Memory   
 Out of memory   
 Illegal Memory Access/ allocation   
 Memory Copies  
 Memory Usage

Environment   
 Driver issues   
 Version Issues   
 Devices Issue/ Environmental Variable

Synchronization (Streams, threads)

Compiler   
 Make Files   
 NVCC compiler   
 Compile   
  
Math Errors  
 Math Functions  
 Floating point   
API function/ Flagging Error

Library   
 NCCL  
 Thrust   
   
Visual/ Graphics issues

Other

Fixes   
Reduce batch sizes, block size, increase memory, limit GPU memory,

Change GPU type, correct package installation, update drivers, update versions,

Use pinned memory, improve multi-stream/ GPU

Matching NVCC versions, correct NVCC path, build make files correctly,

Fix code, correct dimensions, correct NCCL initialization, boundary checking,

METHODOLOGY

1. Sample Collection

In order to collect samples of GPU programming related issues, we retrieve commit samples from GitHub repositories. First, we obtain a list of GitHub issues mentioning keywords ‘GPU’ and ‘GPU programming’ using the GitHub search API dated from January 2020 to December 2020 to identify 118,438 issues posted. From this set of issues, we download all of them for further evaluation. Next, we search the issue types for patterns related to ‘GPU\*’, ‘memory’, ‘threads’, ‘streams’, ‘environment’, ‘synchronization’, ‘concurrent’, ‘multi-GPU’ and ‘compile\*’ to find issues related to GPU programming. This step reduces the number of commits to 1,581. In order to confirm which of these issue links were related to GPU programming related, manual inspection was performed on the issues in the list. After manual inspection, and removing duplicate issues, the number of verified GPU programming bugs is 241.

1. Portion of GPU Programming bugs to other bugs
2. Sample Inspection

After collecting these issues of GPU Programming bugs from GitHub, we manually inspect the collected samples to identify information relevant to our research questions. In particular, we analyze the collected issue reports by inspecting the issues for following traits: the root cause of the bug, how the bug is manifested, and how the bug was fixed. For the issue reports, we inspect the developer comments and the linked commits. Through the inspection we obtained the sample set of 241 GPU programming bug issues.

1. Dataset Composition

Our dataset consists of a diverse of GPU programming bugs. The languages of the GPU programming bugs analyzed are CUDA, OpenCL, and GPU-accelerated computing with Julia, python.

IV. CAUSE OF FLAKINESS

We investigate the collected GPU Programming issues to determine the root cause of the bugs. We manually inspect the related issues of the test in order to locate the code or condition that caused the bug. We base our root cause categories as Memory, Environment, Synchronization, Compiler, Math Errors, API functions or flagging issues, Library, and visual or graphics issues. The categorization results are summarized in Table.

TABLE: Summary of Root Cause Categories Found.

|  |  |  |
| --- | --- | --- |
| Root Cause  Categories | Root Cause  Subcategories | Total |
| Memory | Out of memory  Illegal Memory Access/ allocation  Memory Copies Memory Usage | 35 23 10 10 |
| Environment | Driver issues  Version Issues  Devices Issue/ Environmental Variable | 8  27 20 |
| Synchronization | - | 37 |
| Compiler | Make Files  NVCC compiler  Compile | 5 11 7 |
| Math Errors | Math Functions Floating point | 8 5 |
| API function/ Flagging Error |  | 12 |
| Library | NCCL Thrust | 11 4 |
| Visual/ Graphics issues | - | 6 |
| Other | - | 2 |

1. Categorization

After manual inspection of the bug issues, we identify eight categories that root causes of bugs in these tests can fall under: (1) Memory issues, (2) Environment Issues, (3) Synchronization issues, (4) Compiler issues, (5) Math Error issues, (6) API function or Flagging issues, (7) Library issues, and (8) Visual or Graphics issues.

