Massively Parallel, Highly Efficient, but What About the Test Suite Quality? **Applying Mutation Testing to GPU Programs**

Qianqian Zhu

Co-authors: Andy Zaidman

Software Engineering Research Group, Delft University of Technology, Netherlands

ICST 2020, October 26, 2020

Example of GPU programming

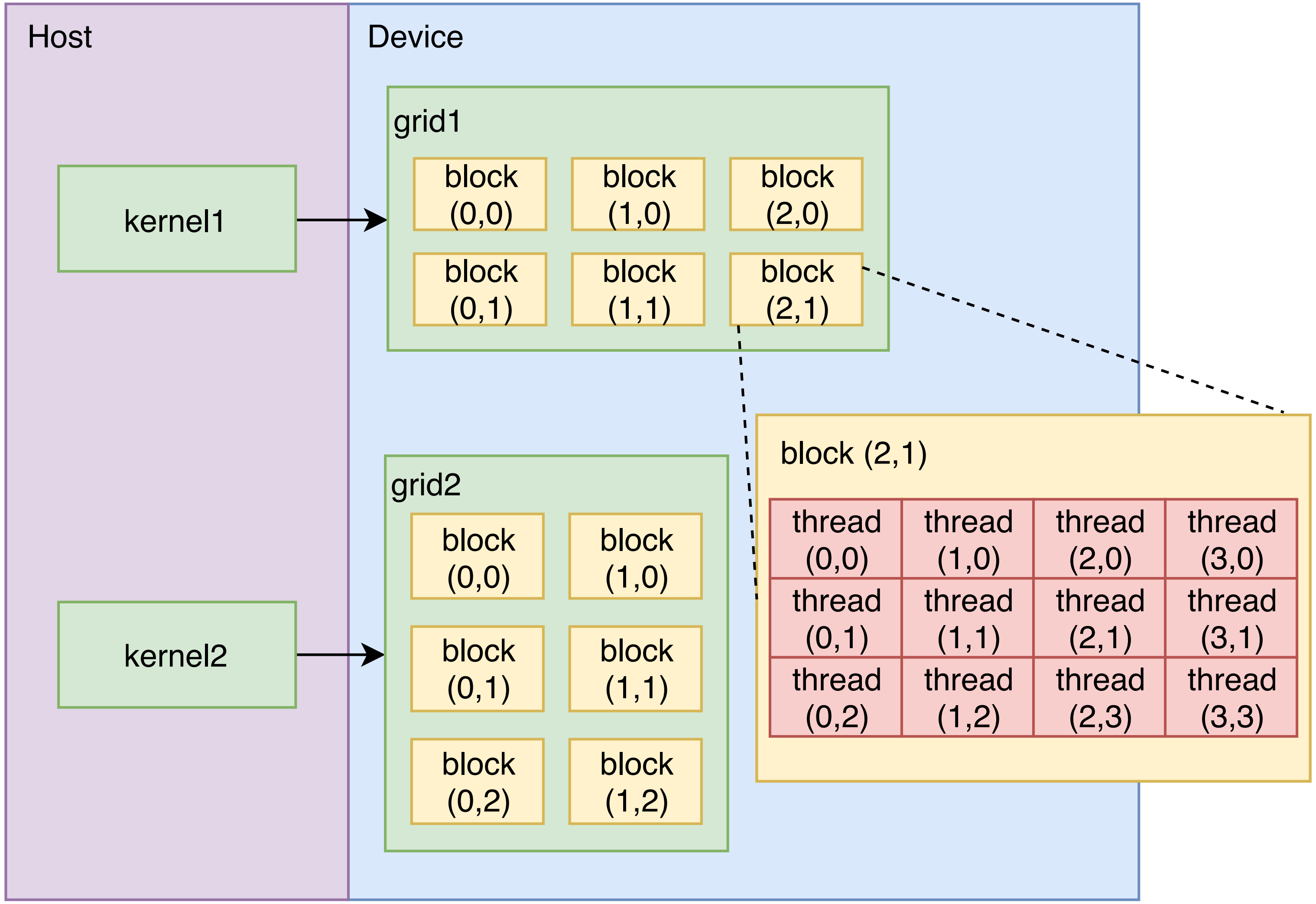
```
void sum(int n, float *a, float *b, float *c){
  for (int i = 0; i < n; i++){
    c[i] = a[i] + b[i];
  }
}
```

sum function in Standard C

```
__global__ void sum(int n, float *a, float *b, float *c){
 int i = blockIdx.x*blockDim.x + threadIdx.x;
 if(i<n){
   c[i] = a[i] + b[i];
  }
}
```

sum function in CUDA C

CUDA programming model



Host

kernel1

kernel2

Device

grid1

block (0,0) block (1,0) block (2,0)

block (0,1) block (1,1) block (2,1)

grid2

block (0,0) block (1,0)

block (0,1) block (1,1)

block (0,2) block (1,2)

