Università della Svizzera italiana

Institute of Computational Science ICS

Università della Svizzera italiana

CARDIOCENTROTICINO

Center for Computational Medicine in Cardiology CCMC
Institute of Computational Science ICS
Faculty of Informatics

SCCER SoE

SWISS COMPETENCE CENTER for ENERGY RESEARCH
SUPPLY of ELECTRICITY

SCCER – FURIES
Shaping the FUtuRe SwIss Electrical InfraStructure

Eurohack 2019

Utopia: a C++ library for parallel non-linear constrained multilevel solution methods

Team: Patrick Zulian, Alena Kopanicakova, Maria Nestola, Daniel Ganellari
Mentors: Nur Fadel, Andreas Fink, Luc Berger-Vergiat (Sandia)

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

NVIDIA

Sandia National Laboratories

October 4th, 2019, Lugano

Introduction



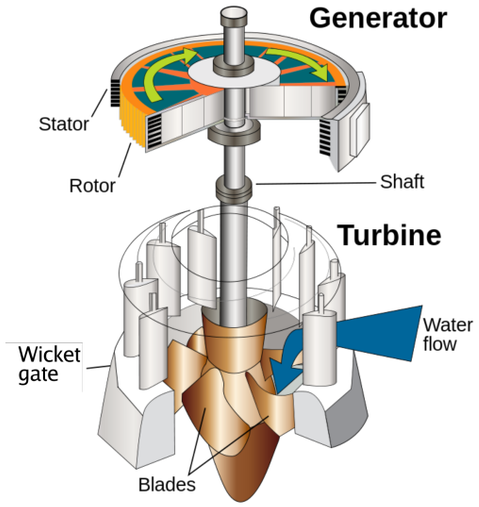
High-performance computing

Usability

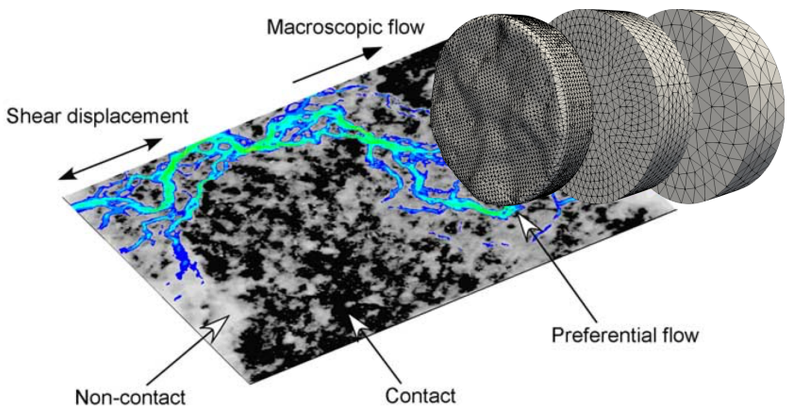
+

Flexibility!

