PC2 Hackathon Reports 2024

Contents

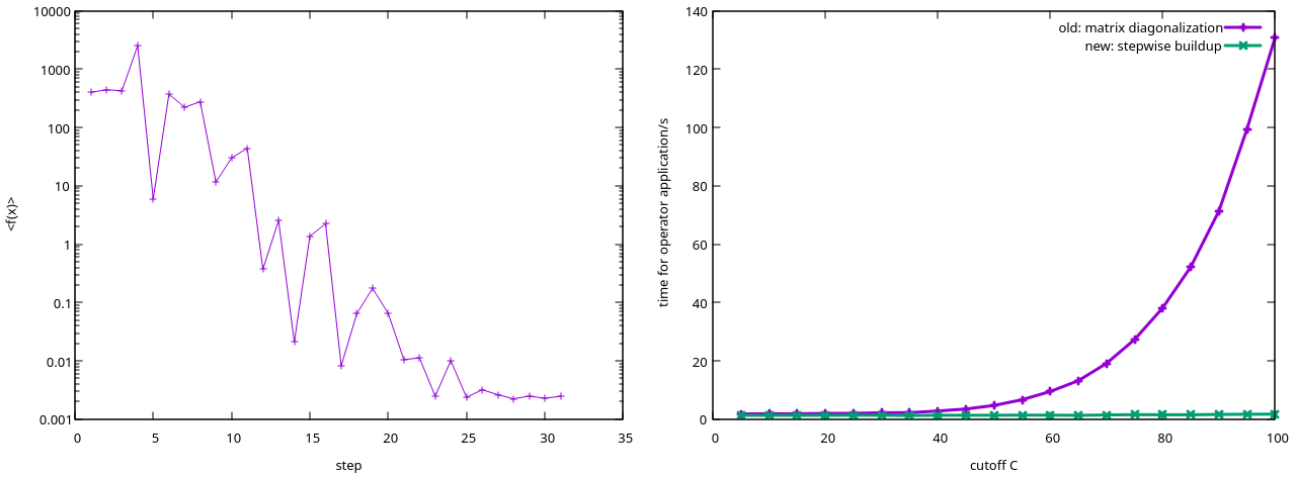
HPC Simulations for the Continuous-Variable Quantum Approximate Optimization Algorithm

Participants

- Zarin Shakibaei, Zuse Institute Berlin

Report

1. A Fock-space-based simulation program based on QuantumOptics.jl: **FockPhotonics.jl**
    - Supports state-sparsity, arbitrary operators, homodyne-measurements based on a monte-carlo-procedure
    - Work program for Hackathon:
        - Optimize sparse representation and threading support ✓
        - (optional) rework data structures (in progress)
        - Investigate numerical issues for very large cutoffs ✓
2. A decomposition program for the operators in Python (postponed)
    - Basic decomposition are already implemented
    - Work program for Hackathon:
        - Verify and optimize decompositions
        - (optional) optimized tree search
3. CV-QAOA program with Bayesian minimization
    - Basic minimization is implemented but noisy minimization
    - Work program for Hackathon:
        - Investigate convergence of minimization (✓)
        - Extend to non-quadratic minimization problems (in progress)



Left: Convergence of the sampled CV-QAOA procedure, Right: Performance optimization of the application of operators by new step-wise schema

Computational Fluid Dynamics in Modern Fortran

Participants

- Andris Rambaks, RWTH Aachen

Report

Goals for this Week

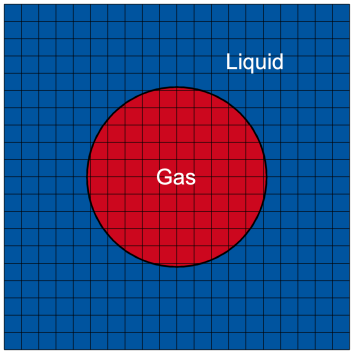
- Acquire in-depth Knowledge about OOP Software Design for HPC Applications
- Leverage CMake and Preprocessor Directives to improve Modularity
- Design OOP Façade for PARAMESH