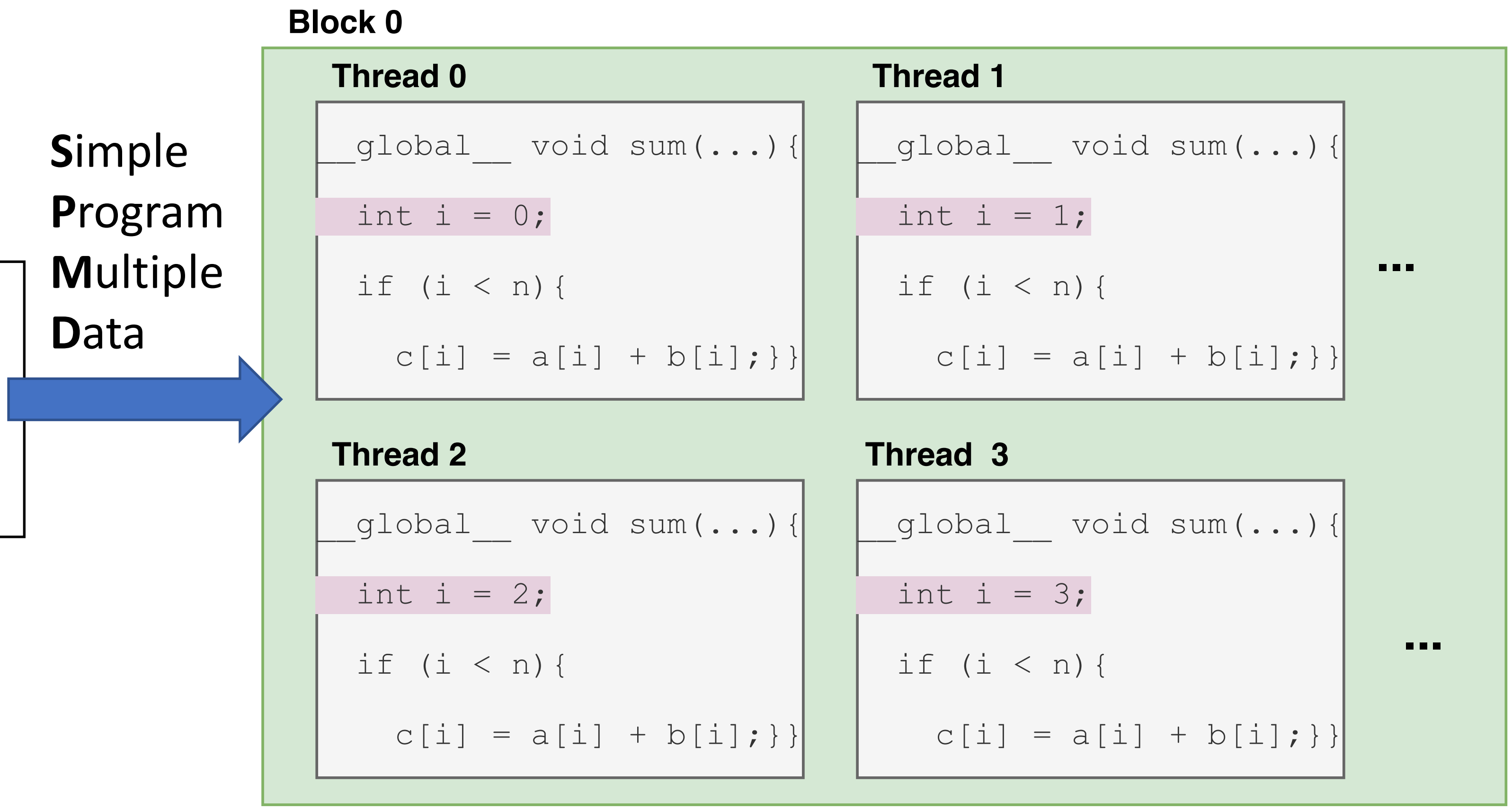
block (2,1)

thread (0,0)	thread (1,0)	thread (2,0)	thread (3,0)
thread (0,1)	thread (1,1)	thread (2,1)	thread (3,1)
thread (0,2)	thread (1,2)	thread (2,3)	thread (3,3)

Example of GPU programming

```
__global__ void sum(int n, float *a, float *b, float *c){
  int i = blockIdx.x*blockDim.x + threadIdx.x;
  if(i<n){
     c[i] = a[i] + b[i];
   }
}
```

Simple
Program
Multiple
Data



Thread 0

```
__global__ void sum(...){
 int i = 0;
 if (i < n){
   c[i] = a[i] + b[i];}}
```

Thread 1

```
__global__ void sum(...){
 int i = 1;
 if (i < n){
   c[i] = a[i] + b[i];}}
```

Thread 2

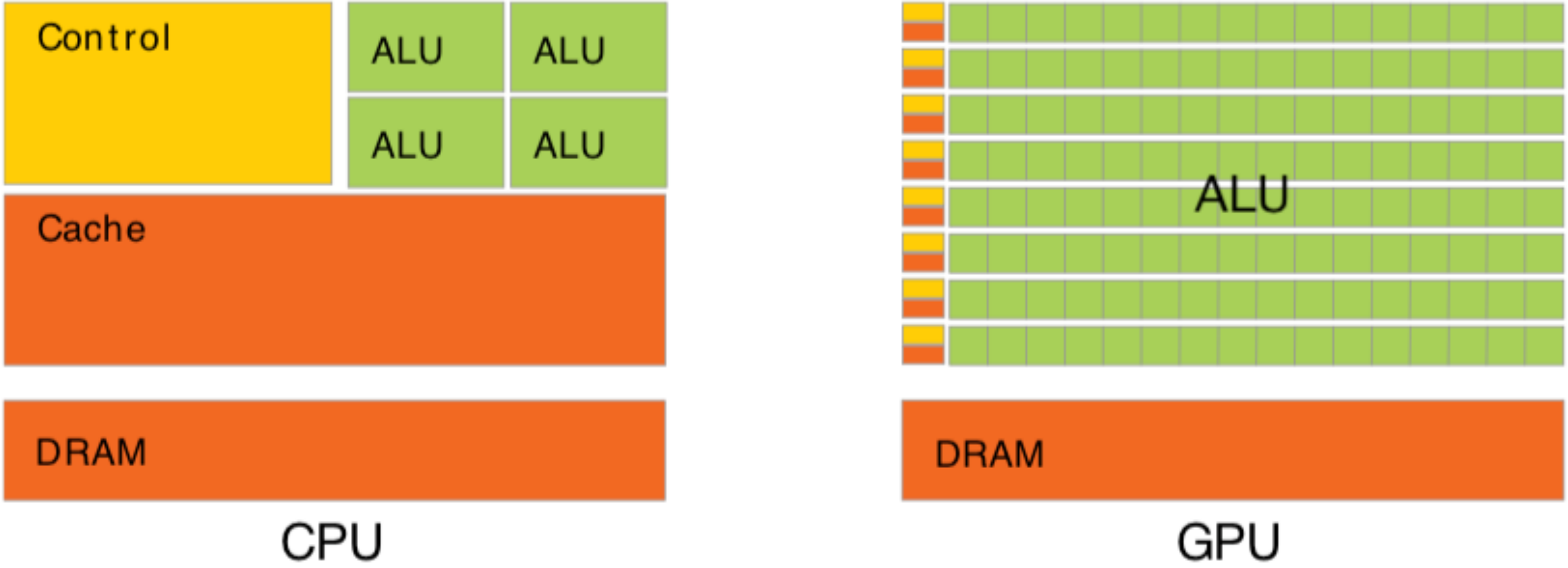
```
__global__ void sum(...){
 int i = 2;
 if (i < n){
   c[i] = a[i] + b[i];}}
```

