Accelerating Automated Extraction of Radio Astronomical Sources from Overseation Data with GPU Accelerators , University of Cape Town

YASEEN HAMDULAY, JARRED DE BEER

1. PROJECT DESCRIPTION

Radio astronomy surveys are blind surveys of the sky for detecting neutral hydrogen emission from galaxies. The process of detection is known as source finding and has traditionally been conducted by Astrophysicists who manually inspect the resulting data to identify sources of hydrogen emissions.

Future radio astronomy surveys, such as MeerKAT (Meer Karoo Array Telescope) and ASKAP (Australian Square Kilometer Array Pathfinder), are going to be orders of magnitude larger than traditional surveys and manual inspection of the data will no longer be viable. Automating the source finding process has become a necessity and various software packages have been developed to integrate this process. However, they are still too slow and research into accelerating them has revealed key bottlenecks in their performance. In this project we identify algorithms from two such bottlenecks and attempt to accelerate them with CUDA (Compute Unified Device Architecture), a parallel programming model integrated into the GPUs (Graphics Processing Unit) of a range of NVIDIA graphics cards.

The GPU is a highly parallel, low-cost computational alternative to the CPU with a considerably lower energy footprint. A GPU is comprised of streaming multiprocessors which each contain dedicated arithmetic units and a variety of memory hierarchies. They exploit data level parallelism by implementing SIMD (Single Instruction Multiple Data) and MIMD (Multiple Instruction Multiple Data) performed by blocks of threads. Threads are lightweight and designed to incur minimal time penalties for context switching, which helps maximise thread occupancy. [1] Fluke et al report that in particular cases GPU implementations of single-threaded CPU algorithms perform ten to a hundred times faster. The most performant solutions, however, can be particularly difficult to achieve due to the unique nature of GPU programming.

The algorithms we have identified as bottlenecks are the 2D-1D wavelet reconstruction and the S+C (Source and Clip) source finder. 2D-1D wavelet reconstruction is a smoothing process to remove noise from data, prior to source detection. This smoothing process is depended upon by intensity threshold source finding algorithms and consumes up to 75detect sources and is one of two source finders integrated into the SoFiA package. SoFiA is a source finding package which is designed to be flexible and used in various types of surveys at ASKAP. In addition to implementing S+C, SoFiA has also integrated the 2D-1D wavelet reconstruction algorithm into the filtering stage of its pipeline. The package is designed to be modular which allows algorithms to be integrated by the community to further enhance its use as a general source finder package. SoFiA is released under the GPLv2 license and is available on Github. The 2D-1D wavelet reconstruction algorithm is also used in the filtering process of the DUCHAMP package. DUCHAMP is an intensity based source finder which has the highest reliability among a range of source finders. The DUCHAMP package is written in C and released under the GPLv3 license.

We aim to make automated source finding more viable for the future larger radio astronomy surveys. This is of particular benefit to the MeerKAT and ASKAP projects, as well as developers and Astronomers making use of these packages for source finding.

2. PROBLEM STATEMENT

Computational bottlenecks in source finding algorithms are a prohibitive factor for future larger radio astronomy surveys. The research questions attempt to identify the degree to which these bottlenecks can be mitigated.

1. Can a CUDA implementation of the 2D-1D wavelet reconstruction algorithm accelerate the DUCHAMP source finding process, and how much speedup can be obtained? 2. Can a CUDA implementation of the S+C source finding algorithm accelerate the SoFiA source finding process, and how much speedup can be obtained?

Successful speedups of these algorithms will increase the throughput of the DUCHAMP and SoFiA packages, which will be used by Astronomers when conducting source finding. In particular SoFiA is intended to be used in ASKAP and will benefit the various surveys which have already been scheduled there. These speedups should also have an impact on surveys conducted at MeerKAT.

3. PROCEDURES AND METHODS

The 2D-1D wavelet reconstruction algorithm written in C in the DUCHAMP package and the S+C algorithm written in Python from the SoFiA package will both be re-implemented in C to run in serial. They will then be integrated back into local versions of their respective packages and run with test datacubes to ensure their correctness. This process will help us to develop an understanding of the intricacies within the algorithms as well as to establish a baseline level for performance comparisons with later implementations.