Computational energy



Water turbines

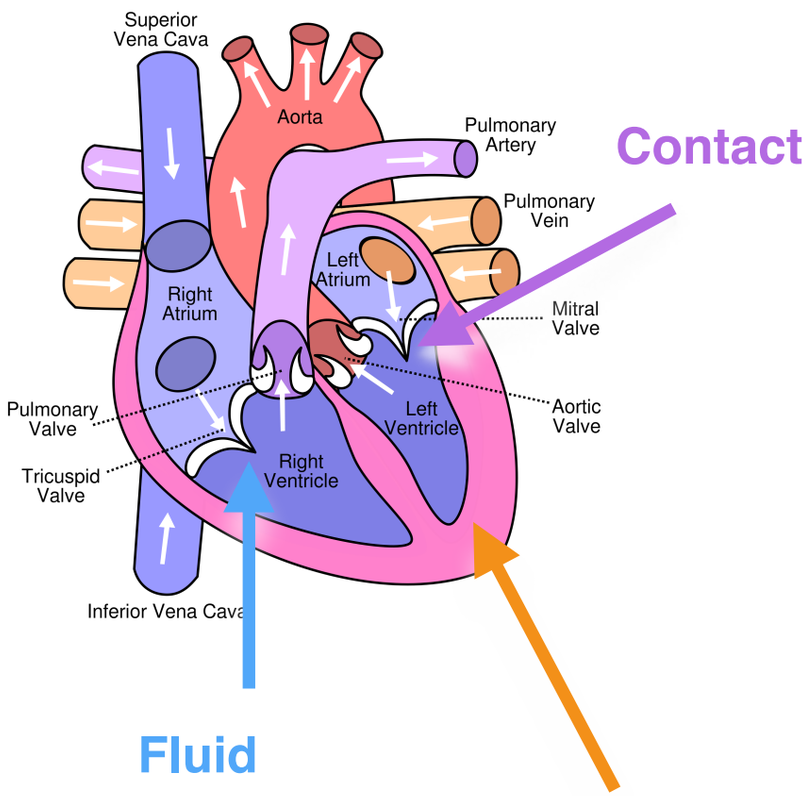


Computational Geophysics

SCCER SoE
SWISS COMPETENCE CENTER for ENERGY RESEARCH
SUPPLY of ELECTRICITY

Other applications

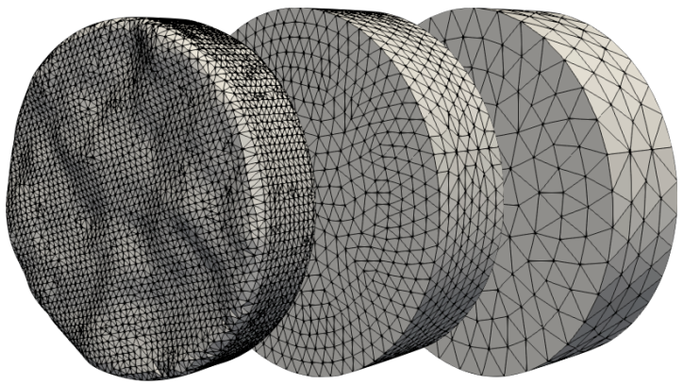
Simulation of the heart

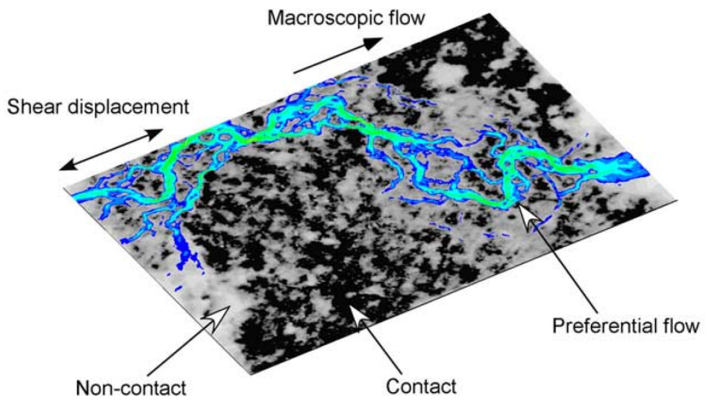


bitbucket.org/zulianp/utopia

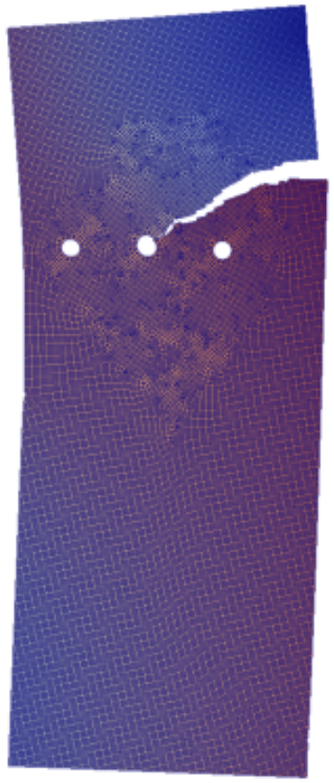
Università della Svizzera italiana
Institute of Computational Science ICS