Thread 3

```
__global__ void sum(...){
 int i = 3;
 if (i < n){
   c[i] = a[i] + b[i];}}
```

Actual code in CUDA parallel threads

Comparison of CPU and GPU



GPU contains many more transistors devoted to data processing rather than data caching and flow control

Challenges in GPU programming

- *Thread management*

```
int tid = blockIdx.x
     + threadIdx.x * blockDim.x;
```

Indexing Bugs

Challenges in GPU programming

- *Memory management*

```
__shared__ float cache[threadsPerBlock];
int i = blockDim.x/2;
while (i != 0) {
    if (caIndex < i)
       cache[caIndex]+=cache[caIndex + i];
    __syncthreads();
    i /= 2;
}
```

Shared Memory Bugs

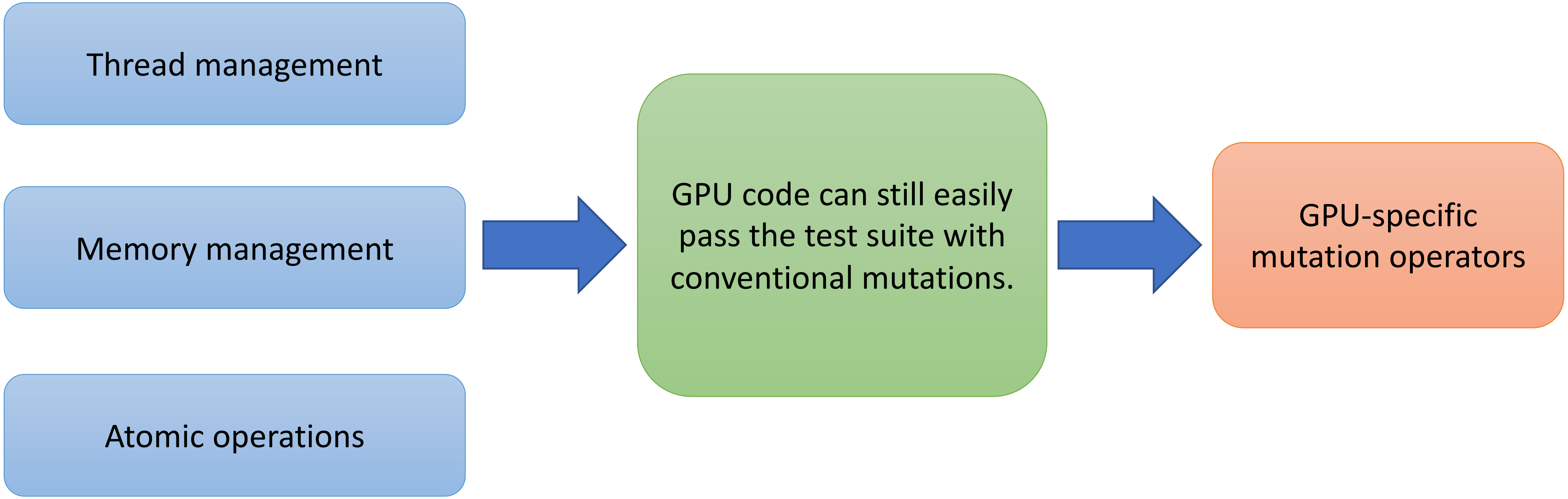
Challenges in GPU programming

- *Atomic operations*

```
__global__ void histo_kernel(
                   unsigned char *buffer,
                   long size,
                   unsigned int *histo ){
int i = threadIdx.x +
             blockIdx.x * blockDim.x;
int stride = blockDim.x * gridDim.x;
while (i < size) {
        histo[buffer[i]] += 1;
        i += stride;
  }
}
```