A naive and unoptimised version of both algorithms will then be implemented in CUDA for GPU processing. The naive implementation will prevent wasted effort on highly specific optimisations and the resulting CUDA kernel will provide proof of the feasibility of the project. The output from the naive implementation will be tested against the original package to ensure it is obtaining the same results. Correctness of the algorithm is critical at this stage prior to doing optimisation.

An optimised version of the CUDA kernel will then be implemented. Various optimisation techniques will be explored, the most difficult of which is predicted to be optimal use of the GPUs memory hierarchy. In addition to this the optimisation of block sizes for latency hiding and maximised thread occupancy will be considered. A high thread occupancy ensures that the streaming multiprocessors are being used to their full capacity. Another optimisation is the asynchronous streaming of kernels which allows data to be transferred from the host onto the GPU device while it is processing. This helps mitigate time wastage when transferring data from the host, which is done over the PCI express bus and is an unavoidable bottleneck with bandwidth of up to only 16GB/s.

Writing highly parallel code correctly is known to be difficult and is highly prone to bugs[2]. The most common issues encountered are data races in which two concurrent threads read or write from the same position in memory. This problem is exacerbated by the GPU which can concurrently run thousands of threads. Data races can go undetected for a long time and cause non-deterministic, incorrect output. Ideally each thread will contain its own independent copy of the input data, however, Westerlunds[3] acceleration of the Parallel Gaussian Source Finder indicates that shared data is unavoidable.

Source finders are measured by two metrics: reliability and completeness. Reliability is the ratio of the number of true positive detections over the total number of detections. Completeness is the ratio of the number of true positive detections over the total number of sources contained within the datacube. The CUDA implementations of the 2D-1D wavelet reconstruction and S+C source finding algorithm should not affect the reliability and completeness of the original packages. We are interested purely in accelerating the respective algorithms and will take effort in ensuring that the behaviour, developed by Astrophysicists, remains unchanged.

The success of the project will therefore be measured by the acceleration achieved. This will by analysed by comparing the runtime of the original package with the same package running the optimised CUDA implementation, taken with datacubes of increasing size.

A machine with a CUDA enabled device is required to run and profile the algorithms during development. There are various technical difficulties with respect to this. The first difficulty is that CUDA enabled devices are expensive and may not be available in home environments. Fortunately, CUDA enabled workstations are accessible at UCT and even accessible via remote login. Power outages will affect these workstations, however, and is likely to be an issue. To mitigate this risk of downtime, access can be obtained to the UCT cluster which is powered by generators. Another option is to utilise Amazon Web Services which offers GPU computing services at affordable prices. Nvidia provides a profiling tool to profile the devices performance. It runs on Linux and windows machines, and will be crucial during evaluation and optimisation stages.

4. ETHICAL, PROFESSIONAL AND LEGAL ISSUES

SoFiA is GPL version 3.0 and DUCHAMP is GPL version 2.0. Verification for releasing the CUDA implementations back into the original packages will need to be obtained from the NRF, who are funding the project.

UCTs IP policy states that UCT holds copyright on the software produced in the capacity of the project, and all produced papers must be published under a Creative Commons license.

5. RELATED WORK

Various research has been made into accelerating source finders and work has been done to develop source finding packages for use by Astrophysicists. An example software package is SoFiA which is designed for the modular inclusion of source finding algorithms and to be used by various surveys conducted at ASKAP. Research into acceleration has been conducted by Badenhorst et al who implemented a multithreaded CPU version of the ATrous wavelet reconstruction algorithm, which is used in DUCHAMP. Further acceleration research has been conducted by Westerlund et al who implemented and compared GPU speedups of the PGSF (Parallel Gaussian Source Finder).