Acquire in-depth Knowledge about OOP Software Design for HPC Applications

- What does it mean to write „good“ code for HPC?
  - Rule-of-Thumb for Polymorphism
    * Compile-time polymorphism for performance-critical sections
    * Runtime polymorphism in other section for flexibility
  - Rule-of-Thumb for performance-critical sections
    * Arrange operations for high data locality
    * Utilize OPENMP for multithreading
    * Use mem. access patterns which favor vectorization
    * Minimize branching

Leverage CMake and Preprocessor Directives to improve Modularity

- Compile and link all sensible combinations of code into one large library
  - e.g. 3 Solvers * 4 Integrators = 12 Combinations
- Leverage preprocessor directives in higher levels of abstraction to select correct precompiled implementation and to generate final executable



Computational Fluid Dynamics Simulations calculate the interactions of Momentum, Heat and Mass between Liquids and Gases.

Enhancing PLSSVM with FPGA support

Participants

- Alexander Van Craen, University of Stuttgart

Report

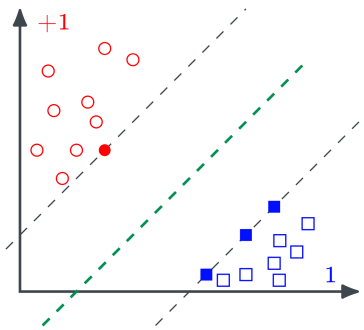
Reached Goals

- Adopted PLSSVM CMake to handle new FPGA backend (somehow hacky)
- hopefully first synthesizing kernels
- implementing first optimizations (like shift registers, unrolling, loop_coalesce, reuse blocks, enable parallelism) + learning why and what similarities / differences there are to GPUs

Lot of lessons learned

- How To procedure when developing for FPGAs
- How to give kernels readable names in the report (extra clases as template parameter)
- place of *.ptj/reports + how to read them
- export LC_NUMERIC="en_US.UTF-8" -> language settings problem: https://upb-pc2.atlassian.net/wiki/spaces/PC2DOK/pag settings-forwarded-from-your-computer
- How to and why implement register shifting, loop collapsing and loop coalescing
- USM + ND Kernel are hiding behind strange things like *memset* or *alloc 2d*
- Getting started literature: "FPGA Optimization Guide for Intel® oneAPI Toolkits"
- How to efficient rewrite ND-Range Kernels to single task + get rid of barriers
- SYCL buffers > malloc_device
- templates in kernel lead to multiple Kernels -> nead a lot of space -> space neaded for optimisatzion -> nead to rewrite cmake build scripts to generate multiple shared libraries to link specific kernel at runtime
- adopt cmake to have kernel as single library to reduce build time

I successfully overcame the first hurdle of FPGA programming with the great help of Heinrich, Tobias and Alex, thank you so much!!! Now I know where to look when I have problems and I have enough basic knowledge that I know what and how to ask to get ahead.



Support Vector Machines are used to classify data.

PC2 Hackaton - Optimizing Magneto-Hydrodynamics 3D design for FPGA

Participants

- François-Xavier Mordant, Maison de la Simulation

Report

In this hackaton report, we present the optimization process for a 3D Magnetohydrodynamics (MH3D) code, targeting FPGA architectures.

MHD modeling is a key tool for understanding the behavior of electrically conducting fluids, such as plasmas, in a variety of scientific and engineering applications (space physics, geophysics, astrophysics, and engineering …)

Given the computational intensity and high dimensionality of these simulations, leveraging the parallel processing capabilities of FPGAs offers significant potential for improving performance and efficiency.

The code is a first order template, i.e. 7 points. For each cell there are 8 variables: density, pressure, 3 velocities for each dimension x, y, z, as well as the 3 magnetic fields for each dimension. The code works in 3 steps: First it calculates the Dt variable (scalar, reduction on all cells), then it calculates the main MH3D (the flux contributions between all the stencil cells), and finally it updates the boundaries. The Dt calculation and the MH3D have already been merged in one kernel, leaving another kernel for the boundary updates.

