THE UNIVERSITY OF MELBOURNE

Performance Improvements with GPUs for Marine Biodiversity

Lev Lafayette

Senior HPC Support and Training Officer
eResearchAustralasia, Melbourne, 2018

Marine Spatial Ecology with Job Arrays

Research in the marine biodiversity and population connectivity, which has significant implications for the design of marine protected areas. In particular there is a lack of quantitative methods to incorporate, for example, larval dispersal via ocean currents, population persistence, impact on fisheries etc.

Probable dispersal routes and for marine populations is a data and processing intensive task of which traditional high performance computing systems are especially suitable, even for single-threaded applications. Whilst there were some data dependencies a large decomposed component of the workflow could operate in a data-parallel manner.

A large level of independence between sets allows for use of job arrays to significantly improve processing time. This was initially carried out by the University of Melbourne's general purpose High Performance Computer system, “Edward”.

This sort of capacity utilisation was common on the system, leading to the development of the highly cost-effective “Spartan” system which was optimised for job throughput over raw performance.

GPGPU Processing Investigation

Despite performance improvements the sheer number of datasets was still significantly greater than the capacity of the system. The possibility of further optimisation utilising GPGPUs was investigated.

The architecture of GPUs is typically with a significantly lower clock speed than typical Central Processing Units (CPUs), providing limited parallelisation of operational functions on a datastream (e.g., map, reduce etc) and especially particular computational problems (e.g., matrices and vectors) in a manner that broadly fits to SIMD (single instruction stream, multiple data stream) architecture, making them particularly well suited for pleasingly parallel problems.

GPGPUs required object code to be compiled for the GPU (e.g., using OpenCL or nvcc). There is no shared memory between the GPU and CPU and any unprocessed data must be transferred to the GPGPU environment and then back to the CPU environment when completed. This said, GPUs typically only have small amounts of cached memory. Replacing the need with GPU pipelining and ensuring very high memory transfer between the GPU and the host.

GPGPU Processing Capacity

Neither the Edward nor Spartan systems at the University of Melbourne had significant GPGPU capability - but this would change!

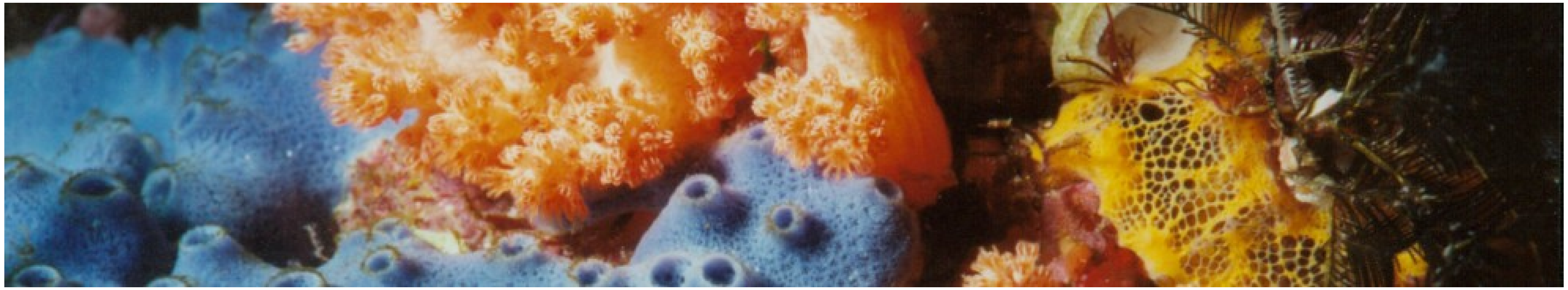
The "Edward" system included two nodes with GPUs attached, both being dual quad-core cpus with 2 nVidia 2070 GPUs with 8GB of memory. These saw minimal usage on "Edward" and as a result a significant expansion was not initially developed for the "Spartan" system.

A small 3-node partition for general availability was implemented on "Spartan" in 2016, each with 12 cores, 251 GB of RAM, and 4 NVidia K80 GPUs. An additional 2-node partition with the same specifications was established for the ARC Centre of Excellence for Particle Physics at the Terascale (CoEPP).

Nyriad's Review

During the first half of 2017 Nyriad reviewed the HPC infrastructure, existing MATLAB(R) source code and sample data, and wrote a test suite designed to run the CPU and GPU versions at the same time.