Multilevel solution methods

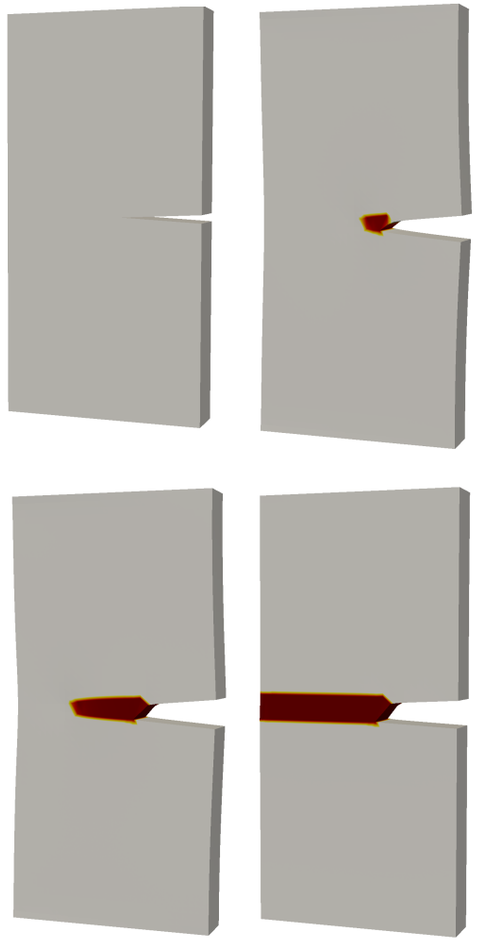




Multigrid solver for **computational geology** (C. Von Planta)



Multigrid solver for **crack simulations** with XFEM[1] (H. Kothari)



Multigrid solver for **Phase-field method**[2] (A. Kopaničáková)

Literature: **1** Z. Goangseup, and T. Belytschko, 2003.
2 C. Miehe, F. Welschinger, and M. Hofacker, 2010.

Architecture

Expression templates, multilevel non-linear solvers

C++ front-end

Petsc **or** **Tpetra/Kokkos**

Linear/Nonlinear algebra

Unified abstractions for communication, and memory management, reduced API complexity