Westerlund et als GPU version of PGSF was conducted on a GPU cluster and they found that the filtering process consumed 70

The CUDA implementation efforts of this project are closely related to Westerlund et als GPU accelerated Parallel Gaussian source finder. The acceleration of the 2D-1D wavelet reconstruction algorithm is related to Badenhorsts[a] multithreaded ATrous implementation. ATrous is a wavelet deconstruction algorithm that is part of the 2D-1D wavelet reconstruction process. The CUDA implementation of the S+C algorithm is related to the SoFiA package as it will be be based on the source code in that repository and integrated back into it.

6. ANTICIPATED OUTCOMES

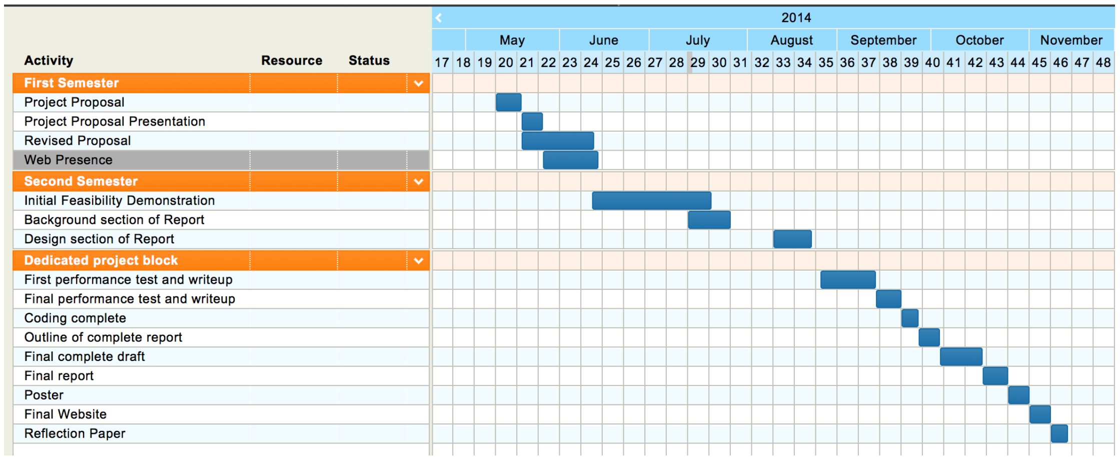
The acceleration of the 2D-1D wavelet reconstruction and S+C algorithms address performance bottlenecks in the overall processing of the DUCHAMP and SoFiA source finding packages. We anticipate a considerable speedup in the overall runtime which will increase the viability of next generation interferometers such as MeerKAT and ASKAP. Quicker performance of the source finding process will facilitate an increased

Risk	Mitigation
Data format of semi-realistic data cubes is incompatible with SoFiA/DUCHAMP packages.	Find a converter and covert it into the FITS data format.
Inability to get access to HIPASS data cubes.	Email Paolo Serra or Popping to ask for the datacubes used in their tests.
The algorithm is memory bound to the point where it cannot be sped up over the CPU implementation.	Research and implement overlapping kernel execution to minimise time wastage in data transfer.
Integrating CUDA implementations back into the respective packages source code becomes infeasible.	Implement standalone versions of the source finders which can read input and output catalogues and is package independent.
GPU programming is difficult and we may not have sufficient experience to implement a CUDA version which is considerably faster than the CPU implementation.	Consult Chris Laidler and NVIDIA papers on similar topics, such as NVIDIAs paper on convolution.
The part of the pipeline we choose to implement turns out not to be the bottleneck.	Choose algorithms which have been identified from reliable sources as being bottlenecks, such as from email conversation with Paolo serra.
A team member drops out of the course.	Design the implementation strategy such that each member has a similar procedure but the code, framework and implementation are entirely independent, and can be tested, compiled and integrated individually.
Processing takes many hours to complete and are interrupted by power outages.	Make use of UCTs cluster which runs on generators and remains online during outages.
GPU Kernels take too long to process and are interrupted by the Watchdog timer.	Divide the datacubes into smaller subcubes, or write smaller CUDA kernels.

number of runs over the data and increase the reliability of the results. The CUDA implementations are expected to have a much lower energy footprint[b], enhancing the viability and scalability of these projects.