The goal of the approach was to check for equivalence of the algorithms without taking away the researcher's ability to evolve the code, providing a 'sandbox' environment that could enable running simulations on a larger scale. There were two review stages; the first for optimisation of the existing MATLAB (R) code base, followed by identification of functions that could be distribution and rewritten for GPUs.



Code Optimisation Part I

Three sections were identified in the code as computationally intensive, Del2, Cell Wall Flux Calculation, and Anti-diffusion Velocity Calculation. The Del2 function was a built-in MATLAB(R) function, that required data padding to avoid edge effects and also used 3D arrays, which was unnecessary for the simulation.

For the Cell Wall Flux Calculation, pre-compilation of variables (UL, UR, VB, and VT) was removed to in-line calculation instead. For the Anti-diffusion Velocity Calculation function dCdx calculations were only done on cells that had non-zero larvae densities. This was with a multiplication by a logical matrix where cells were 1 if larvae were present and 0 otherwise. Indiscriminately computing all the values and removing unwanted ones with the logical multiply removed the overhead of selecting values and conditionally computing them. The expressions for dCdx calculations were also moved inline with the Vd calculations.

Code Optimisation Part II

For the GPU optimisation, the DisperseLarvae and RunSimulation functions were added to the GPU acceleration to the code base. The three main operations of the simulation which account for the bulk of the execution time have been written into optimised GPU functions in CUDA C++, or "kernels" in CUDA terminology. Functions are provided to support both double and single precision calculations. MATLAB(R) uses a gpuArray type to to represent matrices in GPU memory. There are built-in functions in MATLAB(R) which the user can treat as normal matrices, where operations are carried out on the GPU using standard MATLAB(R) code.

Whilst convenient, it is not efficient; specialised GPU code has to be added in if statements that use the USEGPU flag. As the CPU and GPU versions share code, maintaining the bulk of the GPU code is made simpler. Instead the GPU code works by moving several matrices to the GPU enabling MATLAB perform operations and matrix creations of the GPU when a gpuArray is used.

Performance Improvements

Nyriad code review identified bottlenecks that were available for GPGPU workloads. On the University of Melbourne HPC system, "Spartan", using a single GPU, a 90x performance improvement was achieved over the original code and a 3.75x improvement over the CPU version with 12 threads available for the 4.6 GB Atlantic Model simulating 442 reefs. The simulation, previously taking 8 days to complete on one of the most powerful nodes (i.e. GPU or physical), could be completed in 2 hours. On the other hand, for the 4 MB South Africa Benguela Region dataset the GPU version is faster than the original code, but slower than the improved CPU implementation.