Front-end

```
y = transpose(A) * x
```

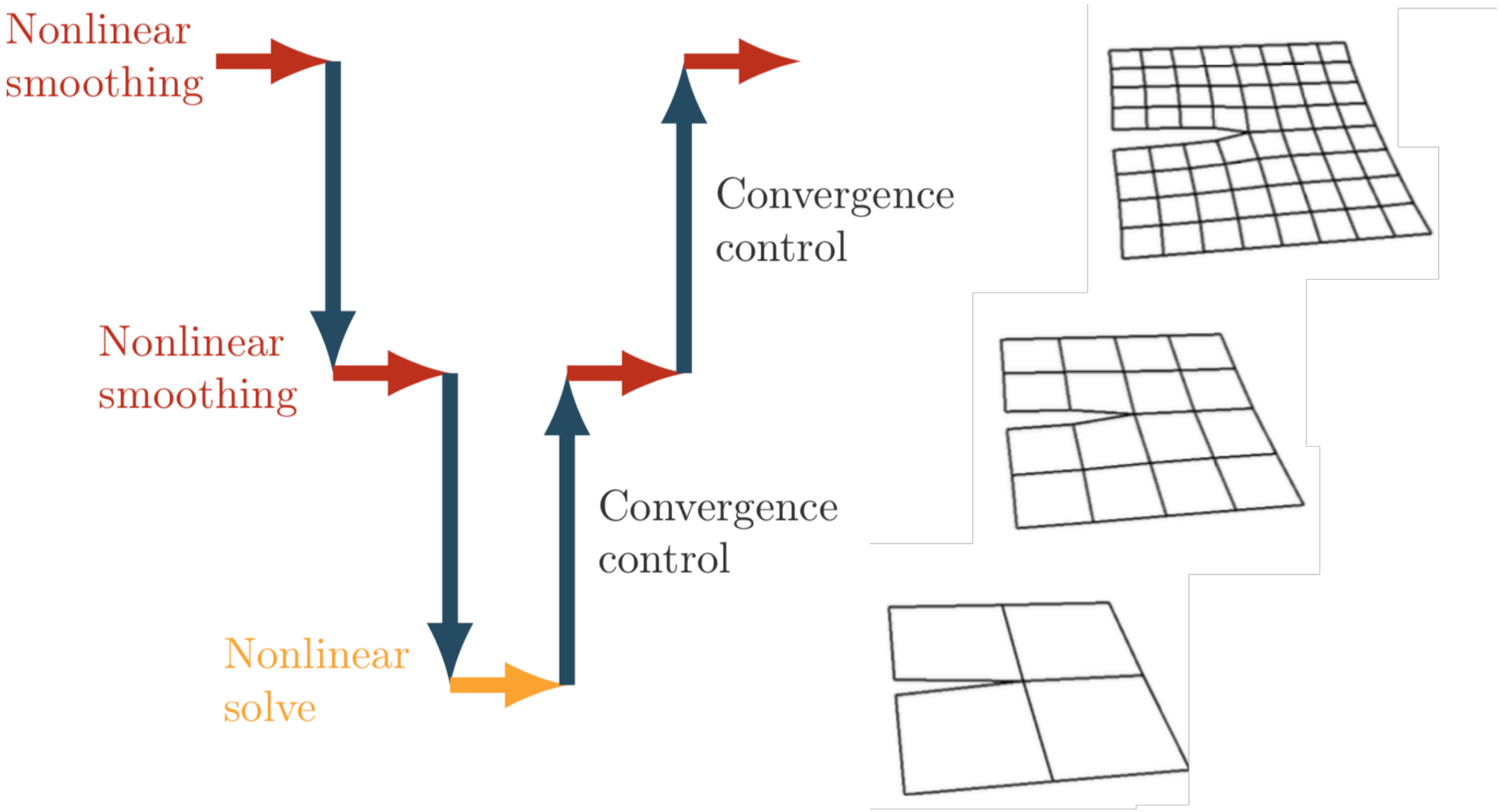
Backend

Petsc —> MatMultTranspose(A, x, y)

Tpetra —> A.apply(x, y, Teuchos::TRANS)

Goal

- **Recursive Multilevel Trust Region aka RMTR on GPU**
  - Globally convergent non-linear solution method
  - Non-convex problems



Steps made for porting the code to GPU

- Using **Kokkos/Tpetra**
- Extend *Utopia* **front end,** expose concept of **device**

```
Vector v1, v2, result; //init vectors to some values
auto d_v1  = const_device_view(v1);
auto d_v2  = const_device_view(v2);
