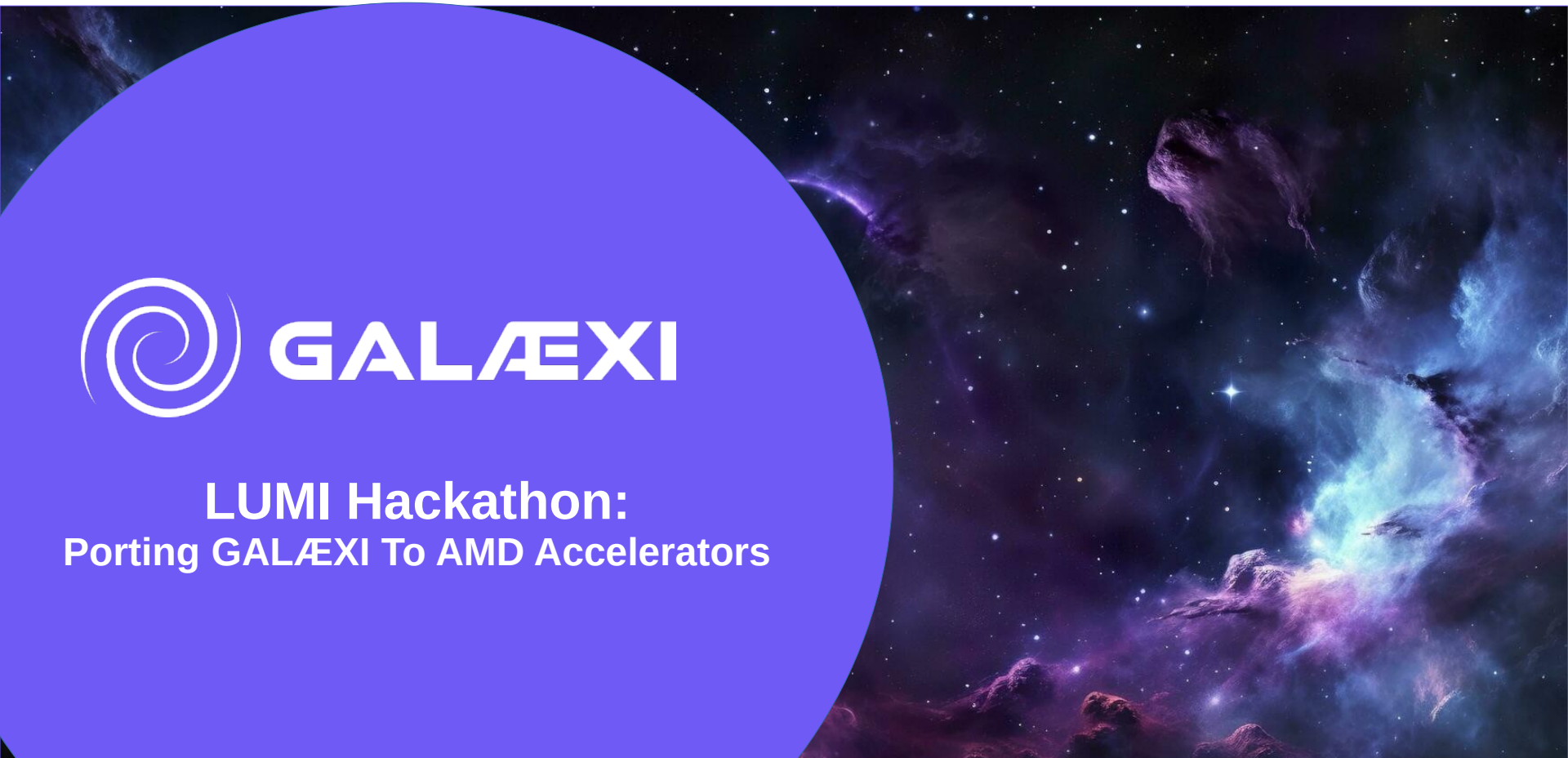
University of Stuttgart
Institute of Aerodynamics and Gas Dynamics