South Africa Benguela Region

Threads	Original	Improved	GPU
1	28s	13s	14s
12	26s	11s	14s

Single 10 day simulation

Atlantic Model

Threads	Original	Improved	GPU
1	258h*	21h*	2h
12	180h*	7.5h	2h

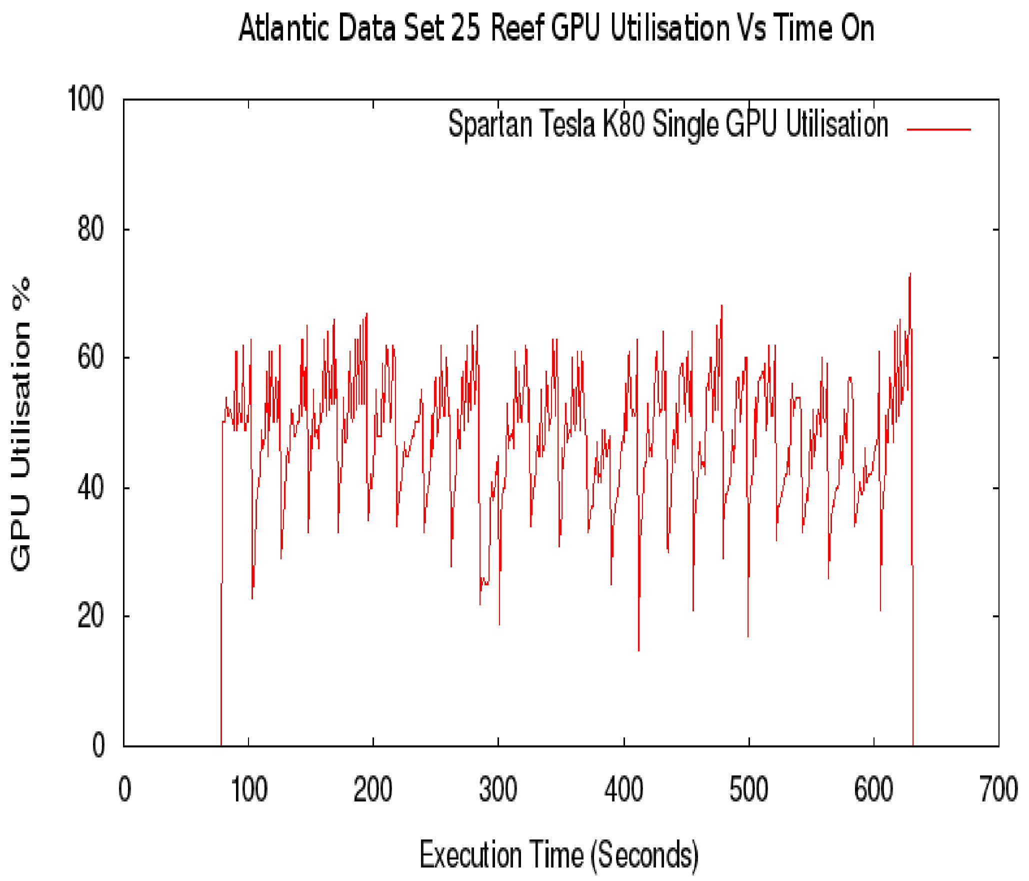
* Performance estimates running simulation for 2 hours

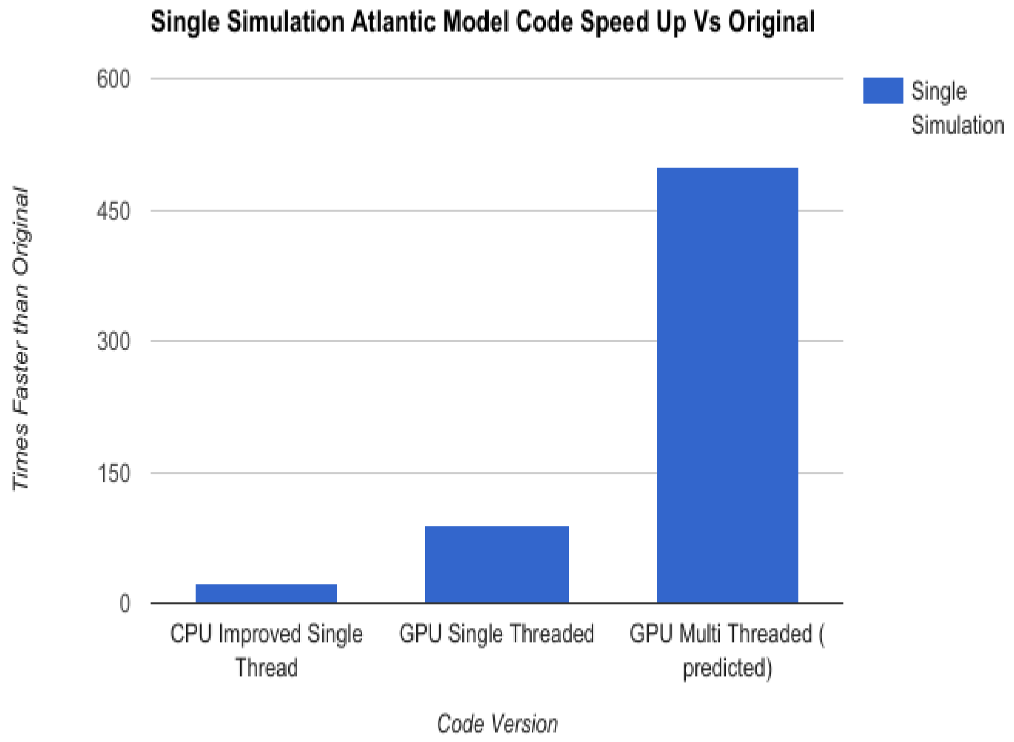
Additional Refractoring

Although a significant performance improvement has been achieved, significant further improvements are possible. A larger refactor to the codebase that simulates many reefs in parallel that would enable greater utilisation of all the GPU compute capabilities available on each Spartan node. This would work by leveraging multiple GPUs and multiple reef simulations into a single simulation across all GPUs in parallel. This would reduce the GPU idle time from waiting for the CPU process to provide it with more processing work. Once again, the most significant bottle-neck from the GPUs perspective is CPU wait-time.

