Design and Develop a CUDA Engine for clang-repl

Proposal

Anubhab Ghosh
Indian Institute of Information Technology, Kalyani

Overview

Graphical Processing Units (GPUs) are a class of hardware in a computer system that produce graphical output for humans. Modern graphics rendering techniques increasingly use a lot of parallel computation for performance reasons that leads to modern GPUs having high parallel computation capacity. This gives rise to General-Purpose Compute on Graphics Processing Units (GPGPU) that expands the parallelization capabilities of the GPU to computation that do not necessarily involve graphics. This is highly useful for highly parallelizable workloads such as machine learning, processing of large amounts of scientific data, running simulation etc.

CUDA is a GPGPU platform and API targeted towards NVIDIA GPUs that gives access to compute elements of the GPU through standard programming languages like C++.

LLVM supports compilation of CUDA code. It has a distinction between host code and device code where host code runs on CPU and device code is compiled to NVPTX, the low level assembly language for NVIDIA GPUs. Only some functions marked as kernel functions are callable from host code and works as the interface between these two.[1]

HIP is a C++ API and "kernel language" for writing GPGPU code in a vendor independent manner. Currently it provides a thin abstraction over both NVIDIA and AMD GPUs.

Clang-repl is a C/C++ interpreter that is part of the LLVM project that can compile C++ code from user interactively just-in-time and execute them using the interactive compilation features of clang.

Cling is another C++ interpreter that is part of the ROOT project that is also based on LLVM and uses similar technologies. It is used for data analysis in high energy physics research. The next version of Cling will be based on clang-repl.[2]

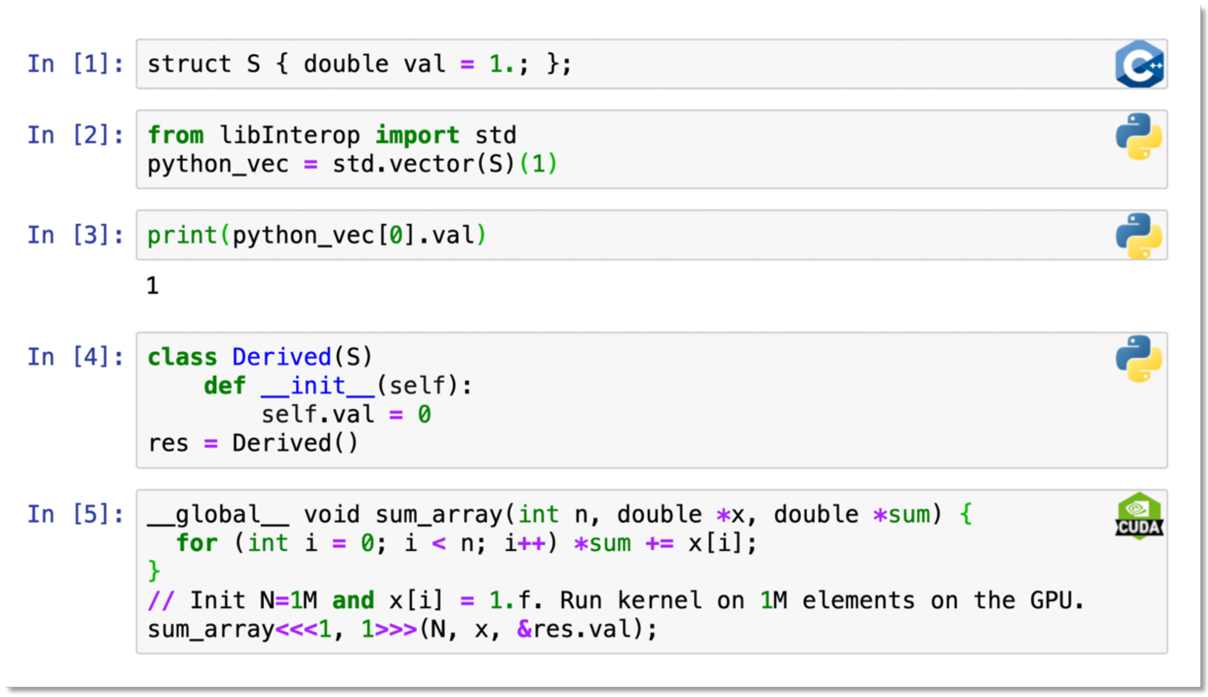
The goal of the project is to implement CUDA support for clang-repl. This would possibly require clang-repl to distinguish between host and device code and separately compile device code to PTX. Most of the functionality has already been implemented in Cling. It has to be generalized for the wider LLVM project and upstreamed into clang-repl. If time permits, the secondary goal of the project is to research the requirements and devise a plan for implementing HIP support in clang-repl.

[1] https://llvm.org/docs/NVPTXUsage.html
[2] https://root.cern/blog/cling-in-llvm

Project Motivation

The CUDA support in clang-repl will be useful for interpreting CUDA C++ code. Once this is achieved it will be integrated into xeus-clang-repl[3]. This is a Jupyter Notebook kernel that allows running C++ interactively. The long term goal is to enable CUDA C++ to be mixed with Python and regular C++.