utopia::parallel_each_write(
        result,
        UTOPIA_LAMBDA(const SizeType &i) -> Scalar
        {
                Scalar a = d_v1.get(i);
                Scalar b = d_v2.get(i);
                return some_function(a, b);
        }
);
```

Fundamental improvements (all backends)

- **Nonlinear smoother (MPRGP variant of CG with constraints)**
  - Identified spurious allocations
    - Removed temporary tensors
    - y = x - 0.5 * w

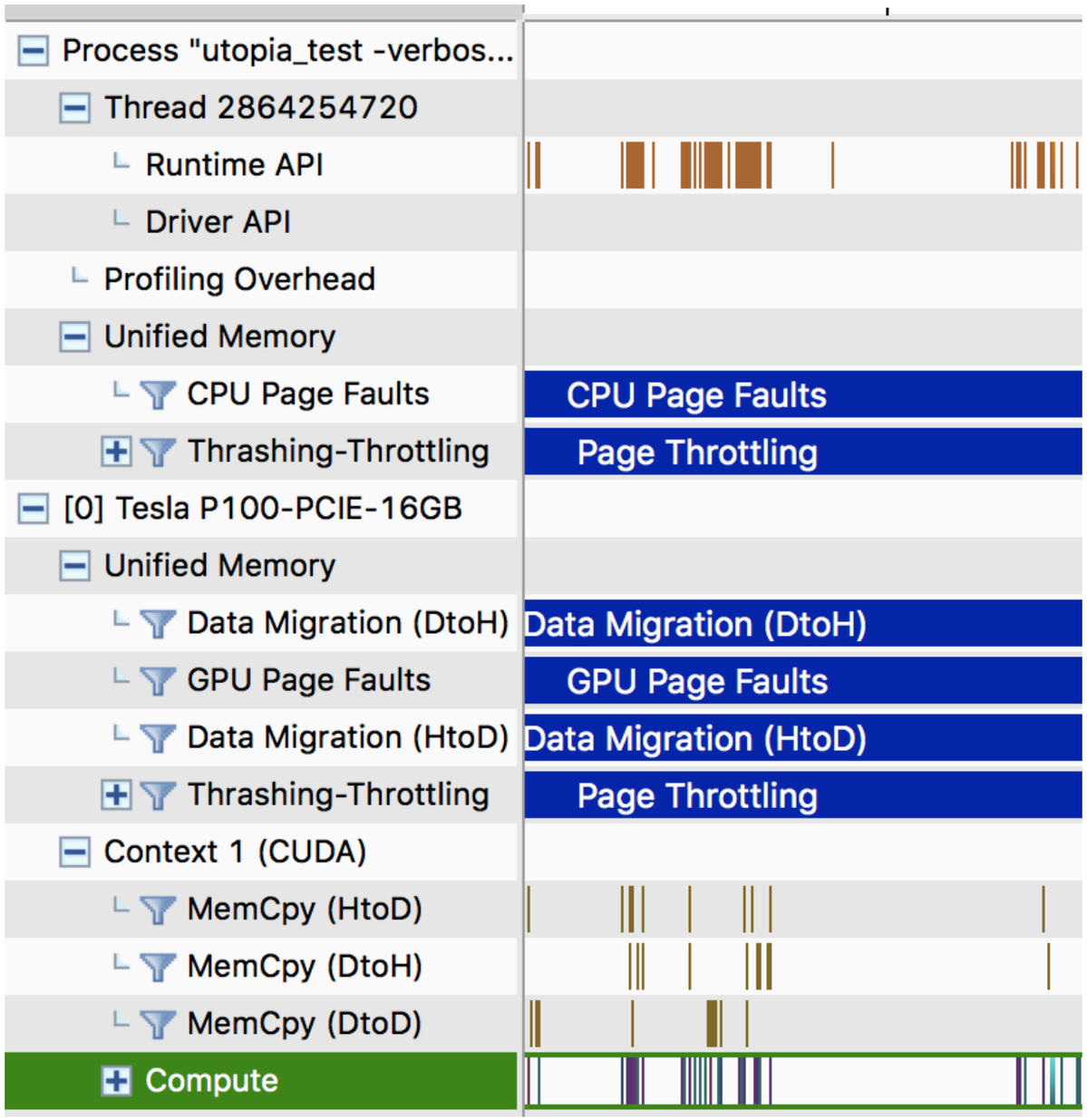
      —> We improved the evaluation for avoiding the extra allocation for y (all existing code improved)