Figure 1. shows the single GPU utilisation of the large Atlantic data set simulating 25 reefs for 100 days. The average utilisation is 48% when the GPU is used. As a Spartan GPU node has 2 K80 GPUs, and they each have two GPU core with a total of 4 GPUs, this shows that the node is not being fully utilised. Processing capability with a faster GPU (such as Tesla V100s) would obviously witness an ever greater performance improvement.

Additional Refractoring cont…





Additional Refractoring cont…

If the code is refactored to process reefs in parallel we anticipate that utilisation of the node would improve on a per-GPU and multi-GPU level, significantly reducing the single simulation time by fully utilising the Spartan GPU node on which it is run. With this change we predict a performance improvement of over 5x compared to the existing GPU code on meaning while using more resources on a node the execution time of a single simulation would greatly reduce. Smaller datasets would also likely achieve some improvement as per-GPU utilisation would increase. Demonstrated in Figure 2. is the performance increase of the current two versions, and the predicted performance of the multithreaded GPU version, when running a single simulation on the Atlantic data set of 442 reefs over 100 days.

Spartan's GPU Expansion

With notable performance improvements to a range of job profiles, a significant expansion of Spartan's GPGPU capacity has just been implemented. The partition, funded by Linkage Infrastructure, Equipment and Facilities (LIEF) grants from the Australian Research Council, has come together as a partnership between the University of Melbourne, La Trobe University, Deakin University, and the Royal Melbourne Institute of Technology (RMIT). Within the University of Melbourne, which makes up the bulk of the share (c80%), the research bodies include Melbourne Bionformatics (the successor to the Victorian Life Science Compute Initiative), the Melbourne School of Engineering, the Medical and Dental Health School, St Vincent's Hospital, and Research Platforms. The partition is composed of 75 nodes and 300 nVidia P100 GPGPU cards and a peak theoretical performance of 900TF.

The major usage of the new system will be for turbulent flows, theoretical and computational chemistry, and genomics, representative of the needs of major participants. Specific software applications have already been optimised for GPGPU (e.g., through library extensions or compilation versions) for these research projects, including NAMD, GROMACS (both molecular dynamics simulation applications), AMBER (biomolecule force field simulations), MATLAB (numerical computing environment), and a in-house developed application, HiPSTAR (fluid dynamics). After being made generally available in at the end of June, 2018 the system typically runs at 100% node allocation.

Future Collaborations

Nyriad's review found that there is significant opportunity in the use of data integrity and mathematical equivalence algorithmic techniques for enabling porting of code to GPUs with minimal impact to the research workflow. Firstly the use of the original language to take advantage of SIMD parallelism on the CPU, enabled the researcher to understand how the problem mapped to a GPU or the HPC cluster environment. When changes were needed to be made, the researcher was more confident to update the parallel version in their chosen language, which made updating native C++ CUDA implementations much easier for the GPU developer.

Nyriad is planning further work and collaboration with the project and the university of Melbourne. Planned activities include configuring one or more GPU nodes with large volumes of local storage, packaging the code into a containerised form, expanding the size, complexity and variety of the simulations, and training machine learning models to predict larvae dispersal based on simulated data Ability to scale out the simulation and processing across a cluster in a reliable fashion. Nyriad is also in the process of introducing NSULATE(TM) nodes on Spartan similar to what they have done with ICRAR in Western Australia.

Acknowledgements

Contributions to this paper have come from Mark Wilcox (Nyriad), Mitch Turnbull (Nyriad), and Eric A. Treml (University of Melbourne and Deakin University).

Slide 5 image from the the Marine Spatial Ecology and Conservation (MSEC) laboratory at the University of Melbourne, now Deakin University.

The authors would like to thank the University of Melbourne for their support in developing this work.

THANKS FOR WATCHING
& LISTENING PATIENTLY