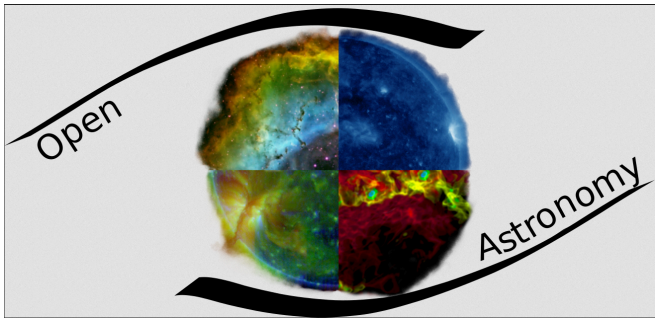
Project Proposal

Gnuastro - OpenCL integration with Gnuastro

Organisation : Open Astronomy
Mentors : Labib Asari, Mohammed Akhlagi



Contact Information:
Full Name: Warren Brandel Jacinto
Degree: Bachelor of Technology in Computer Engineering (B. Tech)
University: Veermata Jijabai Technological Institute (VJTI) , Mumbai
Location: Mumbai, Maharashtra ,India
Time Zone: Indian Standard Time (UTC+5.30)
Email: warrenjacinto@gmail.com, wbjacinto_b22@ce.vjti.ac.in
Github Profile: @DeadSpheroid
Resume: Resume

1.1 About Me: Introduction & Background

Hello! Warren Jacinto here. I am an undergraduate student in my second year at Veermata Jijabai Technological Institute (VJTI), Mumbai pursuing Computer Engineering as my major.

1. As for relevant courses in the institute, I have taken Data Interpretation and Analysis, Design and Analysis of Algorithms, Automata Theory, Operating Systems, and Database Management Systems.
2. I have been developing projects and participating in various competitions since my first year.
    a. Text-Style-Transfer: Changing the style of text while preserving its meaning. Link to its report.
    b. Myusik: Control the music you listen to with hand gestures.
3. I have good experience with **C/C++**, Python, Git/Github. **Linux** has been my daily driver for almost 2 years now and I am fairly comfortable with it.
4. In terms of hardware, I have an Intel i5 processor and an **Nvidia GTX 1650 GPU** for the purposes of this project.
5. I have been contributing to open source for a brief period of time. Here are my some of my open source contributions: sympy, Clad

1.2 Other Commitments

I can give 4 to 5 hours to the project everyday, compiling it to **30-35 hours per week** or even more than that if required.

I do not have any strict commitments to fulfill during this summer and can easily **increase my duration** if required. I am **flexible** with any working hours and can adjust them as per the time zone of my mentors.

I have my summer vacation from **8 June to 4 Aug** so a major part of the work will be easily completed in that duration. I will provide **regular updates** to my mentors regarding progress and will regularly attend the developer meets. I'll continue to post updates on my blog every **two weeks** for reference.

2.1 Project Description

The goal of this project is complete integration of OpenCL with Gnuastro, allowing for highly parallelizable operations to be executed on the GPU if available.

Gnuastro has many such parallelisable operations including convolutions, warps, interpolation that currently make use of the gnuastro thread library to execute on the CPU.

Why OpenCL?

OpenCL is an open standard framework supported by most GPU manufacturers like Nvidia, AMD, Intel and more. This makes it an ideal candidate for the task at hand, as we would like for users to be able to use Gnuastro with whatever hardware they have, not limiting them to a specific manufacturer.

OpenCL supports both offline and online compilation. That is, one can either load the kernel source code and compile it at runtime, or precompile it before and load the binary at runtime.

For kernels provided by Gnuastro, offline compilation would be a better choice considering the time taken to compile at runtime.
However, we should also support runtime compilation for the user kernels.

Additionally unlike CUDA or other frameworks, OpenCL can target the CPU threads as well, eliminating the need for separate CPU/GPU functions, greatly reducing code complexity.

2.2 Deliverables

- ➔ **A robust low level wrapper infrastructure to handle OpenCL functionality within gnuastro, replacing the current pthreads based wrapper.**
- ➔ **OpenCL kernels for gnuastro core programs that work on both CPU and GPU.**

- ➔ **Scripts for testing and benchmarking the new kernels and wrappers.**
- ➔ **Thorough documentation and command line wrappers allowing users to choose between CPU and GPU execution.**

2.3 Implementation: Plan of action

Basic Goal Of Project :

The ultimate goal is to leverage the power of GPUs to speed up execution times for the gnuastro library.
Per the report of GSoC '23, a barebones implementation of OpenCL is already complete. So, the first step is **understanding the existing integration of OpenCL** and the work that is already done.

Build System Integration :

OpenCL offers two ways of compiling kernels:

1. **Just In Time**, compiled at runtime of the main program
2. **Precompiled**, loaded at runtime of the main program

Although providing precompiled kernels while shipping gnuastro would lessen the burden on users, it is important to note that these binaries are **device specific,** and there is **no guarantee** they will work on the users system.

Of course, we could provide **compiled kernels** for most of the popular devices, but what about Just In Time?
Just In Time compilation means the OpenCL kernels are compiled **at runtime**, which adds **unnecessary complexity and lag** to gnuastro programs.

Instead, a better way to achieve this goal would be including OpenCL compilation as **part of the build system**, compiling the kernel source code when the user is **building the project** and storing the binaries in a known location to access later.
Another perhaps better way to handle this is to compile the kernel when the program is first run, save it(cache), and later reuse this **same** kernel. This avoids any unnecessary modification to the build system.

