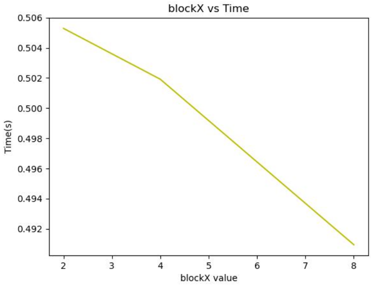
Parameter Graphs

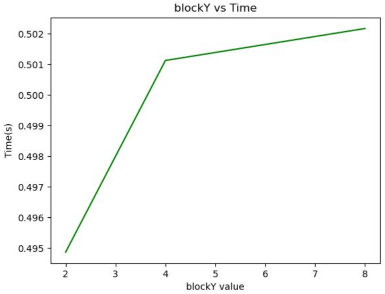


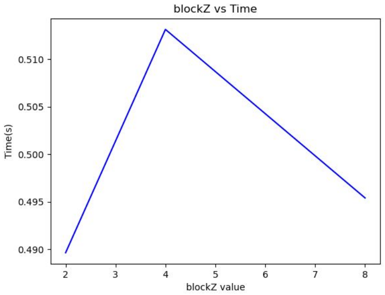






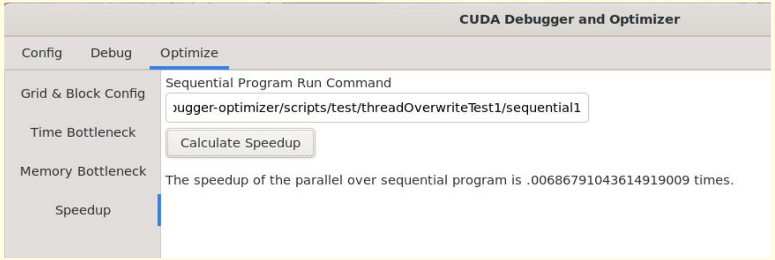






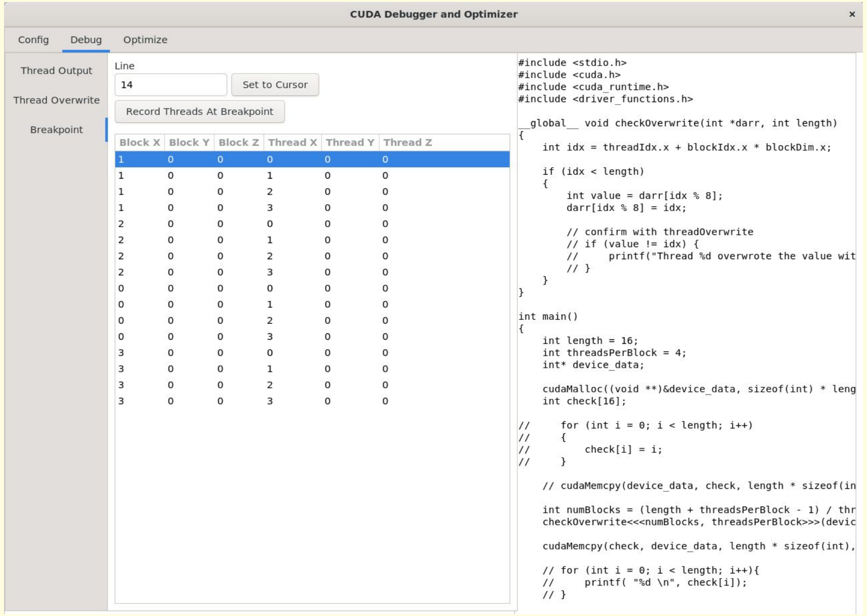
Breakpoint & Speedup

- ➔ Output: sorted list of threads (blockIdx, threadIdx)
  - ◆ Purpose of sort: easier to identify missing threads



Sequential Program Run Command: ›ugger-optimizer/scripts/test/threadOverwriteTest1/sequential1

The speedup of the parallel over sequential program is .00686791043614919009 times.