A bonus outcome is for the CUDA implementations to be successfully integrated back into their original packages and adopted by Astronomers when conducting source finding.

7. PROJECT PLAN



The following resources will be required:

— **Generated data cubes**
These will be used to test the reliability and completeness of the CUDA implementations and to compare output with the original package.

— **Access to the GPU cluster**
The GPU cluster at UCT is necessary to process large data cubes without blocking local use of personal computers and mitigate the effects of power outages.

— **HIPASS Parkes All Sky Survey data cubes as well as the catalogues for testing on real data and comparing reliability**
HIPASS survey data cubes are required to test the performance of our accelerated source finder on real data in addition to generated data.

— **DUCHAMP and SoFiA source code**
The CUDA implementations will be based on code in the DUCHAMP and SoFiA packages and integrated back into their respective repositories.

— **Workstations with CUDA enabled devices**
CUDA runs on a range of NVIDIA graphics cards and it is a necessity to be able to run the implementations on a workstation with a CUDA enabled device.

A final report will be provided illustrating the comparison of the CUDA implementations overall processing time to that of the original package. These will be taken with respect to datacubes of varying sizes. An accompanying project poster and website will also be provided.

The following milestones relate to the sequence of work required to implement the CUDA version into the package. The higher level honors project milestones are provided in a gantt chart.

(1) Check out and build the package source code. We will have the source code checked out locally and in a buildable state. This is necessary in order to run the package with input.
(2) Obtain FITS datacube and run it through the package. This will be the first time the package has been run with input and produce output. We will need to understand the input parameters of the package in order to do so.
(3) Implement a single threaded version of our algorithm in C. This will require careful consideration into how the algorithm will be integrated and used by the package, and we will strive for minimal friction and maximal transparency to the user.

(4) Run performance tests with the single threaded implementation. These will be used as a base benchmark for the CUDA implementation.
(5) Naive GPU implementation. Have implemented a running, straightforward CUDA kernel, aiming for correctness and simplicity.
(6) Run performance tests on the naive GPU implementation. Use Nvidias profiling tools to analyse the CUDA implementation and gain insight to further optimise the CUDA kernel.
(7) Optimised GPU implementation. Have found grid sizes that maximise occupancy levels; local variables, data representation and use of memory hierarchies for optimal memory access times; and overlapping kernel execution to maximise on data transfer.
(8) Run performance tests on the optimised GPU implementation, to measure and profile the speedups over the naive implementation. We will likely iterate multiple times between this stage and the previous stage.
(9) Generate graphs to illustrate the running times of the optimised GPU implementation compared with the original package, over varying datacube sizes.
(10) Integrate code back into the packages original source code repositories, or have approached the packages owners with permission to do so.

Work has been divided into two independent projects designed to run in parallel with minimal dependencies. This mitigates the risk of a member dropping their project and allows the second project to continue unhampered. The two are essentially identical but each extends a different source code repository and attempts to accelerate aspects of different source finding algorithms. Each can read the same input data cubes, generate the same output and analyse the performance independently from the other.

Yaseen will accelerate 2D-1D wavelet reconstruction using CUDA and contribute towards the DUCHAMP package, released under GPLv3. Jarred will accelerate the S+C source finder using CUDA and contribute towards the SoFiA package, released under GPLv2.

Fig1 illustrates the work division between both Yaseen and Jarred, and indicates the stages that each will undergo. The stages are almost identical yet highly independent between the two projects.

