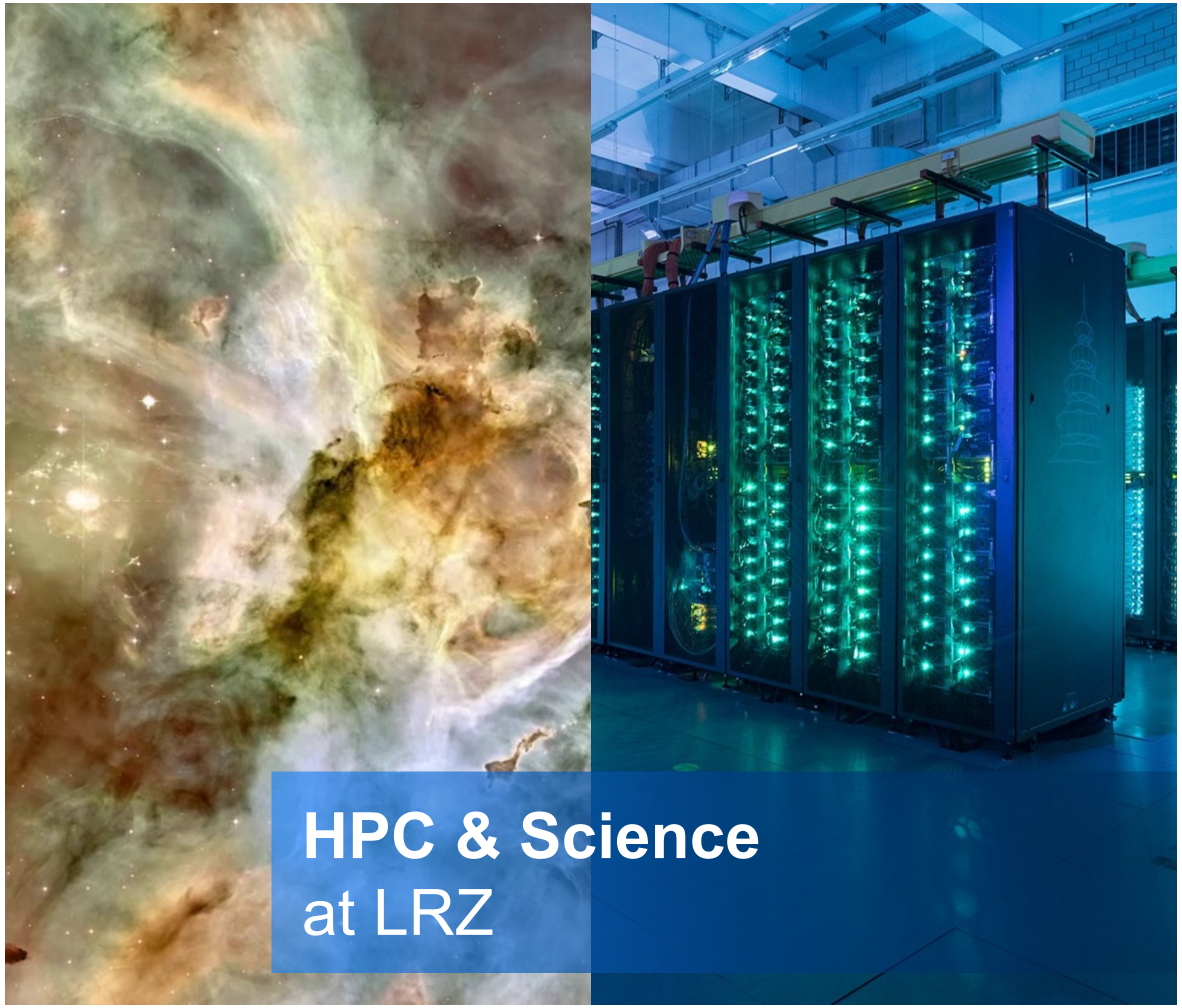
Leibniz Supercomputing Centre
of the Bavarian Academy of Sciences and Humanities

