Formatting Submissions for a USENIX Conference: An (Incomplete) Example

Sabrina Brogren
University of Michigan

Christina Foshiem-Hoag
University of Michigan

Oliver Miles
University of Michigan

Abstract

Multithreaded programs suffer from nondeterministic behavior ranging from data races to uninitialized variables. GPU programs are especially vulnerable to bugs related to nondeterministic behavior due to their high level of paralellization. These bugs can be especially difficult to catch, as merely running the program again with the same inputs does not guarantee that the bug will rear its head again. Frameworks like PinPlay and Instant Replay allow a user to record and deterministically replay a program, but no tool yet exists with the same functionality for GPU programs.

Our goal is to create a tool that allows a user to capture the execution of their GPU program and replay the execution. The source of nondeterminism we're focusing on is data races. We want to be able to replay a GPU program such that not only do threads interacting with memory run in the same order, but also the state of memory throughout the program is identical to when it was recorded. We used NVBit (NVidia Binary Instrumentation Tool) to instrument instructions that operate on memory (loads and stores). On record, we record all loads and stores. On replay, we use locks to replay the loads and stores for each memory address in order. This paper will describe the describes the design of our tool, as well as discusses its usage on racey GPU programs. We hope our tool will serve as a proof of concept for further endeavors into GPU record and replay tools.

1 Introduction - 1 page

"Define and motivate the problem with examples, briefly talk about challenges. Explain your core high-level idea. Give an overview of your solution and summarize key results" Discuss other solutions (none exist). Talk about other kinds of nondeterminism like multiple kernals, uninitialized memory. Describe the core concept, which is to get all values when recording and replay them in order.

2 Design - 3 pages

Using NVBit and Cuda. What was instrumented and recorded? What challenges did we face? (ASLR, had to use a count to keep track of order when recording, had to implement our own locks, needed to manually replay loads and stores to make sure order is maintained) What other design decisions have we considered (replaying just by inserting data values, but not maintaining ordering. This gives us the same output from a program, but does not replay memory as well.)

3 Implementation - 0.5 page

"Explain subtle implementation details"

4 Evaluation

"Convince the reader that your system works and explain when it fails" 8-9 benchmarks discuss overhead (maybe look at overhead with only loads/only stores)

5 Related Work - 0.5-1 page

Pin-Play How is our work the same? How is it the same? What ideas were adopted from other works?

6 Discussion - 0.5 page

"Discuss your limitations and possible workarounds"

7 Conclusion - 2-3 paragraphs

"Summarize your paper and breifly point out the future work"

8 Footnotes, Verbatim, and Citations

Footnotes should be places after punctuation characters, without any spaces between said characters and footnotes, like so.[1] And some embedded literal code may look as follows.

```
int main(int argc, char *argv[])
{
    return 0;
}
```

Now we're going to cite somebody. Watch for the cite tag. Here it comes. Arpachi-Dusseau and Arpachi-Dusseau coauthored an excellent OS book, which is also really funny [**?**], and Waldspurger got into the SIGOPS hall-of-fame due to his seminal paper about resource management in the ESX hypervisor [**?**].

The tilde character (~) in the tex source means a non-breaking space. This way, your reference will always be attached to the word that preceded it, instead of going to the next line.

And the 'cite' package sorts your citations by their numerical order of the corresponding references at the end of the paper, ridding you from the need to notice that, e.g, "Waldspurger" appears after "Arpachi-Dusseau" when sorting references alphabetically [**?**, **?**].

It'd be nice and thoughtful of you to include a suitable link in each and every bibtex entry that you use in your submission, to allow reviewers (and other readers) to easily get to the cited work, as is done in all entries found in the References section of this document.

Now we're going take a look at Section 9, but not before observing that refs to sections and citations and such are colored and clickable in the PDF because of the packages we've included.

9 Floating Figures and Lists

Here's a typical reference to a floating figure: Figure **??**. Floats should usually be placed where latex wants then. Figure**??** is centered, and has a caption that instructs you to make sure that the size of the text within the figures that you use is as big as (or bigger than) the size of the text in the caption of the figures. Please do. Really.

In our case, we've explicitly drawn the figure inlined in latex, to allow this tex file to cleanly compile. But usually, your figures will reside in some file.pdf, and you'd include them in your document with, say, \includegraphics.

Lists are sometimes quite handy. If you want to itemize things, feel free:

fread a function that reads from a `stream` into the array `ptr` at most `nobj` objects of size `size`, returning returns the number of objects read.

Fred a person's name, e.g., there once was a dude named Fred who separated usenix.sty from this file to allow for easy inclusion.

The noindent at the start of this paragraph in its tex version makes it clear that it's a continuation of the preceding paragraph, as opposed to a new paragraph in its own right.

9.1 LaTeX-ing Your TeX File

People often use `pdflatex` these days for creating pdf-s from tex files via the shell. And `bibtex`, of course. Works for us.

Acknowledgments

The USENIX latex style is old and very tired, which is why there's no \acks command for you to use when acknowledging. Sorry.

Availability

USENIX program committees give extra points to submissions that are backed by artifacts that are publicly available. If you made your code or data available, it's worth mentioning this fact in a dedicated section.

[1] Remember that USENIX format stopped using endnotes and is now using regular footnotes.