Wrapper infrastructure

The first step toward the goal of this project is setting up the wrappers that will load and execute OpenCL kernels. For this purpose, a new gnuastro module "**cl_utils**" is needed, the idea for which came from the '**gpu_utils**' in the '23 report.

This module would function as the **low level wrapper** allowing developers to write OpenCL kernels.

A basic structure of the module would contain **functions** for:

1. Querying available devices
2. Creating a global context, command queue
3. Loading/Compiling kernels
4. Enqueuing these kernels for execution
5. Transferring gnuastro data to and from the GPU

This wrapper would aim to replace the current pthreads wrapper utilised by gnuastro with a **single unified module** capable of executing workers on both CPU and GPU threads. OpenCL makes it easy to execute workers on CPU threads in a variety of ways, even going as far as providing a way to **prevent memory duplication** by transfer of data, should code be executed on the CPU.

By the end of this, we would be able to call OpenCL kernels from any part of the program and **execute them** on either the CPU or the GPU.
After this is complete, we would be ready to start writing kernels for Gnuastro programs.

Writing kernels

Currently, Gnuastro utilises a **pthreads based wrapper** to leverage CPU parallelisation for certain operations:

1. Convolution in convolve.c
2. Reading .fits columns in fits.c
3. Interpolation in interpolate.c
4. Dimension collapsing in dimension.c

and many **lower level internal programs** like table reading in table.c, outlier removal in tile-internal.c and more.
These would be rewritten in the form of OpenCL kernels one by one and smoothly integrated with the higher level programs(astconvolve, astcrop, etc) starting with the more intensive **convolution** and **interpolation** operations.

Choosing between CPU and GPU

The gnuastro library functions(for eg, gal_convolve_spatial) would have an **extra argument** provided by the user indicating whether the operation should be executed on the **CPU or the GPU(if available)**.
The core programs provided by gnuastro(for eg, astconvolve) would include a **--gpu flag** on the command line to indicate which device should be used

User Defined Kernels

In some cases, the user may want to use **their own kernels** within their gnuastro program. Therefore, as a secondary goal, we will make the **cl_utils** functions available for the user to use for their own kernels as well.
In this case, users would write their own kernels, and using the helper functions, they would execute it.

Benchmarks

After implementation, to verify whether work is done correctly, benchmarks must be introduced that compare the execution speed of code on the **GPU versus the CPU.**
This would be crucial to understanding which modules benefit the most from **GPU speedup** and if any **changes are needed** in the kernels.

2.4 Project Timelines

Up Till May 1	**Proposal accepted or rejected** ➔ Familiarise myself with the codebase and existing work on GPU integration

	➔ Fix a few issues to get a feel for working with the codebase ➔ Discuss with the mentor about the project details ➔ Learning the OpenCL C API.
May 1 - May 28	**Community Bonding Period** ➔ Ensure Gnuastro is properly setup for development on my machine ➔ Develop a complete plan for the integration and make note of all the kernels to be written. ➔ I will be having my End Semester Exams from May 20 to June 7, so although progress may be slow, the work will be completed in time
Week 1 & 2	**Coding officially begins!** ➔ I will start by Implementing the cl_utils module allowing OpenCL kernels to be used within gnuastro programs. ➔ Then, I will work on ensuring that the functions provided by cl_utils are integrated with gnuastro on the whole ➔ I will discuss with my mentors about which kernels should be written, and their priority
Week 3 & 4	➔ After the completion of cl_utils, I will write test scripts and documentation for the cl_utils module. ➔ Then, I will start writing OpenCL kernels with the first most likely being convolution.h ➔ Alongside writing the kernel, I will also work on a general benchmark script to compare the execution speeds of CPU and GPU for the various kernels I will be writing.
Week 5	**Buffer Period** ➔ Try to complete any task mentioned before if not completed or if any unforeseen situation arises ➔ Coordinate with mentors, reinforcing the tasks to be done ahead.
Week 6 & 7	➔ I will continue implementing the kernels as discussed with my mentors.

	➔ I will also work on the command line wrappers, allowing users to simply pass a --gpu flag to choose the device for execution when using gnuastro's provided programs(say astconvolve)
Weeks 8 & 10	➔ I will complete any remaining kernels. ➔ Begin replacing the pthreads library wrappers used throughout gnuastro with the new cl_utils wrapper. ➔ Ensure smooth integration of the kernels with the wrappers.
Week 10 & 11	**Buffer Period** ➔ Complete any remaining tasks, fix any bugs that may arise previously. ➔ Try to fulfil all necessary requirements of mentors regarding the project.
Until Pens Down	➔ Work on documentation and blog posts about my work. ➔ Implement caching of compiled kernels for the core kernels provided by gnuastro, to save time in future uses of the kernel. ➔ Work on addressing the data transfer overhead described in the '23 Report.
Post GSoC	➔ Continue to develop and improve the new GPU support, and contribute to the project in general.

3. Other Comments

Low Level Systems Programming has always been one of favourite domains of Computer Engineering, because it explains the "magic" behind a lot of the developer tools that we take for granted. This project represents a chance for me to contribute to a real world application used by many.

By participating in this project, I hope to learn a lot from my mentors, and from the gnuastro community as a whole. I would love to continue contributing to gnuastro even after GSoC '24 continually improving and adding new features to the project.

4. References

- GSoC ‘23 report by Labib Asari
- Learning OpenCL
- Gnuastro source
- Cuda Integration of gnuastro