- We implemented specific optimizations
  - **double** alpha = dot(u, v)/dot(w, v);
  - **double** m = multi_min(u, v)

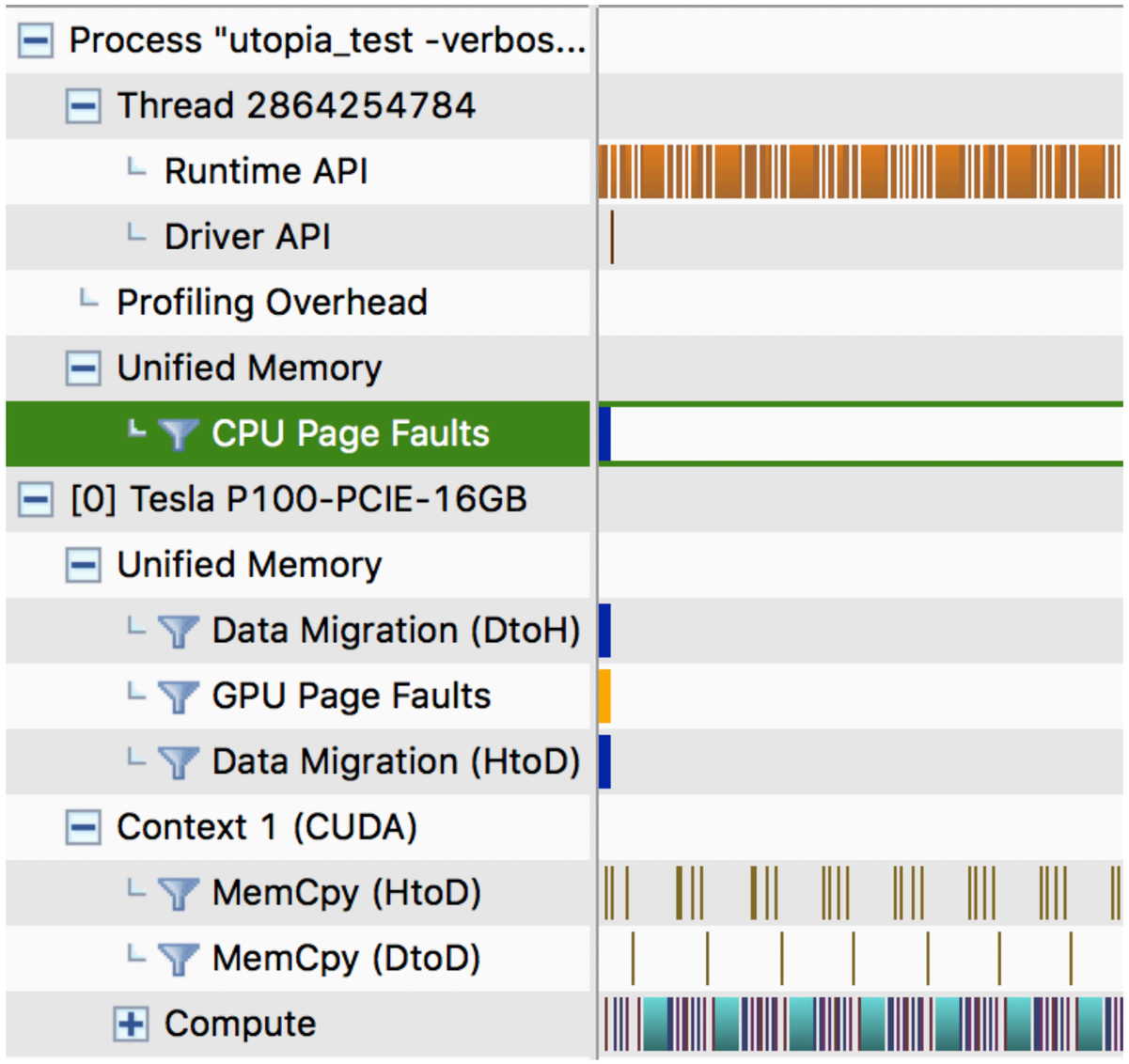
    —> Single GPU kernel (Kokkos only), one MPI_reduce (instead of 2)