Atomic Operation Omissions

Challenges in GPU programming



Thread management

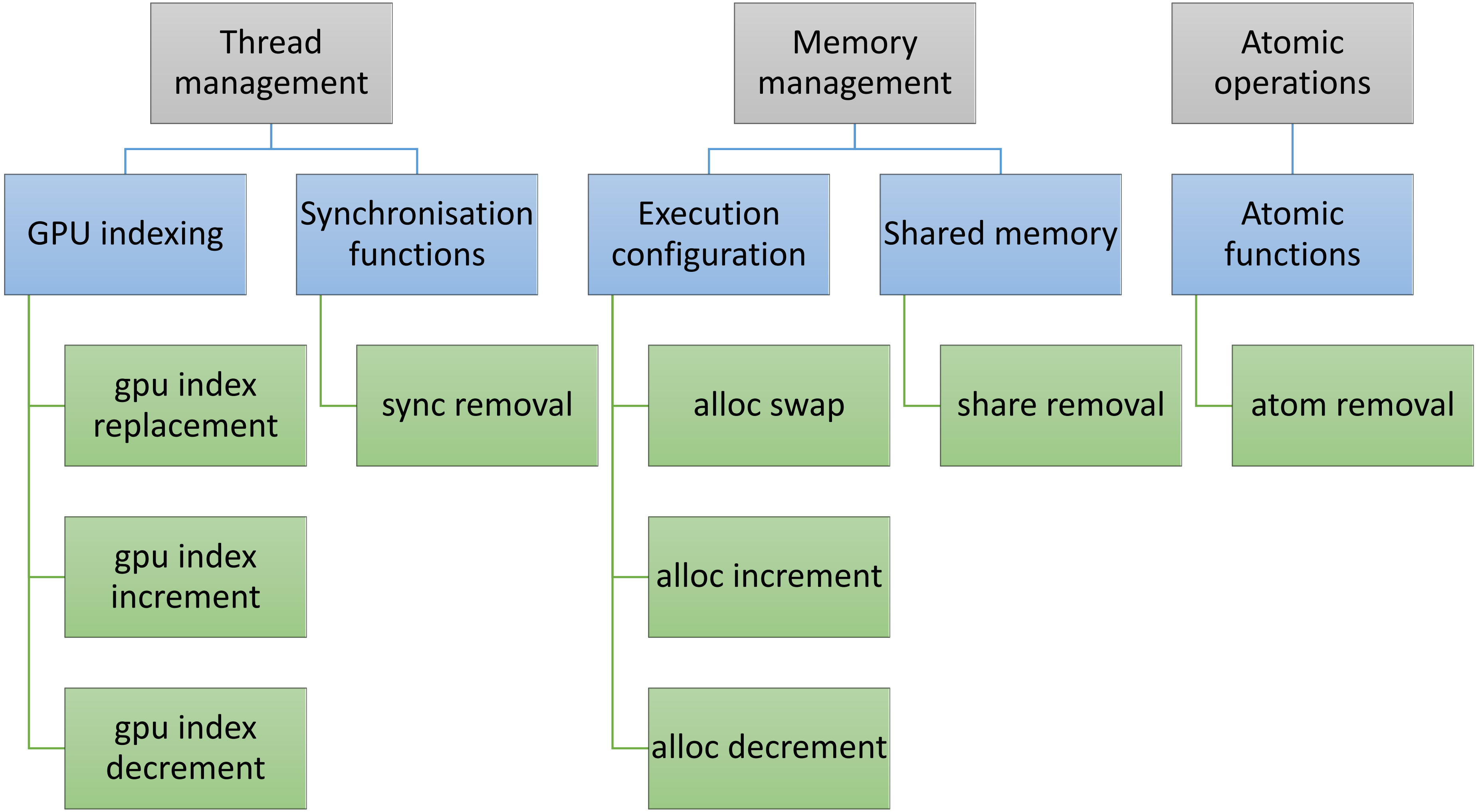
Memory management

Atomic operations

GPU code can still easily pass the test suite with conventional mutations.