NRG

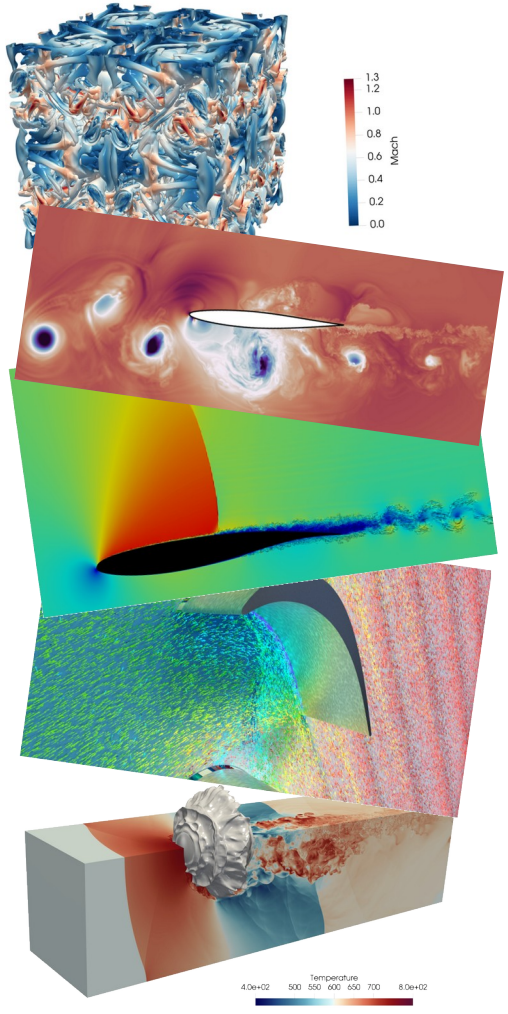


GALÆXI

LUMI Hackathon:
Porting GALÆXI To AMD Accelerators

GALÆXI

- Member of the code framework FLEXI with growing feature set
- Developed by the Numerics Research Group
  IAG, University of Stuttgart, Germany
- OpenSource HPC solver for unsteady compressible Navier–Stokes eq.
- High Order Discontinuous Galerkin Spectral Element Method

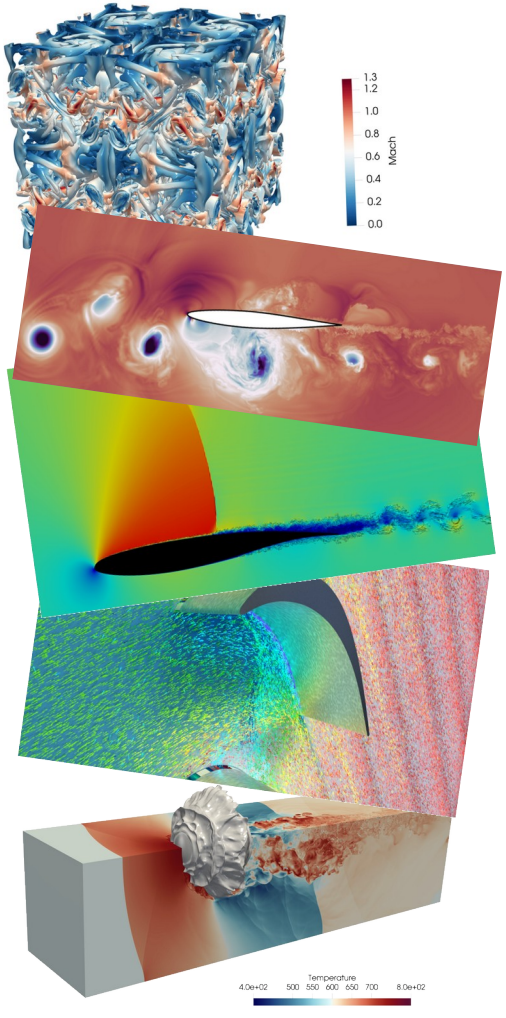


https://numericsresearchgroup.org

https://github.com/flexi-framework/galaexi

GALÆXI

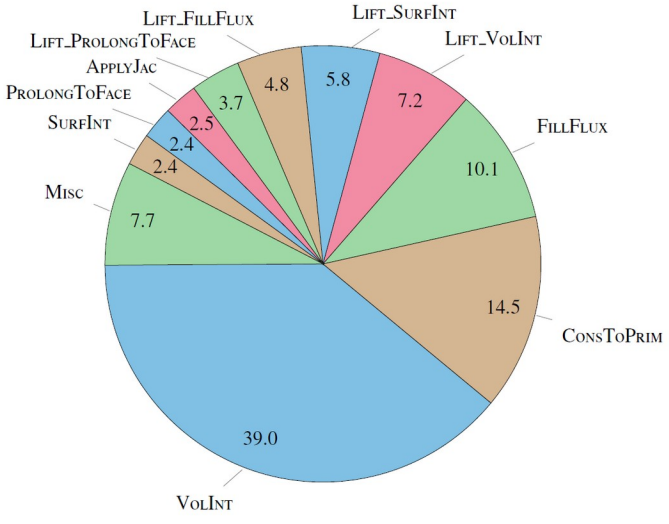
- Member of the code framework FLEXI with growing feature set
- Developed by the Numerics Research Group
  IAG, University of Stuttgart, Germany
- OpenSource HPC solver for unsteady compressible Navier–Stokes eq.
- High Order Discontinuous Galerkin Spectral Element Method
- Current feature set
  - Supported: NVIDIA (CUDA Fortran)
  - Running CI/CD pipeline
  - Gauss / Gauss-Lobatto points
  - Variable polynomial degree
  - Several Riemann solv. (comp. flag)
  - Split DG (several types, comp. flag)
  - Boundary conditions
  - MPI parallelized
  - Shock capturing
  - Sponge zones
  - Testcases (TGV, channel, …)
  - …