Current state

Before



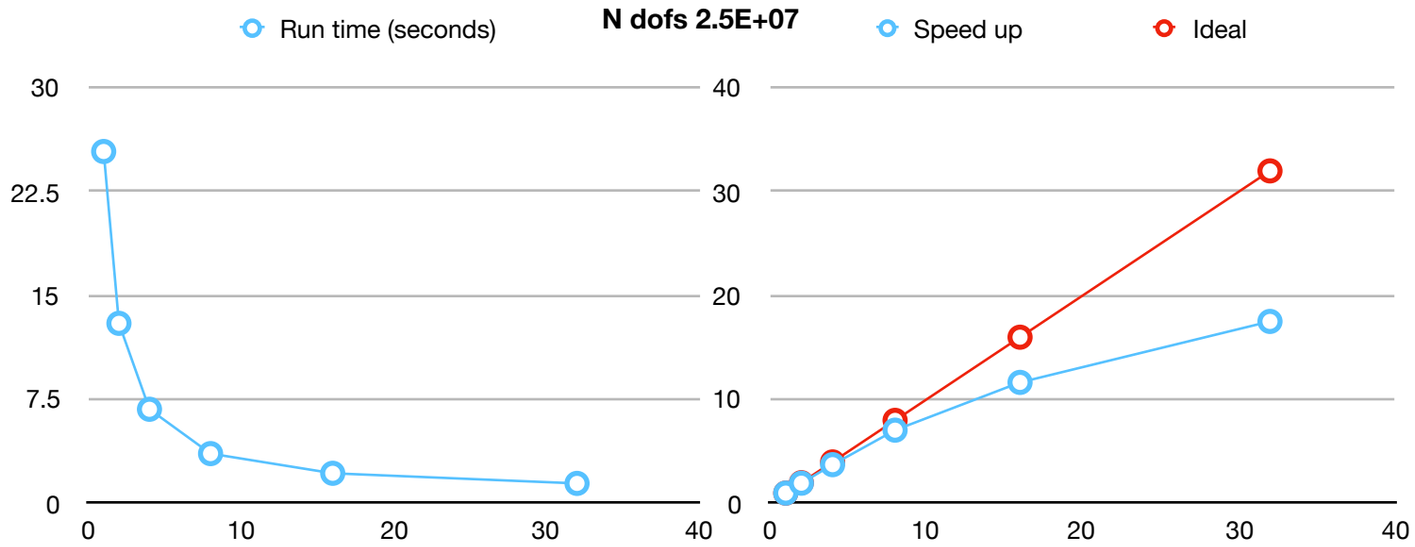
After

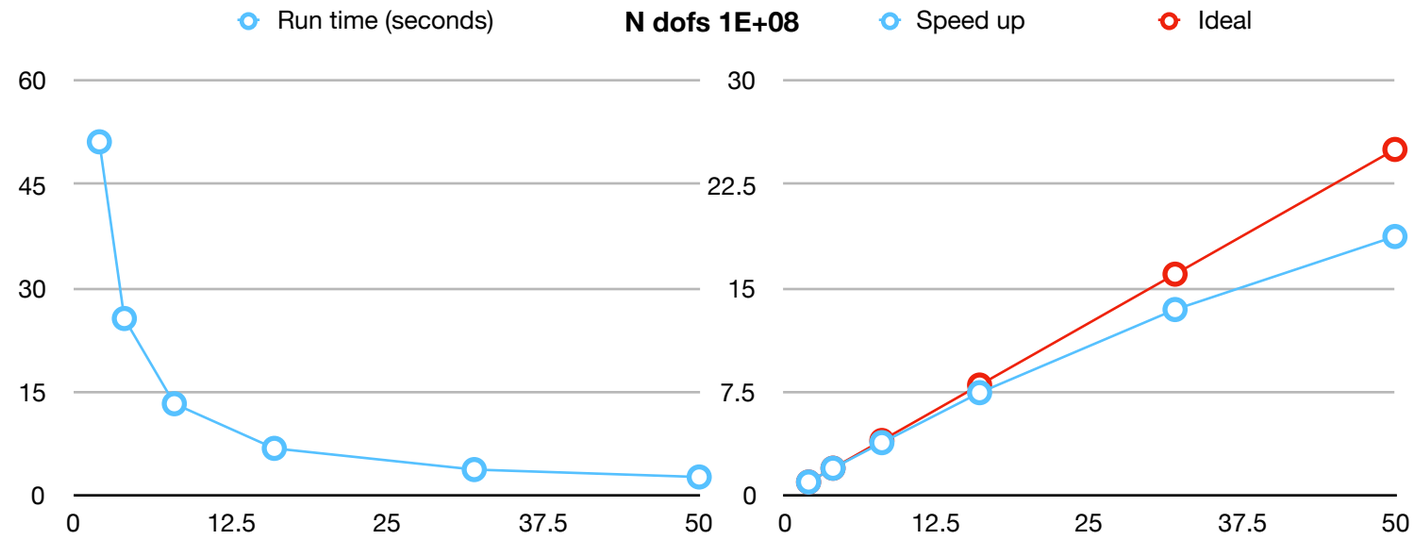


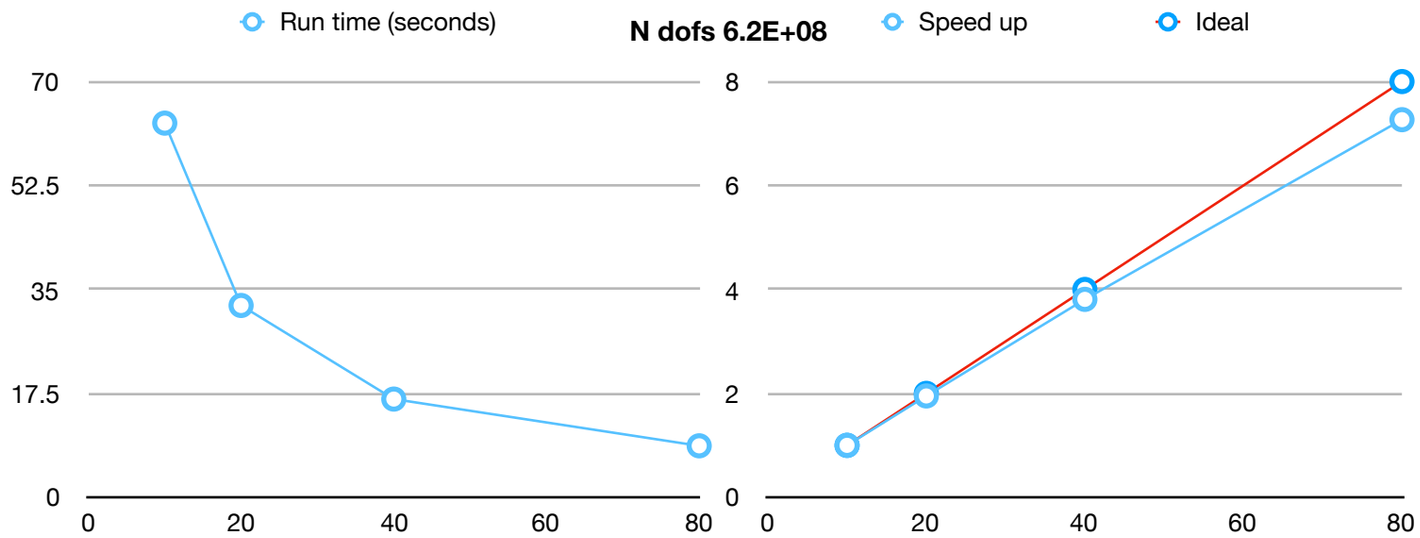
MPRGP performance (1)