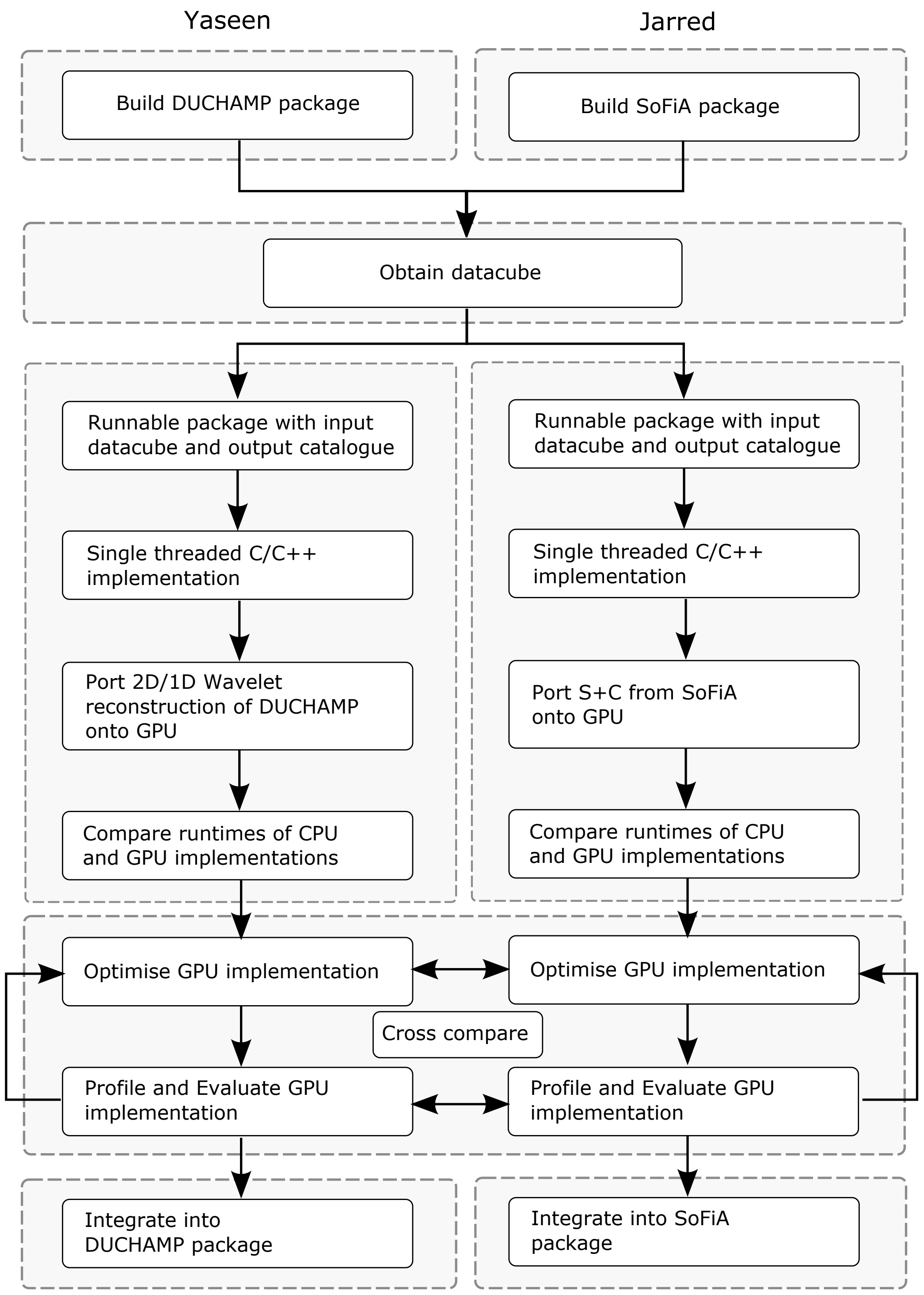

Yaseen
Jarred
Build DUCHAMP package
Build SoFiA package
Obtain datacube
Runnable package with input datacube and output catalogue
Runnable package with input datacube and output catalogue
Single threaded C/C++ implementation
Single threaded C/C++ implementation
Port 2D/1D Wavelet reconstruction of DUCHAMP onto GPU
Port S+C from SoFiA onto GPU
Compare runtimes of CPU and GPU implementations
Compare runtimes of CPU and GPU implementations
Optimise GPU implementation
Optimise GPU implementation
Cross compare
Profile and Evaluate GPU implementation
Profile and Evaluate GPU implementation
Integrate into DUCHAMP package
Integrate into SoFiA package

Fig. 1. Work division between Yaseen and Jarred