Implementing an interactive Mandelbrot Visualization on a GPGPU cluster using HPXCL

Martin Stumpf[1,2], Hartmut Kaiser[1], Thomas Heller[2]

[1] Louisiana State University, Center for Computation and Technology
[2] Friedrich-Alexander-University Erlangen-Nuremberg

Abstract

Recently built supercomputers demonstrate that the number of GPUs and OpenCL devices in clusters increases rapidly. While these devices offer a whole new level of computing power, GPUs are still rather unpopular. Writing scalable OpenCL applications takes more effort than the average user is willing to spend. We tried to overcome this obstacle by implementing HPXCL, a scalable OpenCL API for distributed systems, based on HPX. To demonstrate the scalability, we implemented a distributed Mandelbrot visualization using HPXCL and evaluated its performance in a cluster environment. Using this visualization, we created an interactive demo utilizing the Google Maps API.

HPXCL and HPX

We wrote HPXCL (HPX Compute Language) to create an open source API that is able to control OpenCL devices in a cluster environment. Our main focus was to keep the interface simple and easy to use, while still providing scalability and performance.
HPXCL uses HPX (High Performance ParalleX) to utilize the parallel resources as efficient and asynchronous as possible. HPX is an open source and freely available C++ runtime system, providing remarkable performance and scaling on many distributed and parallel systems. It is available for Windows, Linux and OSX, and can be used on a variety of cluster architectures.

The Mandelbrot Set

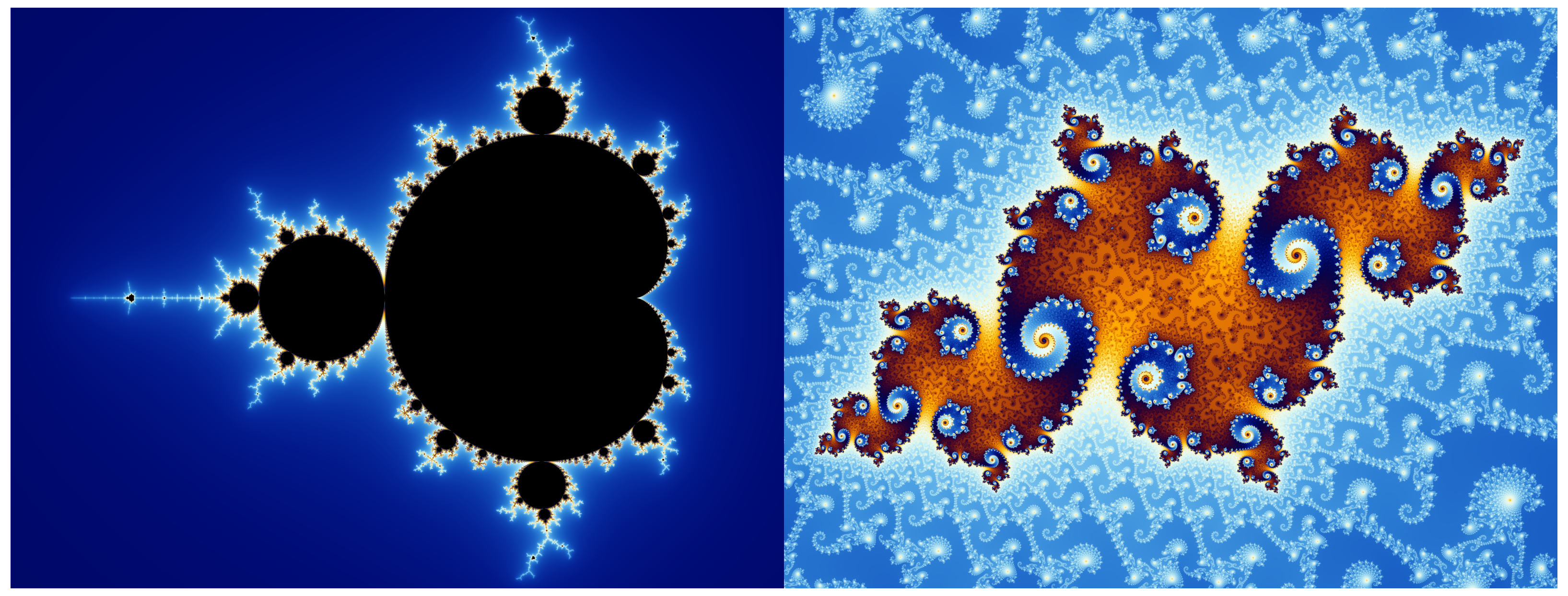
Fig. 1: Left: The Mandelbrot Set; Right: Detail of the Mandelbrot Set

We chose to use a visualization of the Mandelbrot set to test our HPXCL implementation. It is a well known example of an embarrassingly parallel problem, which makes it an ideal testcase to demonstrate the efficiency of HPXCL. The set is based on the divergence behavior of the complex series $z_{i+1} = z_i^2 + c$ with $z_0 = 0$ for different c values. Using the real and imaginary part of c as coordinates, and applying color based on how rapidly z diverges, one can create impressive looking pictures. Furthermore, its computational characteristics make it a perfect fit for OpenCL devices.