GALÆXI: DGSEM

- Semi-discrete formulation of the DGSEM (weak form):

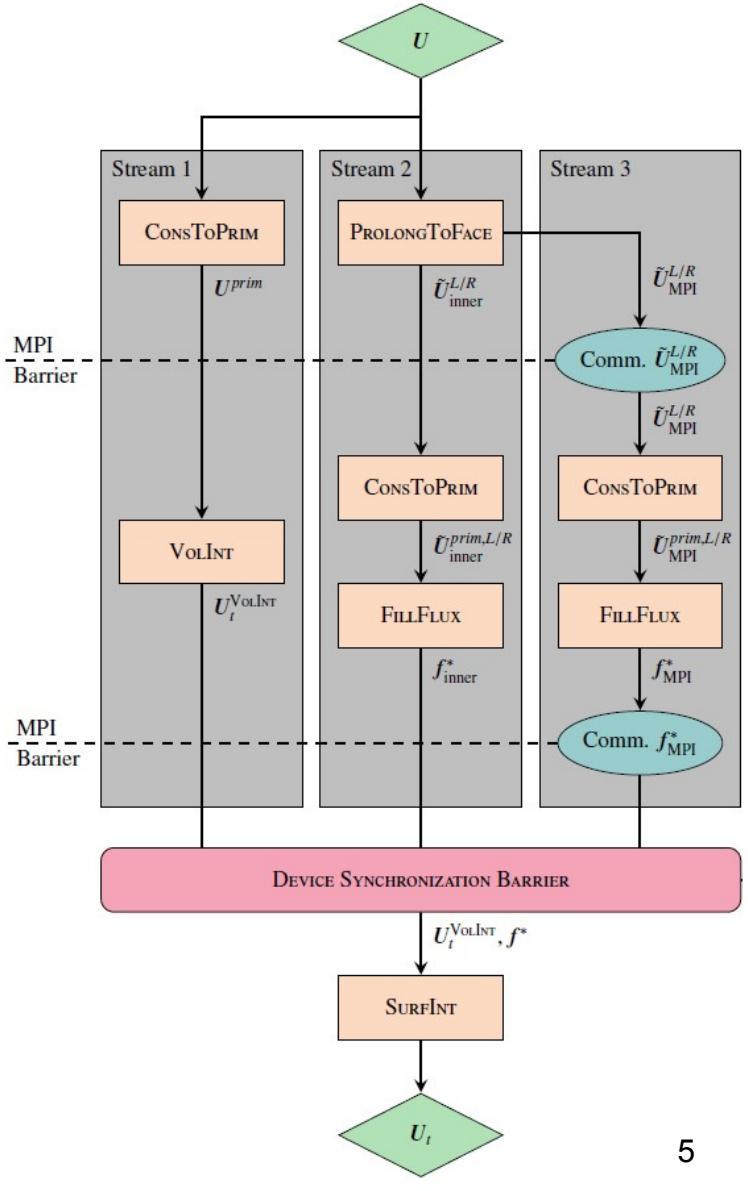
$$\frac{\partial \hat{\boldsymbol{U}}_{ijk}}{\partial t} = \overbrace{-\frac{1}{\mathcal{J}_{ijk}}}^{\text{ApplyJac}} \Bigg[\underbrace{\sum_{\alpha=0}^{N} \mathcal{F}^{1}_{\alpha jk} \hat{D}_{i\alpha} - \sum_{\beta=0}^{N} \mathcal{F}^{2}_{i\beta k} \hat{D}_{j\beta} - \sum_{\gamma=0}^{N} \mathcal{F}^{3}_{ij\gamma} \hat{D}_{k\gamma}}_{\text{VolInt}} + \underbrace{\left(\overbrace{[\boldsymbol{f}^* \hat{s}]^{\xi^+}_{jk}}^{\text{FillFlux}} \hat{\ell}^+_i - \overbrace{[\boldsymbol{f}^* \hat{s}]^{\xi^-}_{jk}}^{\text{FillFlux}} \hat{\ell}^-_i \right) + \left([\boldsymbol{f}^* \hat{s}]^{\eta^+}_{ik} \hat{\ell}^+_j - [\boldsymbol{f}^* \hat{s}]^{\eta^-}_{ik} \hat{\ell}^-_j \right) + \left([\boldsymbol{f}^* \hat{s}]^{\zeta^+}_{ij} \hat{\ell}^+_k - [\boldsymbol{f}^* \hat{s}]^{\zeta^-}_{ij} \hat{\ell}^-_k \right)}_{\text{SurfInt}} \Bigg]$$