Such implementation would enable a single ecosystem for HEP where physicists can mix Python, C++ and CUDA codes in the same environment. Upon project completion, it would be easier to use CUDA-based GPUs for data analysis in HEP.



```
In [1]: struct S { double val = 1.; };

In [2]: from libInterop import std
        python_vec = std.vector(S)(1)

In [3]: print(python_vec[0].val)

        1

In [4]: class Derived(S)
            def __init__(self):
                self.val = 0
        res = Derived()

In [5]: __global__ void sum_array(int n, double *x, double *sum) {
          for (int i = 0; i < n; i++) *sum += x[i];
        }
        // Init N=1M and x[i] = 1.f. Run kernel on 1M elements on the GPU.
        sum_array<<<1, 1>>>(N, x, &res.val);
```

Personal Motivation

Low level details of computer systems always fascinate me. I have been interested in system programming and compilers since my childhood. Over the years, I have also attempted to create my own toy operating systems and languages.

Last summer I worked on LLVM JITLink as part of Google Summer of Code 2022.[4] Clang-repl is a user of the library, so I believe the experience gained will be useful towards this project as well.

Finally, I love and fully believe in open source and contributing to open source software is always a pleasure for me.

[3] https://github.com/compiler-research/xeus-clang-repl
[4] https://summerofcode.withgoogle.com/programs/2022/projects/peDPYCpm

Implementation Details

CUDA code is a superset of C++ syntax and as such cannot be considered valid C++. The Nvidia CUDA Compiler (nvcc) splits the source code into device code and host code that respectively run on the GPU and CPU. They are passed through different compilation processes. The device code is compiled to PTX and embedded within the host code. The host code is appropriately modified to call CUDA runtime APIs and pass the device code. A similar approach needs to be taken for interactive C++ as well.

Clang already supports CUDA code compilation with `-x cuda` and related arguments. However this is not enabled in clang-repl. Support for this needs to be added by dynamically loading CUDA runtime libraries like `libcuda.so`.

Deliverables

- Generalize the CUDA backend implemented in Cling and write a detailed request for comment (RFC) document describing the design that is to be implemented in clang-repl.
- Gather feedback from the LLVM community and apply necessary changes to the design.
- Implement the backend as per the design in the RFC. Add proper tests for it.
- Write tutorials or blog posts describing how to use the new CUDA backend in clang-repl.
- Adopt the work in xeus-clang-repl which will bring us closer to the vision of a one-ecosystem being able to mix code between C++/Python and CUDA.

Stretch goal

If time permits, we plan to research the requirements supporting HIP in clang-repl.

Obligations

- I will dedicate 20 hours/week to work related to this project.
- I will maintain a document describing my current progress.
- I will provide regular updates to my mentors regarding my progress.

Timeline

1st December - 14th December	Study the relevant code that exists in Cling, especially around IncrementalCUDADeviceCompiler. **Deliverable:** More concrete plan of work presented at

	one of the relevant meetings.
15th December - 28th December	Iterate on various designs for the new backend with my mentor. **Deliverable:** Incorporated suggestions of my supervisor's comments in the work plan**.**
29th December - 11th January	Prepare an RFC describing the plan and design for the new CUDA backend for clang-repl. **Deliverable:** Submit it to the LLVM discussion forums to gather further input and direction.
12th January - 25th January	Gather feedback from the LLVM community through the RFC and iteratively improve the design. Simultaneously start implementation according to the current draft. **Deliverable**: A document describing the final design for the new backend.
26th January - 8th February	Primarily implement the new backend in clang-repl according to the design. This includes debugging, fixing problems and adding tests for the new code. Also focus on passing CUDA specific tests that already exist in Cling. At the same time continue to improve the design as needed based on new feedback or insights gained during development. **Deliverable**: CUDA backend implemented in clang-repl.
9th February - 22nd February	
23rd February - 8th March	
9th March - 22nd March	Buffer week for college exams.
23rd March - 5th April	Prepare a blog post with tutorials for using CUDA in clang-repl with the new backend. **Deliverable**: Tutorial article
6th April - 19th April	Research on requirements for providing an HIP backend in clang-repl.
20th April - 3rd May	Iterate on designs for a HIP backend for clang-repl and iteratively improve it by discussing with mentors. **Deliverable**: Prepare an RFC document describing the design to gather feedback from the LLVM community.
4th May - 17th May	Buffer week for any remaining work. Continue to improve the design based on feedback from the LLVM community or work on implementation.

Personal Details

Name: Anubhab Ghosh

University: Indian Institute of Information Technology, Kalyani

Email: anubhabghosh.me@gmail.com

Github: argentite

LinkedIN: https://www.linkedin.com/in/anubhab-ghosh-44b451194

LLVM Discord: argentite#0791

Time Zone: IST (UTC + 5:30)

Country: India