Leibniz Supercomputing Centre

oneAPI Case Study: DPEcho | Salvatore Cielo

The LRZ **CXS** group and **Application Labs**

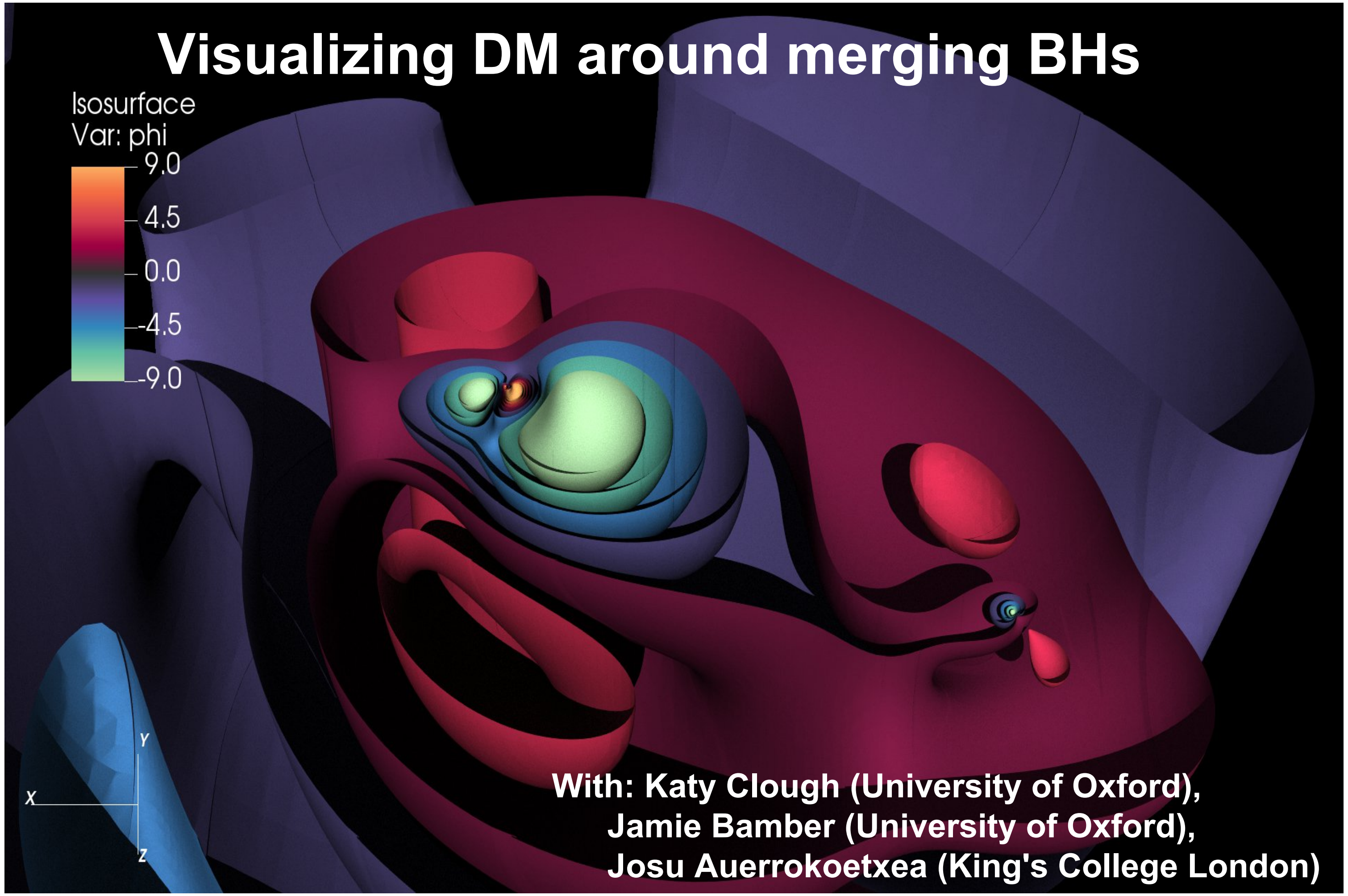


Supporting basic **research**:

- GCS/PRACE mentoring
- Maintaining **software stack** at LRZ
- HPC **courses** (scivis, parallel coding, GPUs, ...)
- Collaborations for **code modernization**