Size	GPU-old (Cuda)	CPU (OpenMP)	GPU (Cuda)
1E+06	12.5	1.14	0.39
2.5E+07	272.1	33.79	4.94

MPRGP performance (2) - strong scaling

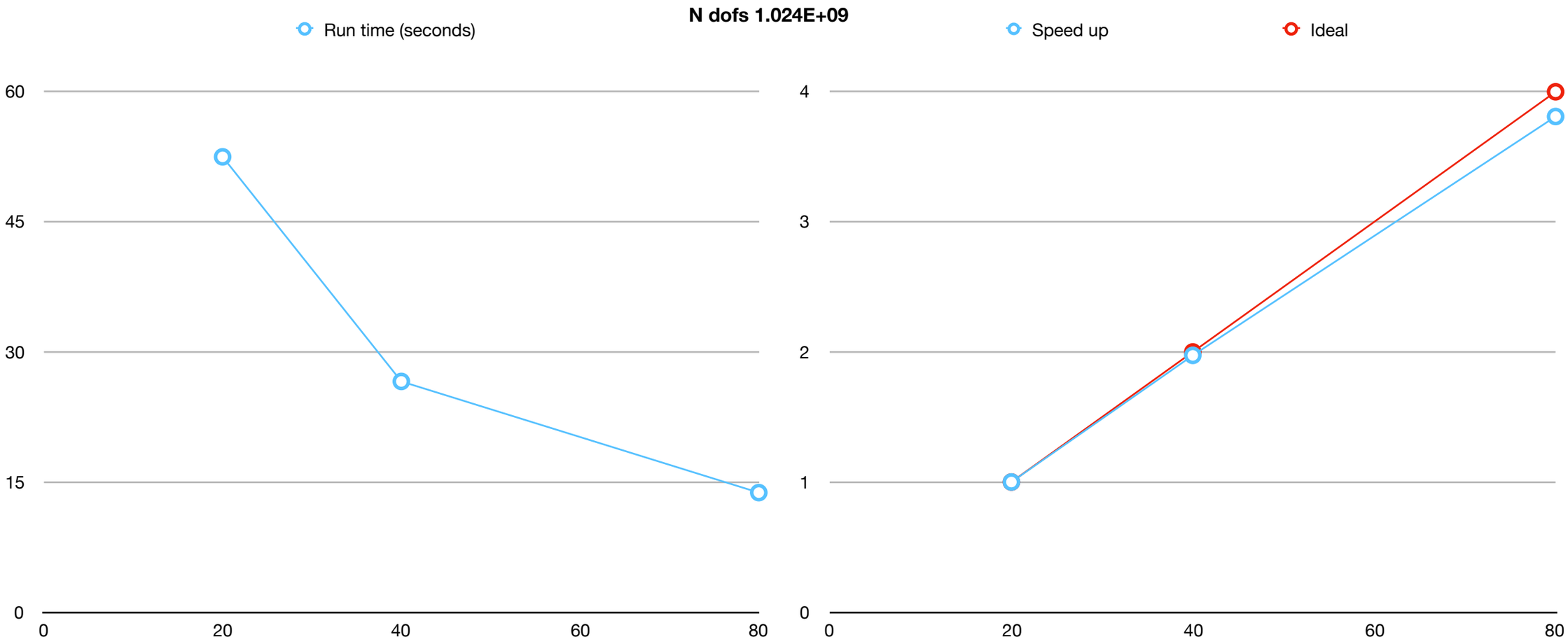






MPRGP performance (3) - strong scaling

Largest example



Size of examples is limited
by the Petsc DM library compiled
on Daint (integer representation)

Conclusion

Future work

- Complete RMTR GPU port
- Clean-up and improve hackathon output
- Nonlinear finite element assembler with Kokkos

Special thanks to



CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre



- Our mentors