1. *Memory Issues:* We have found the root cause for a significant portion (33%) of GPU Programming bugs analyzed arise from issues related to the memory. The common cause for this issue is allocating memory that is larger than the available memory in the kernel.   
    Among these memory issues we, we identified four subcategories that group similar root causes together.
2. *Out of Memory issue:* GPU programming issues related to this category attempt to allocate the device memory that exceeds the available memory. The access to larger block sizes can lead this problem. An example is seen in the [2] when computing a Jacobian matrix leads to an out of memory issue because of the large block sizes.
3. *Illegal Memory Allocation issue:* GPU programming issues related to this category mostly occur during the runtime when the program access illegal memory. An example is seen in the [3] when the program tried to access the shared memory, however, it gives the error because of address is out of bounds. The code snippet in Figure shows that when executing the code works correctly, however, if we remove the keyword @inbounds in lines 5 and 7, the program gives an illegal memory access error at the address location 0xe7000004 is out of bounds.



V. MANIFESTATION

Reproducing bugs is a challenging task due to the non-deterministic behavior. If developers provide details on how the bugs were initially encountered and subsequently reproduced, this information provides possible strategies to apply for similar cases. We explore the strategies used by developers to manifest the underlying buggy behavior and construct categories for similar manifestation actions taken. Our categories are summarized in Table

TABLE V: Summary of Manifestation of Categories.

|  |  |
| --- | --- |
| Manifestation Category | Total |
| Unspecified | 78 |
| Given Error messages/logs | 55 |
| Provide Nvidia SMI | 5 |
| Provide Visual Profilers | 9 |
| Provide Code | 56 |
| Specify Problematic Platform/ Environment Condition | 38 |

Memory   
 Out of memory

1. "https://github.com/oreilly-japan/deep-learning-from-scratch-2/issues/16" (fix)
2. "https://github.com/PlasmaControl/DESC/issues/1"

Illegal Memory Access/ allocation

1. "https://github.com/JuliaGPU/CUDA.jl/issues/558"
2. "https://github.com/ptillet/torch-blocksparse/issues/15"
3. "https://github.com/microsoft/onnxruntime/issues/5555"

Open Issue but solve the problem:  
 "https://github.com/cupy/cupy/issues/3452"

Memory Copies

1. "https://github.com/rapidsai/dask-cuda/issues/438"
2. "https://github.com/JuliaGPU/CUDA.jl/issues/105"

Memory Usage   
  
8. "https://github.com/uber-research/LaneGCN/issues/2"  
9. "https://github.com/AlexeyAB/darknet/issues/6492"  
10. "https://github.com/codezonediitj/adaboost/issues/8"

Environment   
 Driver issues

11. "https://github.com/yshui/picom/issues/537"

12. "https://github.com/rsanchezgarc/deepEMhancer/issues/4"

Version Issues

13. "https://github.com/tmcdonell/cuda/issues/66"

14. "https://github.com/open-mmlab/mmdetection/issues/4012"

Devices Issue/ Environmental Variable

15. "https://github.com/PyTorchLightning/pytorch-lightning/issues/2420"

16. "https://github.com/JuliaGPU/CuArrays.jl/issues/589"

Synchronization

17. "https://github.com/NVIDIA/spark-rapids/issues/15"

18. "https://github.com/m4rs-mt/ILGPU/issues/222"

Compiler   
 Make Files

19. "https://github.com/NVIDIA/gvdb-voxels/issues/87"

NVCC compiler

20. "https://github.com/NVIDIA/cuda-samples/issues/44"

Compile

21. "https://github.com/ying09/TextFuseNet/issues/27"

Math Errors  
 Math Functions  
 "https://github.com/JuliaGPU/CUDA.jl/issues/71"  
  
 Floating point   
 "https://github.com/limbo018/DREAMPlace/issues/21"

API function/ Flagging Error   
"https://github.com/halide/Halide/issues/5443"  
"https://github.com/cupy/cupy/issues/578"

Library   
 NCCL  
 Thrust   
   
Visual/ Graphics issues

"https://github.com/runelite/runelite/issues/12777"

"https://github.com/wiremod/wire/issues/1998"