Application Support at LRZ

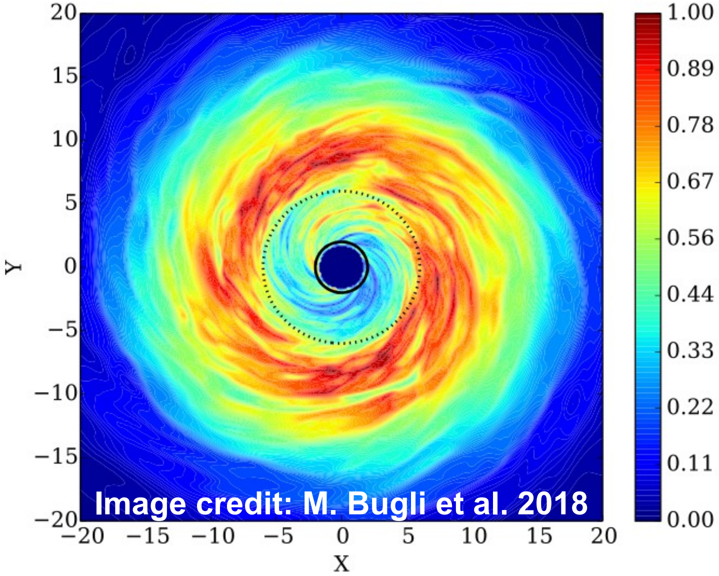
Scientific Visualization with oneAPI (OSPRay on SuperMUC-NG)

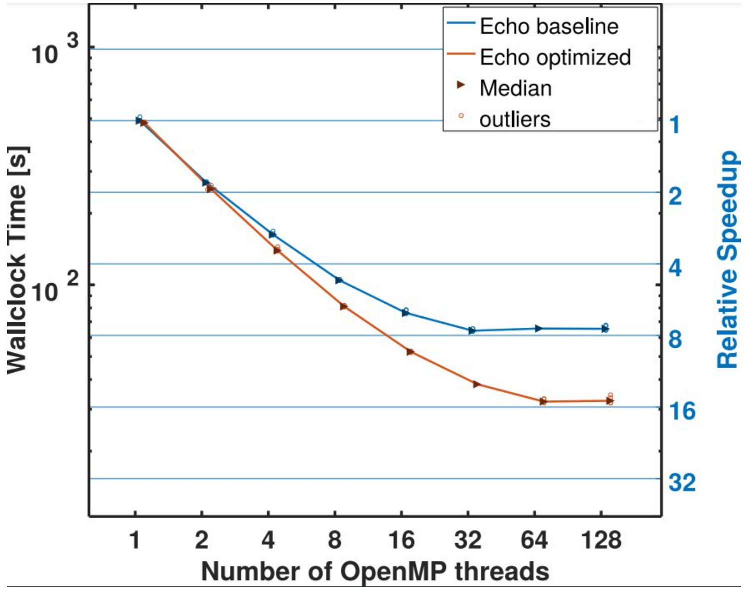


The ECHO GR-MHD code

GR-MHD for astrophysics

- finite differences, Godunov type, shock-capturing scheme
- Black Hole accretion disks ▶
- high-energy sources: pulsars, wind nebulae, neutron stars





3D-HPC version at LRZ - 2018

- Fortran90 code, pure CPU
- hybrid parallelization with MPI+ OpenMP
- Memory-bound (Analysis with Intel Advisor)
- Run on LRZ Cool-MUC3 (KNL)

ECHO: an Eulerian Conservative High Order scheme for genera relativistic magnetohydrodynamics and magnetodynamics

L. Del Zanna[1], O. Zanotti[1], N. Bucciantini[2], and P. Londrillo[3]

ABSTRACT

Aims. We present a new numerical code, ECHO, based on an *Eulerian Conservative High Order* scheme for time dependent three-dimensional general relativistic magnetohydrodynamics (GRMHD) and magnetodynamics (GRMD). ECHO is aimed at providing a shock-capturing conservative method able to work at an arbitrary level of formal accuracy (for smooth flows), where the other existing GRMHD and GRMD schemes yield an overall second order at most. Moreover, our goal is to present a general framework, based on the 3 + 1 Eulerian formalism, allowing for different sets of equations, different algorithms, and working in a generic space-time metric, so that ECHO may be easily coupled to any solver for Einstein's equations.

Performance and Portability

SYCL™ khronos.org/sycl

"SYCL (pronounced 'sickle') is a royalty-free, cross-platform abstraction layer that enables code for heterogeneous processors to be written using standard ISO C++ with the host and kernel code for an application contained in the same source file."

