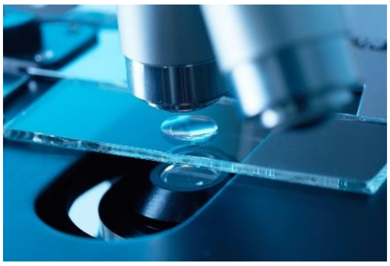
Accelerate your code
Discover your system's full potential

Software in scientific computing and instrumentation



How is software used in scientific instrumentation?

- Controlling workflows
- Integrating components
- Acquiring and storing data
- Implementing algorithms
- Visualising results

Speed matters

Accelerating the operation of the acquisition & computation speed of a scientific instrument does not only mean reducing waiting times. In many cases acceleration can become the enabling force to simplify workflows and provide real-time presentation of results that completely transform the device capabilities. As an example, a processing step that is accelerated by 2 orders of magnitude can turn an offline process to a real-time experience that immediately provides feedback to end users for the quality of results and save the hassle and cost of repeating the scanning of samples. It can also open the capability to simplify offline processing by converting offline processing pipelines to a single click experience.

Identifying Bottlenecks and choosing the right solution

Acceleration can only start when you have identified the bottleneck delaying your process. Analysing the nature of the problem and locating the hot-spots is the starting point of any serious attempt to accelerate an application.

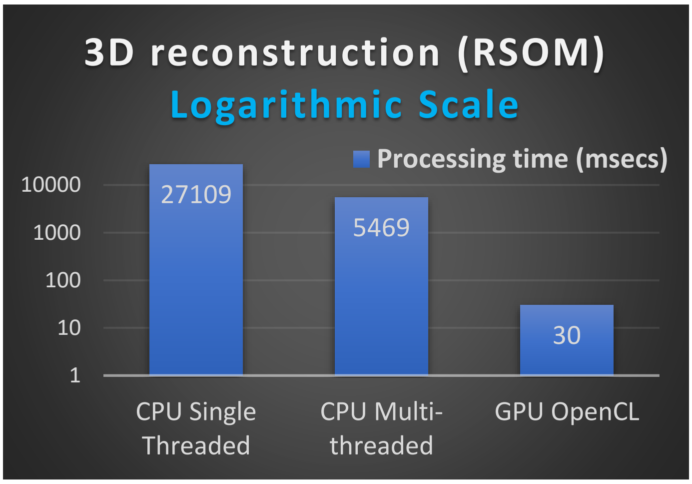
In almost all cases acceleration requires parallelization of processes/algorithms. It is not possible to parallelize all processes or algorithms, but there is almost always a way to identify parts of the process that are parallelizable. Once this is done, the next step is to decide what is the best technology to use. Possible solutions are:

a) Multi-core/multi-processor CPU systems.
b) General Purpose GPU processing solutions like OpenCL or CUDA.
c) FPGA based solutions.
d) Hybrid solutions using a mix of the above.

Real world examples

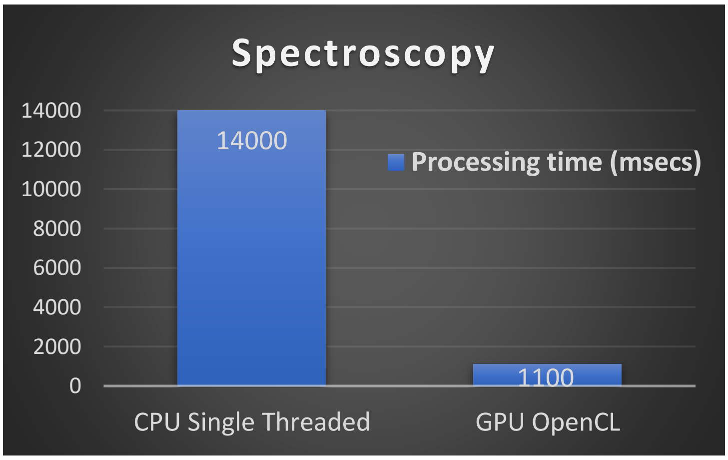
In the last 10 years we have worked closely with both academic researchers and companies ranging from innovative start-ups to NASDAQ listed multi-nationals. Each project has its own constrains and challenges and the solutions may differ significantly per case, but there is a common denominator: we managed to achieve typical speed ups in the range of 2 orders of magnitude in comparison to the initial code.

As an example, in Optoacoustics – a new imaging modality introduced during the last 15 years – we have worked on 2D and 3D reconstruction algorithms used in research and commercial devices. The following chart shows the performance improvement gained through the application of multi-threading, particularly when using GPU accelerators.



In RSOM (Raster Scan Optoacoustic Mesoscopy), the most demanding task is the 3D reconstruction process which can take minutes to complete. This is a big impediment for the usage of the device in medical applications. Optimization managed to achieve performance improvements in the range of **200 times faster t**han the multi-threaded CPU code and almost **1000 times faster** than the single threaded CPU case. This completely transformed the capabilities of the system and radically changed the way the technology was applied.

In another case, a multi-national leader in scientific instrumentation, was facing a challenging problem in advancing its spectroscopy solutions using multi-channel Focal Plane Arrays. The technology was exciting, but migration from single-channel to multi-channel operation presented challenges that could not be addressed without parallelizing the problem. The basic processing pipeline was



built around the FFT algorithm and the acceleration started by porting the FFT to the GPU. The chart shows the speed up achieved by implementing a GPU accelerated FFT in comparison to the original single threaded CPU FFT implementation. This “moderate” acceleration by *a* factor of 12 was only the beginning of the process. When the pre and post processing steps needed around the core FFT function were also ported to the GPU, the end-to-end acceleration reached was a factor of **20 times** faster than the original implementation.

How Rayfos can help you accelerate your code

In both cases discussed above, the achieved improvement not only accelerated the code and improved the user experience, but also became the enabling technology that transformed the usage of the products and made new applications feasible.

We have a long experience in creating and optimizing data acquisition systems that offer real-time processing for high bandwidth data streams at sustained rates in excess of several Gbps.

OpenCL

We have developed a stack of technologies that can efficiently acquire, store, process and present data/images at video rate speed. We are using solutions like OpenCL and Cuda to accelerate and parallelize code in GPUs or OpemMP and similar concurrency frameworks to parallelize code that is more suitable for multi-core CPU architectures.

We even go beyond software-only acceleration solutions, when the above tools cannot meet the performance requirements. We port part of the processing to FPGAs and try to find the right balance between FPGA logic and software. All the above solutions are useful not only in real-time driven applications, but also in cases where speed-up is essential in an offline setting.

If you think that your system is not running at its full potential, you are probably right. Feel free to contact us for a preliminary discussion of your system and the things you would like to accomplish, that today may seem out of reach but may be more feasible than you think.

Copyright