GPU-specific mutation operators

GPU-specific mutation operators



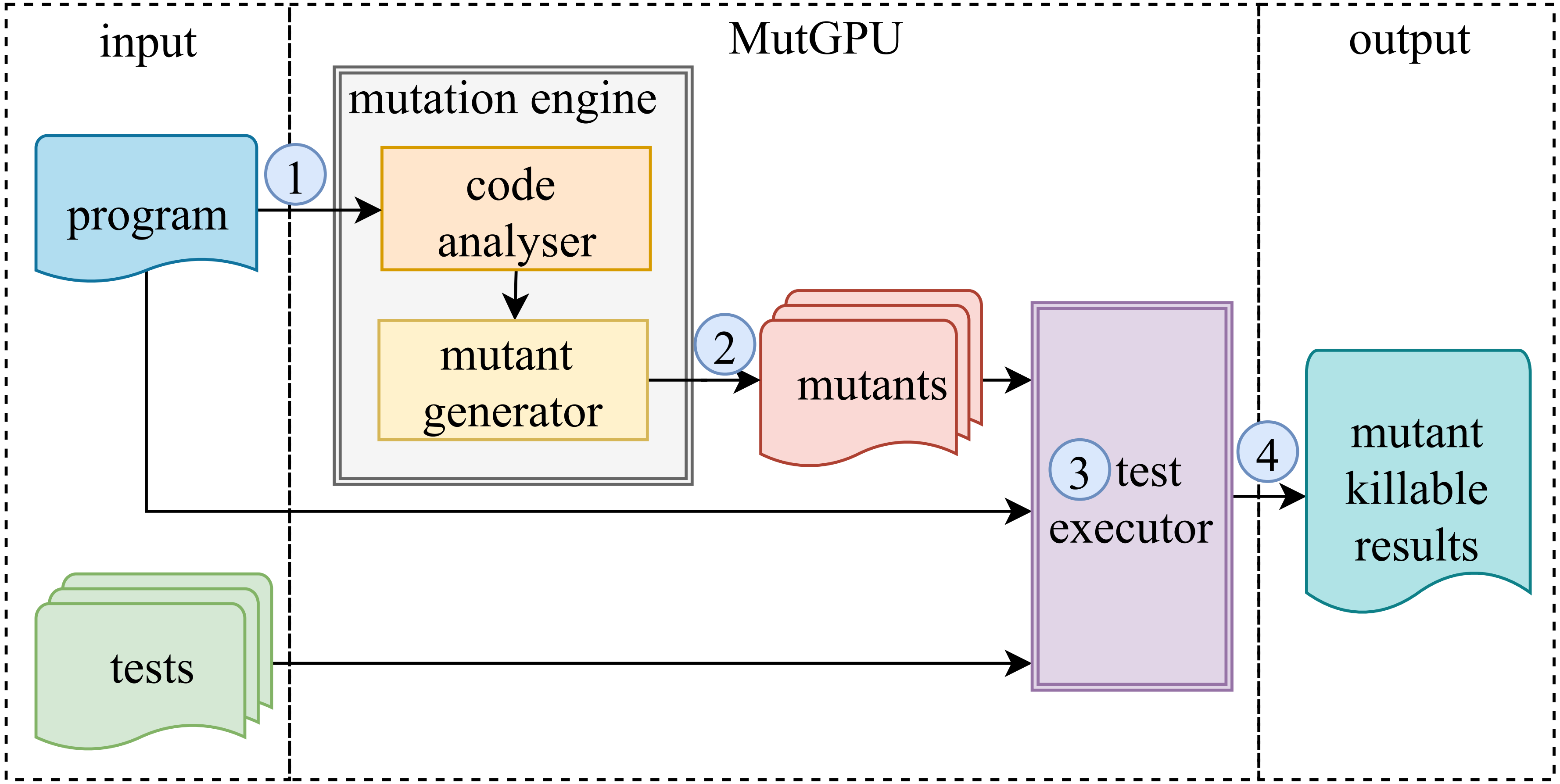
Experimental study

- Subjects

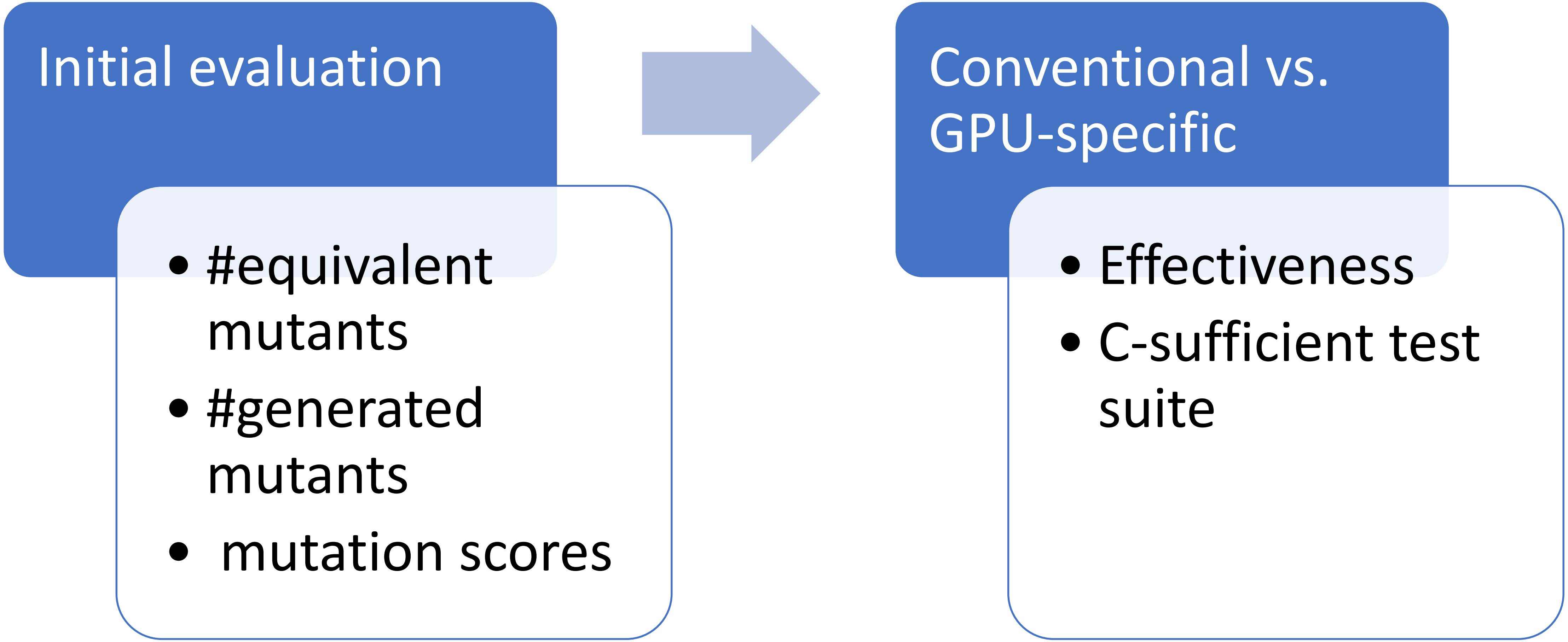
- 6 GPU benchmark projects from CUDA SDK
- 2 NVIDIA graphic cards
  - GeForce MX150 & GTX 960
- 2 releases of CUDA toolkit (9.0 & 9.1)