By Uni-Heidelberg:
HIP, OpenMP, CUDA

OpenSYCL

By Codeplay:
OpenCL + SPIR-V
Now: oneAPI on AMD and NVIDIA HW

ComputeCpp™

…

x86 CPUs, GPUs, FPGAs

oneAPI, TBB
DPC++ / LLVM

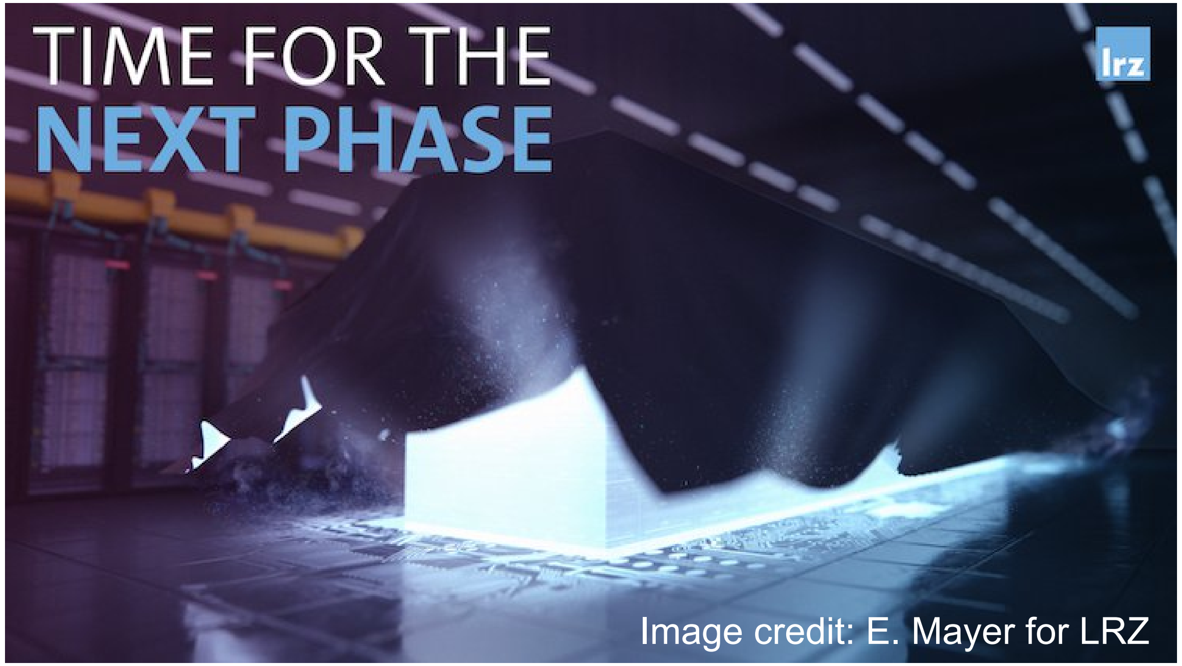
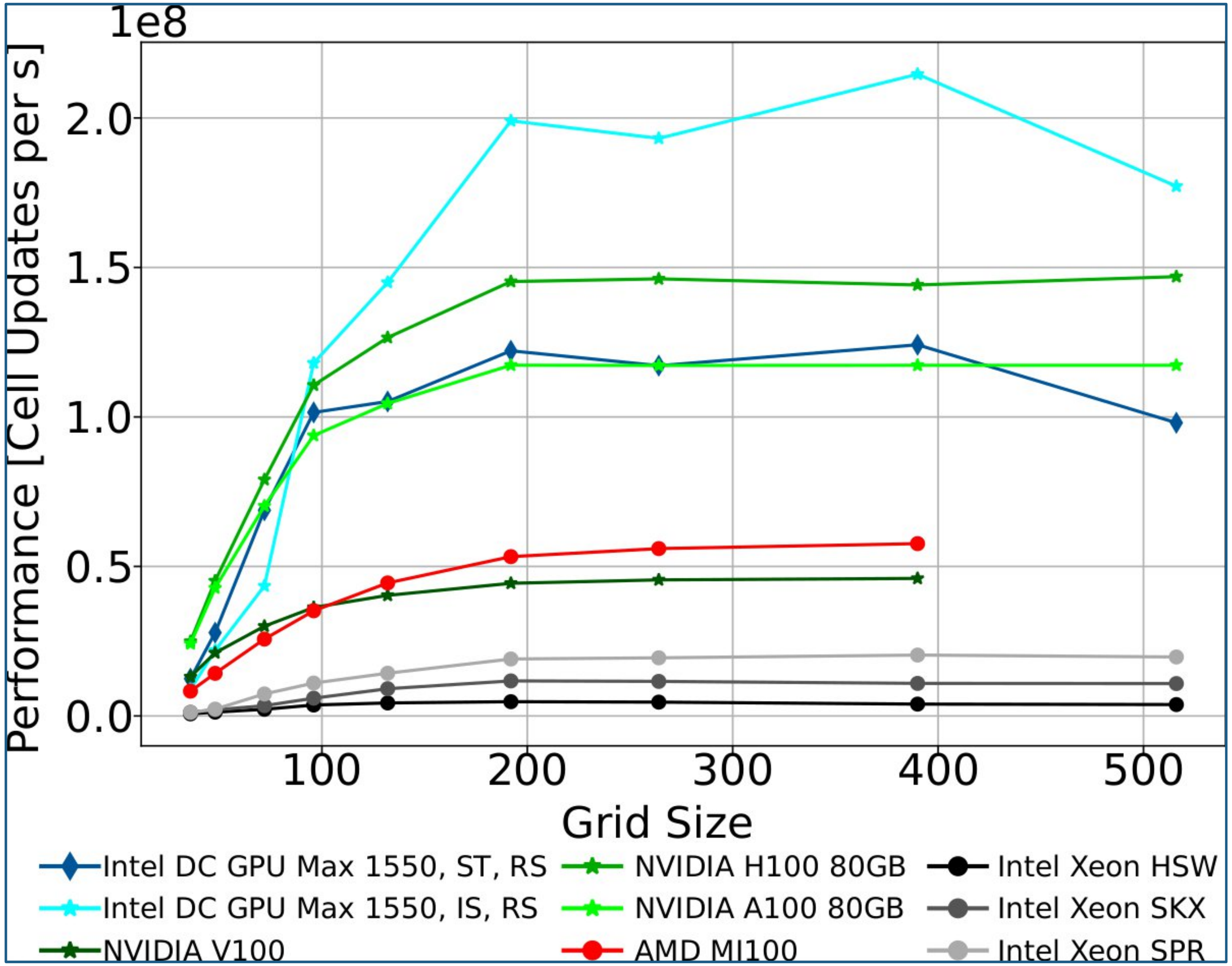