For simplicity, the grid is regular and edge conditions are periodic.

In the long run, this will serve as a base application to perform adaptive mesh refinement optimization for FPGA, which is the subject of my thesis.

This report lists (and details a bit) the strategies employed to optimize the code for FPGA using intel OneAPI, focusing on maximizing computational throughput and minimizing resource utilization.

Hackaton objectives were :

1. Optimize stencil-like computations

- Reduce design size (logic, RAM) so it can be synthesized
- Fully pipeline the main computation loop (reduce II from 21 to 1)

2. Domain decomposition - multiple kernel execution

- Use multiple pipes per design where possible
- Strategies for running across multiple MPI nodes

What could be achieved

Most Optimizations were performed to reduce design space, as it was the main issue - Remove a whole kernel that was used for the first Dt computation, since other Dt computation are made within the MH3D kernel for the next iteration. Sadly, this doesn’t reduced by much … - Replace the array of 8 variables per cell by an array of struct that contains the 8

variables. This reduce resource usage a bit, but mostly improved by 3 the Initiation Interval - Function inlining. Some computations were performed multiple times, and inlining those computations into a function and reusing the result registers avoids performing the same computation multiple times. - Replace similar resource-intensive operations with a loop over the various values. Makes code a bit harder to read, but big reduction in the number of logical resources used, surprisingly. - Replace the size_t datatype to ac_int for index computations. Big improvement in logic reduction.

		max ALMs :	1825600			
DATATYPE	**VERSION**	**SYNTHETISE**	**ALMs**	**ALMs % USAGE**	**II on MH3D loop**	**EXPLANATION**
DOUBLE	MH3D		1790967	98.1	21	BASELINE
DOUBLE	2. MH3D_remove_dt		1719069	94.2	21	delete a whole kernel from design that was called once
DOUBLE	3. MH3D_rdt_struct		1709876	93.7	6	use **array of struct** instead of array so the compiler understands the pattern
DOUBLE	4. MH3D_rdt_struct_better		1702908	93.3	6	small improvements
DOUBLE	5. MH3D_rdt_sb_merge_compute		1673183	91.7	6	function inlining, reduce computation that was performed multiple times
DOUBLE	6. MH3D_rdt_sb_merge_compute_tests1		1654748	90.6	6	very impressive : remove few divisions and operations
DOUBLE	7. MH3D_rdt_struct_ac_int		1614109	88.4	6	use **ac_int** for indexing computations instead of size_t
DOUBLE	8. MH3D_rdt_struct_ac_int_better		1558701	85.4	6	merge the ac_int version with all small improvements performed
FLOAT	4. MH3D_rdt_struct_better_float	YES	566074	31	6	from the struct function, use float variables instead of double
FLOAT	5. MH3D_rdt_sb_merge_compute_float	YES	543110	29.7	6	apply some of the optimizations & use float
FLOAT	8. MH3D_rdt_struct_ac_int_better_float	YES	460772	25.2	6	use ac_int, all small optimizations & use float

Versions of the iteratively improved code.

> Max ALMs for our Agilex board : 1825600. The two middle columns ALMs % Usage and II on MH3D Loop are the indicators for logic resources reduction and performance improvement.

We manage to get the design running, and improve performance !

Going to float was the biggest improvement, although not quite what we want, there is a trade-off between synthesizable design and simulation accuracy.