Experimental study

- Tool



Experimental setup



Initial evaluation

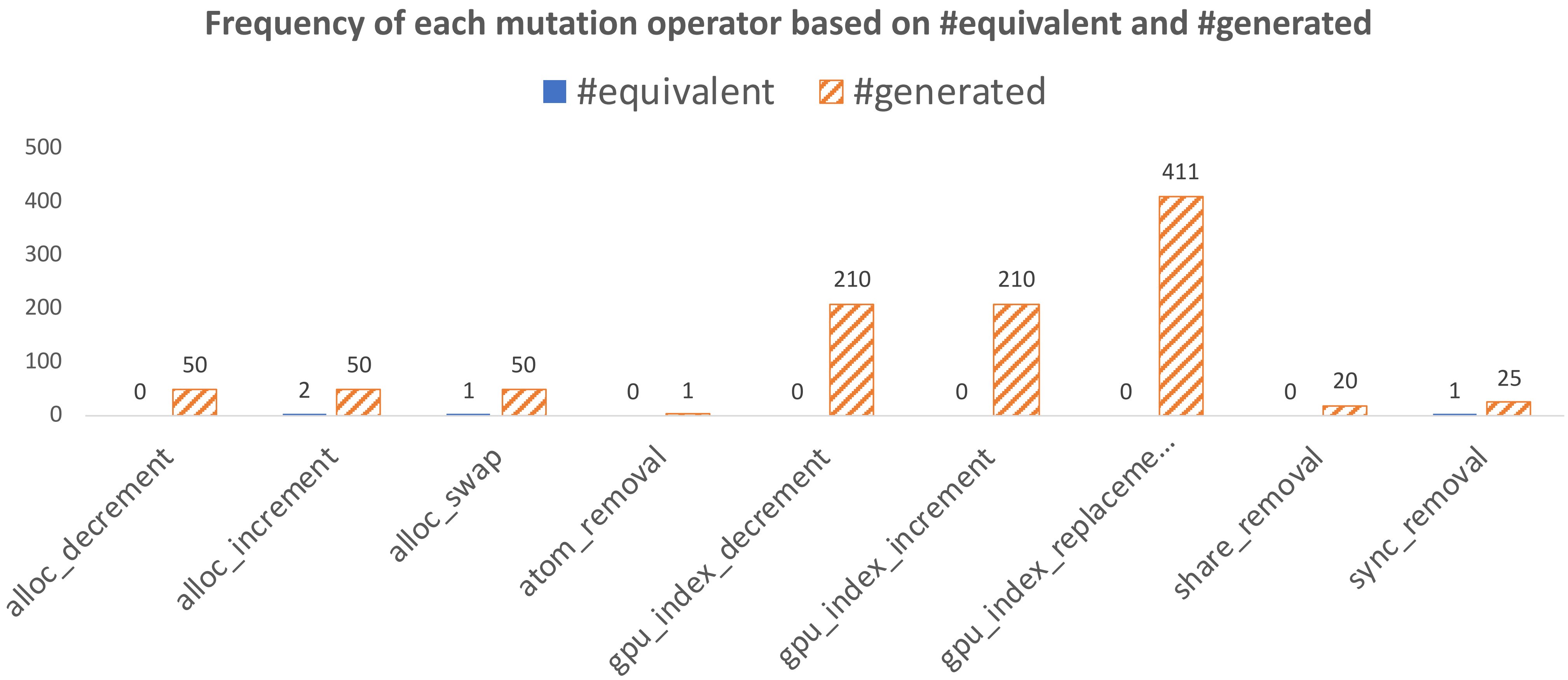
- #equivalent mutants
- #generated mutants
- mutation scores