Image credit: E. Mayer for LRZ

DPEcho: SYCL + MPI ECHO porting

- Classic and relativistic MHD ported, both in Minkowski or any general relativistic metric
- Showing MHD waves test
- SYCL / Intel DPC++ with MPI, CMake
- Improved performance on CPU + GPU. Targeting next-gen Intel GPU (PVC +)



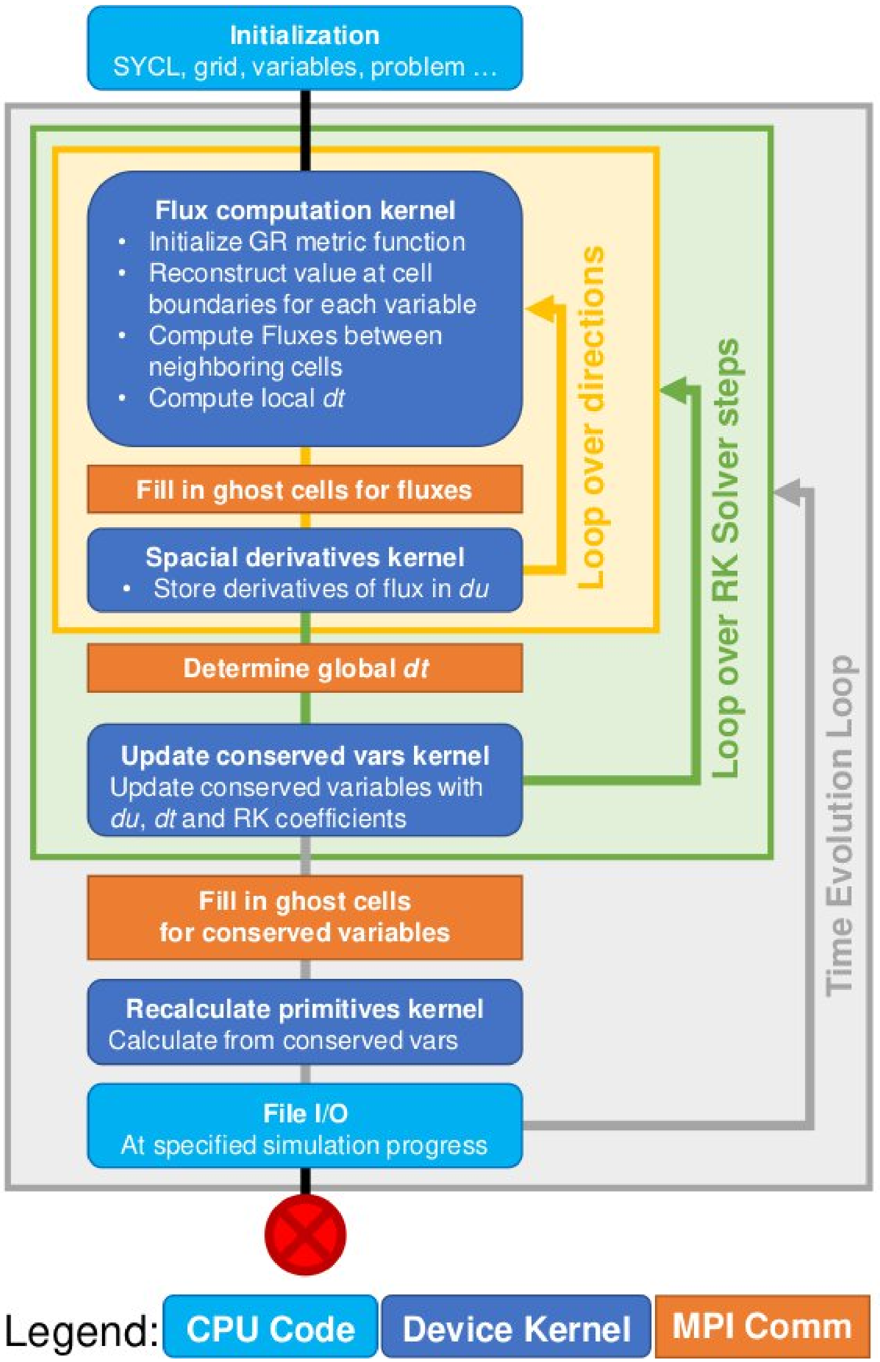
Developers: S. Cielo, A. Pöppl, M. Egelhofer, L. Del Zanna (University of Florence), M. Bugli (CEA-Saclay)

Experiences from a SYCL rewriting

- Reworking main algorithm around `sycl::queue`
- USM (readabilty, compactness, performance)
  => care never to touch data on host
- Built-in reductions (simplified in SYCL2020)

Pros & Cons

- Solid, time-proof codebase won‘t limit offload performance
- Easy performance tuning, SYCL updates
- GPU-aware MPI (coming soon!)
- Large initial time investment
- Need to port decades of accessory features
- Other roads are sometimes possible



A code extract (abridged): grid flux computation

host code

```
mysycl::gpu_selector  sDev;          mysycl::queue qDev(sDev);  // Example of device and queue
// -- Allocations: using Unified Shared Memory (USM) : variables, fluxes, …
double v[i] = malloc_shared<double>( FLD_TOT*Ncell, qDev);
double f[i] = malloc_device<double>( FLD_TOT*Ncell, qDev);  [ ... ]
//-- SYCL ranges and related accessories
range<3> rStd  = range(grid.n[0], grid.n[1], grid.n[2]),  rLoc  = range(8, 8, 8);
auto maxReduction = sycl::reduction(aMax+directionIndex, sycl::maximum<field>());
//-- Code loops: time evolution, Runge-Kutta method, loop over XYZ
while(t <= tMax){        for (int irk=0; irk<NRK; irk++){        for(unsigned direction=0; direction<3; direction++){
qDev.parallel_for(nd_range<3>(rFlux, rLoc), maxReduction, [=](nd_item<3> it, auto &max) { // loop-like: range, item, index.
                                                                                          // pragma-like: reduction, size
```