Line: 14

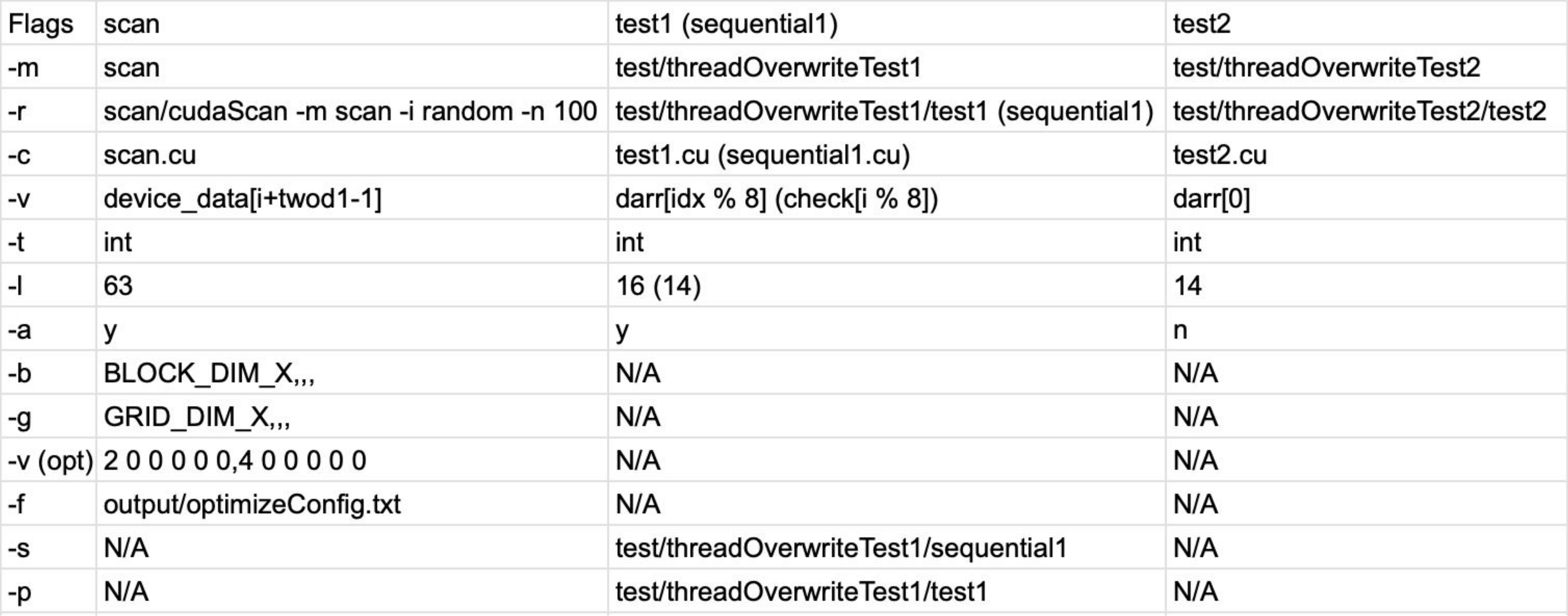
Block X	Block Y	Block Z	Thread X	Thread Y	Thread Z
1	0	0	0	0	0
1	0	0	1	0	0
1	0	0	2	0	0
1	0	0	3	0	0
2	0	0	0	0	0
2	0	0	1	0	0
2	0	0	2	0	0
2	0	0	3	0	0
0	0	0	0	0	0
0	0	0	1	0	0
0	0	0	2	0	0
0	0	0	3	0	0
3	0	0	0	0	0
3	0	0	1	0	0
3	0	0	2	0	0
3	0	0	3	0	0

```
#include <stdio.h>
#include <cuda.h>
#include <cuda_runtime.h>
#include <driver_functions.h>

__global__ void checkOverwrite(int *darr, int length)
{
    int idx = threadIdx.x + blockIdx.x * blockDim.x;

    if (idx < length)
    {
        int value = darr[idx % 8];
        darr[idx % 8] = idx;

        // confirm with threadOverwrite
        // if (value != idx) {
        //     printf("Thread %d overwrote the value wit
        // }
    }
}

int main()
{
    int length = 16;
    int threadsPerBlock = 4;
    int* device_data;

    cudaMalloc((void **)&device_data, sizeof(int) * leng
    int check[16];

//     for (int i = 0; i < length; i++)
//     {
//         check[i] = i;
//     }

    // cudaMemcpy(device_data, check, length * sizeof(in

    int numBlocks = (length + threadsPerBlock - 1) / thr
    checkOverwrite<<<numBlocks, threadsPerBlock>>>(devic

    cudaMemcpy(check, device_data, length * sizeof(int),

    // for (int i = 0; i < length; i++){
    //     printf( "%d \n", check[i]);
    // }
```

CUDA Debugger & Optimizer

kflorend & huiningl

Interactive Test Cases!



Flags	scan	test1 (sequential1)	test2
-m	scan	test/threadOverwriteTest1	test/threadOverwriteTest2
-r	scan/cudaScan -m scan -i random -n 100	test/threadOverwriteTest1/test1 (sequential1)	test/threadOverwriteTest2/test2
-c	scan.cu	test1.cu (sequential1.cu)	test2.cu
-v	device_data[i+twod1-1]	darr[idx % 8] (check[i % 8])	darr[0]
-t	int	int	int
-l	63	16 (14)	14
-a	y	y	n
-b	BLOCK_DIM_X,,,	N/A	N/A
-g	GRID_DIM_X,,,	N/A	N/A
-v (opt)	2 0 0 0 0 0,4 0 0 0 0 0	N/A	N/A
-f	output/optimizeConfig.txt	N/A	N/A
-s	N/A	test/threadOverwriteTest1/sequential1	N/A
-p	N/A	test/threadOverwriteTest1/test1	N/A