Conventional vs. GPU-specific

- Effectiveness
- C-sufficient test suite

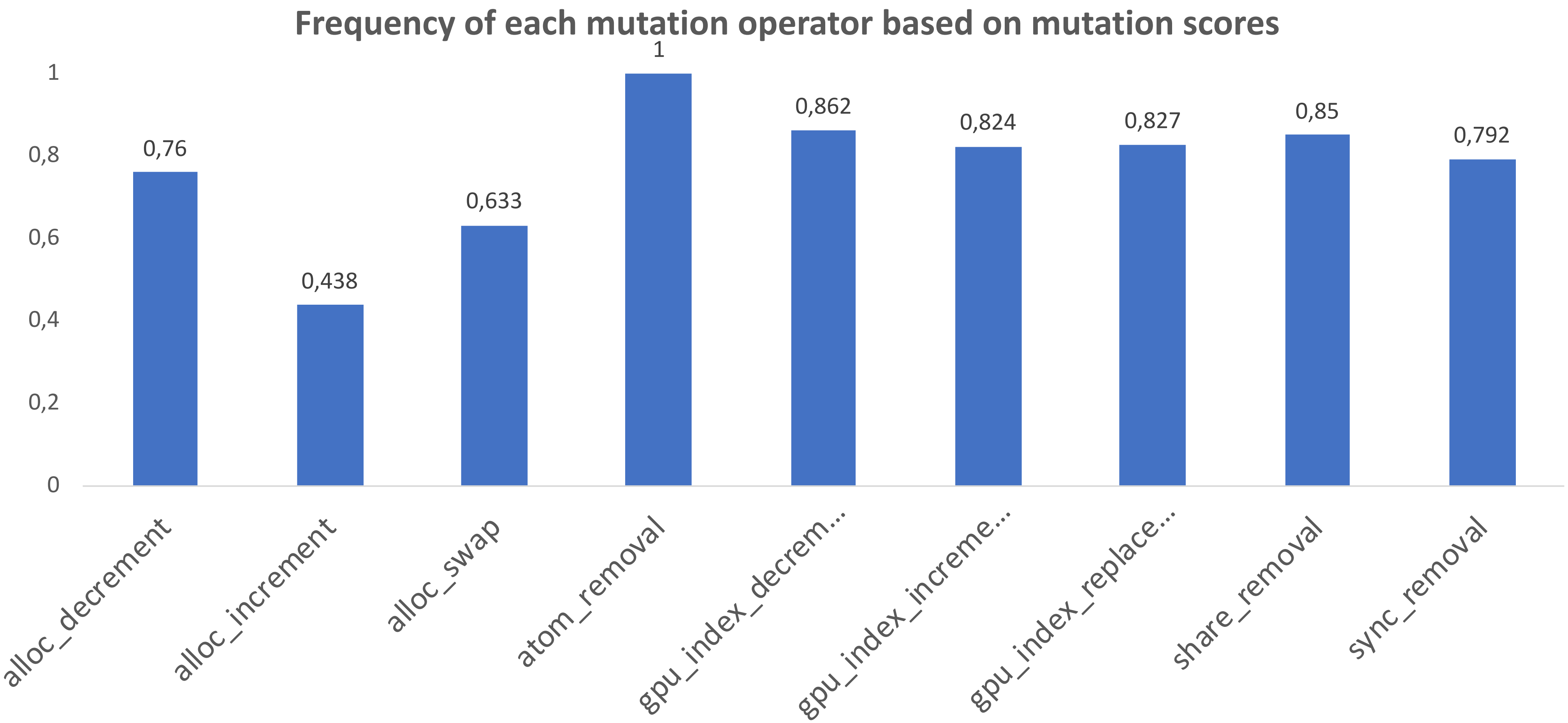
Results

- Initial evaluation of GPU-specific mutation operators



Results

- Initial evaluation of GPU-specific mutation operators



Results

- Conventional vs. GPU-specific
  - Manual analysis of surviving non-equivalent mutants

Conventional

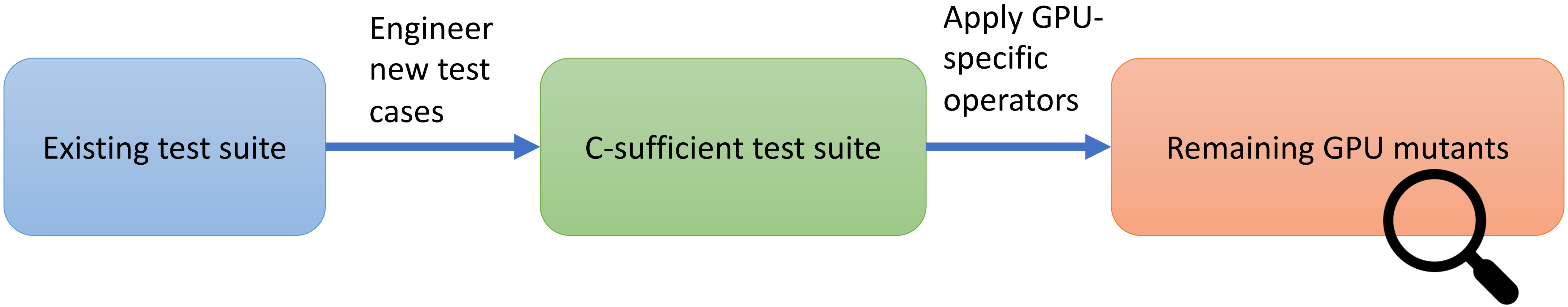
- Guide to write direct tests
- Effort to kill a mutant is within 1 min

GPU-specific

- Evaluate the test quality in the context of GPU programming
- Effort to kill a mutant is up to hours