Implementation

In order to solve a problem in a distributed fashion, the problem needs to be split into several parts. Splitting the computation of a mandelbrot image is rather easy, as every pixel can be calculated independently from its surrounding pixels. Hence, a number of sub-images can be created, using a grain size modifier to control the split size. These sub-images can then be computed by several independent distributed workers. This approach scales even with heterogeneous workers, enabling the simultaneous use of GPU, accelerator and CPU based OpenCL devices.
We wrapped the resulting Image Generator in a webserver and used the Google Maps API to create an interactive Mandelbrot renderer. (Fig. 2)

Results

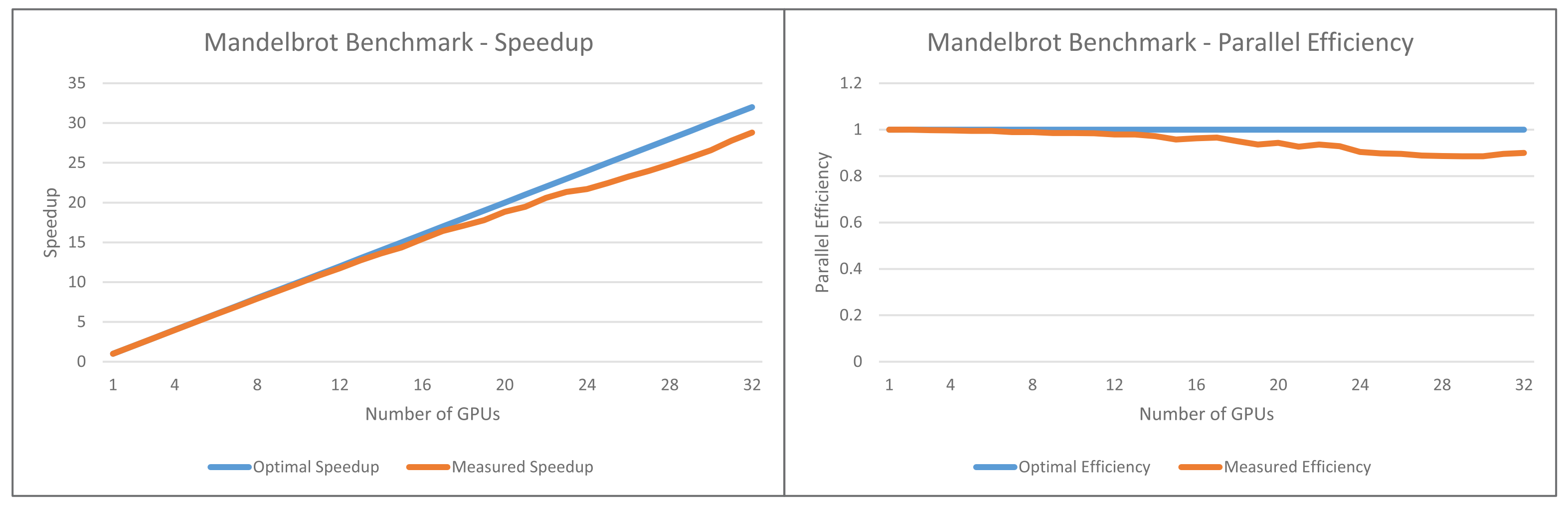

Fig. 3: Left: Speedup vs Number of Devices; Right: Parallel Efficiency vs Number of Devices

One of the most important factors on distributed programs is scalability, measured in parallel efficiency. The perfect value would be a constant parallel efficiency of 100%, which, however, is impossible to achieve in a real life application. Therefore, having an efficiency of 90% on 32 devices is a notable result.

Conclusion

Our Mandelbrot renderer was able to demonstrate the performance and scalability of HPXCL. Still, it is a rather simple use-case and only reflects a certain class of problems. Therefore, the next step would be to try HPXCL on a more complex task, like a multi-dimensional stencil code, whilst improving and extending it along the way.

Acknowledgements

- The Ste||ar Group
- Center for Computation and Technology, Louisiana State University
- Stampede (XSEDE)
- Friedrich-Alexander-University Erlangen-Nuremberg
- National Science Foundation

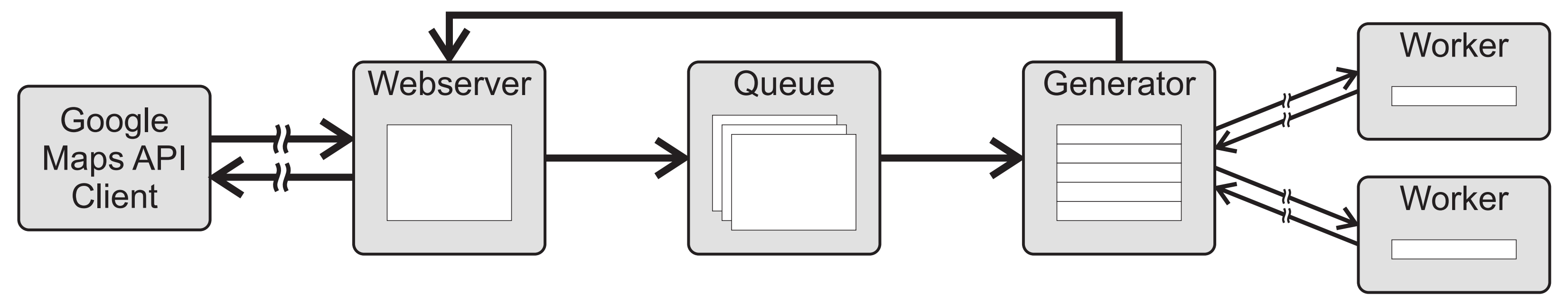

Fig. 2: Structure of the interactive Mandelbrot renderer