Unfortunately, doing domain decomposition and multi-fpga/MPI was a bit optimistic in the given timeframe, and is postponed

```
DATATYPE DtDx = Dt / DX;
DATATYPE DtDy = Dt / DY;
DATATYPE DtDz = Dt / DZ;
```

```
DATATYPE dt_inv_a[3];
DATATYPE dt_a[3];
DATATYPE dxyz[3] = {DX, DY, DZ};

for(unsigned char a = 0; a < 3; ++a){
    dt_inv_a[a] = 1 / dxyz[a];
    dt_a[a] = dt_inv_a[a] * Dt;
}

DATATYPE DtDx = dt_a[0]; // Dt / DX;
DATATYPE DtDy = dt_a[1]; // Dt / DY;
DATATYPE DtDz = dt_a[2]; // Dt / DZ;
```

Replacing divisions with multiplication of the reciprocal leads to surprising effect on the performance.

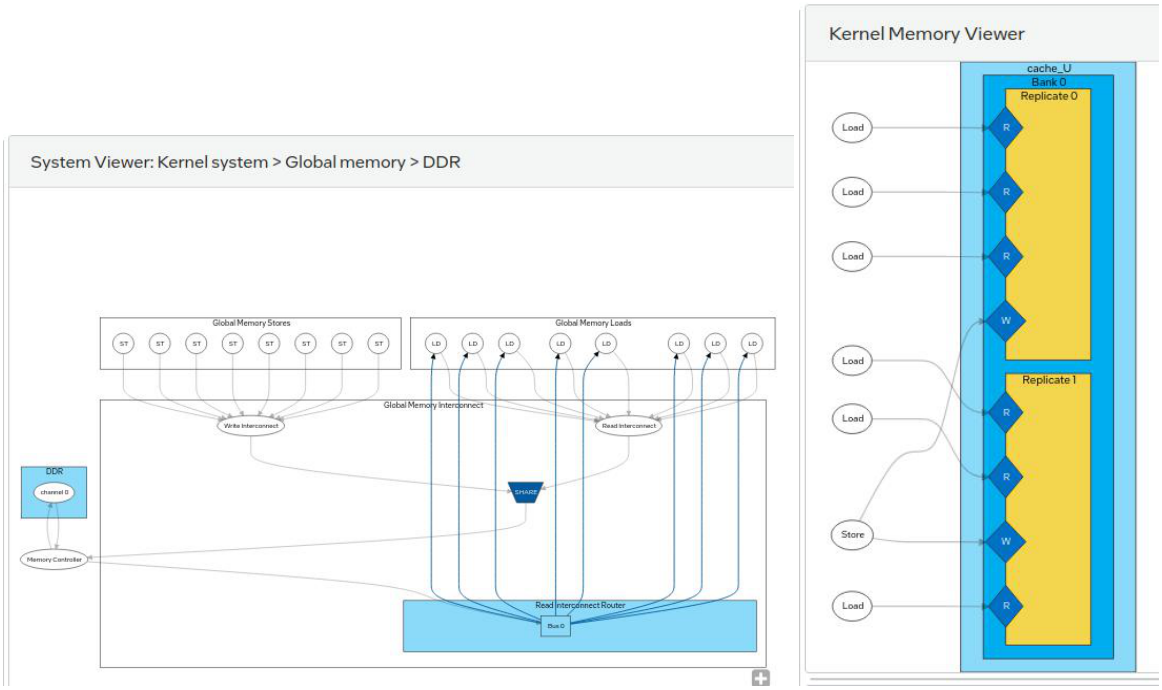
Future work directly related, ideas from the hackaton

- Use `ap_float` to find the tradeoff between datatype precision and resource usage.
- Some stencil optimization technique for simpler code in 2D, using dual cache and some shift registers
- Boundary update function optimization by linearizing loads & writes in burst
- Various ideas for area reduction in performance, e.g. compiling code with lower clock frequency can reduce logic usage, and more …

My take-aways!

A lot of optimization techniques to reduce design resources, improved report reading, especially on the global & kernel memory inner-working and how some code desicion change memory movements and influence performance. Also learn the existence of some tools for synthesized design, and a lot of very FPGA specific concepts that directly influence design conception & final precision.

And a lot of good time and discussions with talented engineers and with highly motivated students !



Screenshots of the Synthesis Reports.