Results

- Conventional vs. GPU-specific
    - C-sufficient test suite



Existing test suite

Engineer new test cases

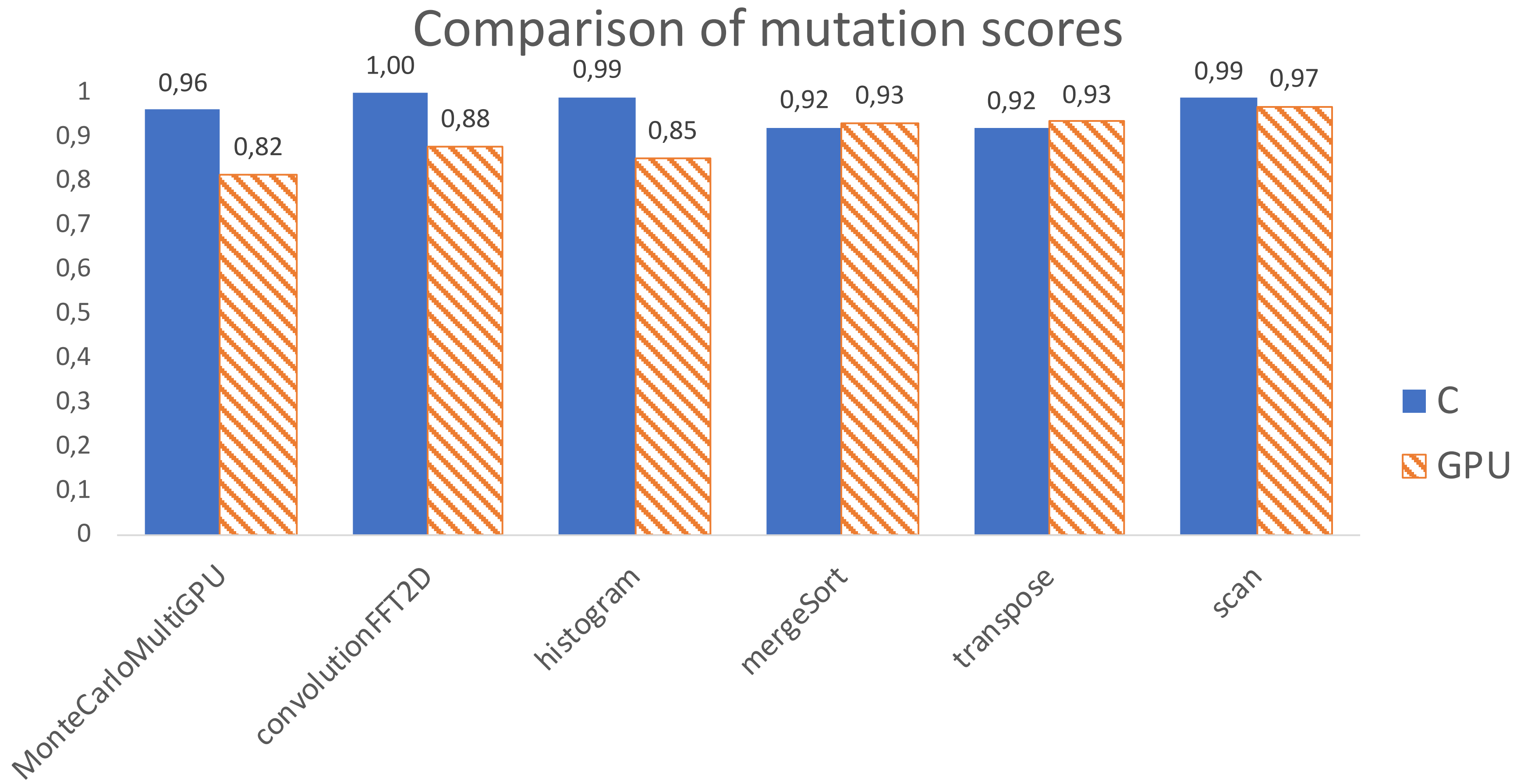
C-sufficient test suite

Apply GPU-specific operators

Remaining GPU mutants

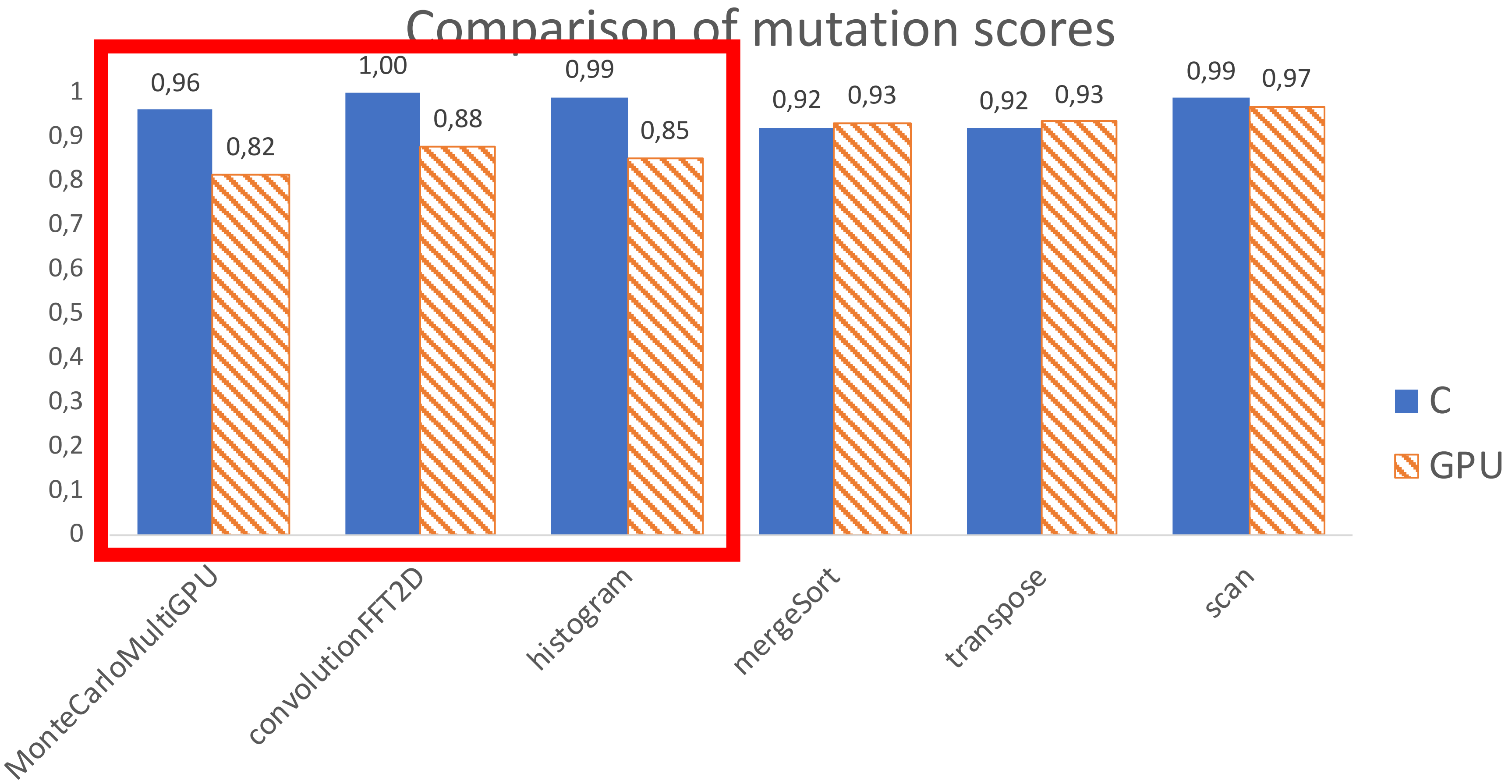
Results

- Conventional vs. GPU-specific
    - C-sufficient test suite



Results

- Conventional vs. GPU-specific
  - C-sufficient test suite



Summary

- What we have done:
  - 9 GPU-specific mutation operators
  - Comparison of conventional and GPU-specific mutation operators
  - A preliminary experiment on 6 GPU applications
- What we have learned:
  - Conventional operators: simple direct tests
  - GPU-specific operators: more delicate test cases (thus higher quality and more test effort)
  - equivalent mutants: bug detection