device code

```
  [ ... ]  // E.g. allocate Local variables on GPU registers
  // Parallel Kernels
  holibRec(myId, v[i], dStride, vRecL, vRecR);          // 1D Stencil
  Metric g(xCenter, yCenter, zCenter);                  // Compute the metric
  physicalFlux(directionIndex, g, vRecL, vRecR, ...); // Hotspot
  [ ... ]
  max.combine(localMax);   // Reduction for timestepping
```

```
 };  qDev.wait_and_throw(); // "Barrier"
```

host code

```
[ ... ]  } } }
[ ... ] // BC exchange, RK scheme, variable evolution
free(v  , qDev);    free(f  , qDev);
[ ... ]
```

Legend
- Main feature (orange)
- Keep an eye on it (yellow)

Outlook

Performance Optimisation

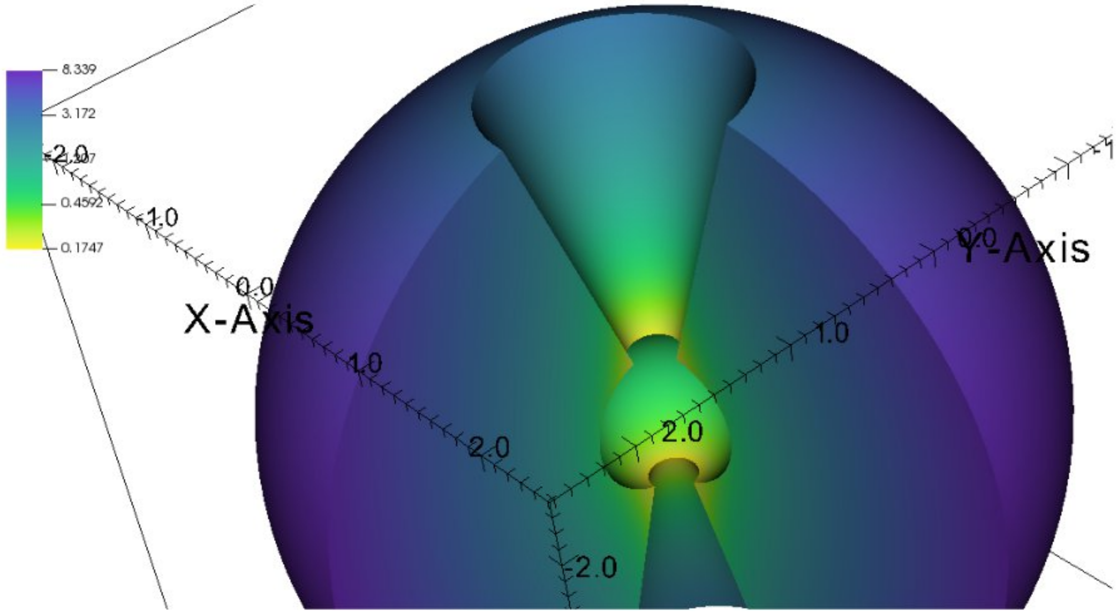
Your application may underutilize the GPU.

Run a GPU Offload (Preview) or a GPU Compute/Media Hotspots (Preview) analysis with VTune Profiler to discover how to better utilize the GPU.

	Current run	Target	Tuning Potential
MPI Time	13.26%	<10%	
Memory Stalls	20.9%	<20%	
Disk I/O Bound	10.81%	<10%	
GPU Utilization when Busy	28.2%	>80%	

DPEcho is natively instrumented for **profiling** with the oneAPI tools *VTune* and *APS* (figure), also for MPI. Optimising **GPU memory and register layout** (in progress) may largely improve GPU usage. The MPI layer seems mostly limited by barriers in parallel logging.

Black Hole Spacetime



The implementation of a rotating black hole, on a *Kommissarov* disc, in *Kerr-Schild spherical co-ordinates* (figure) is currently in progress. **We seek to involve domain scientists** for $\vec{B}$ field divergence-cleaning, and actual research runs.

Resources

lrz.de

https://github.com/LRZ-BADW/DPEcho

https://www.intel.com/content/www/us/en/developer/tools/oneapi/toolkits.html

https://www.khronos.org/sycl/