Routine	Vol/Surf	DOF-local	Lift_*	Operations	Explanation
ConsToPrim	Surf, Vol	YES	NO	$\mathcal{O}(N^{2,3})$	Computes primitive variables $\boldsymbol{U}^{prim}$ from state $\boldsymbol{U}$.
VolInt	Vol	NO	YES	$\mathcal{O}(N^4)$	Evaluates volume fluxes $\mathcal{F}$ and multiplies with $\hat{\boldsymbol{D}}$.
ProlongToFace	Vol→Surf	NO	YES	$\mathcal{O}(N^2)$	Evaluates solution at element faces $\boldsymbol{U}^{L/R}$ to compute $\boldsymbol{f}^*$.
FillFlux	Surf	YES	YES	$\mathcal{O}(N^2)$	Computes common flux $\boldsymbol{f}^*$ on faces with Riemann solver.
SurfInt	Vol←Surf	NO	YES	$\mathcal{O}(N^2)$	Computes surface integral with $\boldsymbol{f}^*$ and $\hat{\boldsymbol{\ell}}^{\pm}$.
ApplyJac	Vol	YES	YES	$\mathcal{O}(N^3)$	Applies Jacobian $\mathcal{J}$ to $\hat{\boldsymbol{U}}_t$.

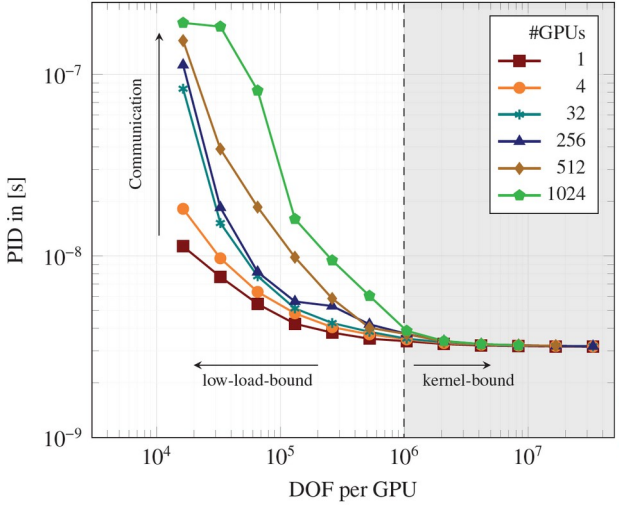
GALÆXI v1.0: Implementation

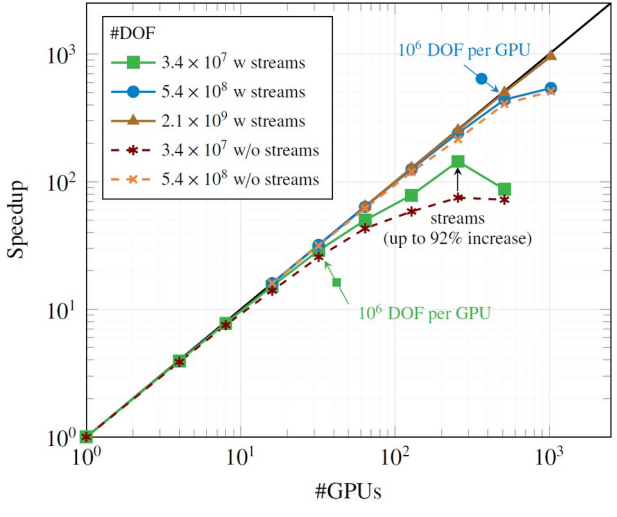
- CUDA Fortran
- Design principles
  1. Retain the general data structure and parallelization strategy of FLEXI for unstructured meshes
  2. Retain the majority of the codebase and the associated features of the original Implementation
  3. All routines called during the time-stepping are executed on the accelerator without the need of data transfers
- **Inter-GPU parallelization:** Distributed memory approach using *GPU-aware MPI* and systematic *hiding of communication* latency
- **Intra-GPU parallelization:** Launch multiple kernels concurrently and asynchronously to *maximize the occupancy* using e.g. *streams*
- **Kernel implementation:** Maximize the utilization of the available parallel resources, individual parallelization strategies for *pointwise* and *volume surface* operations

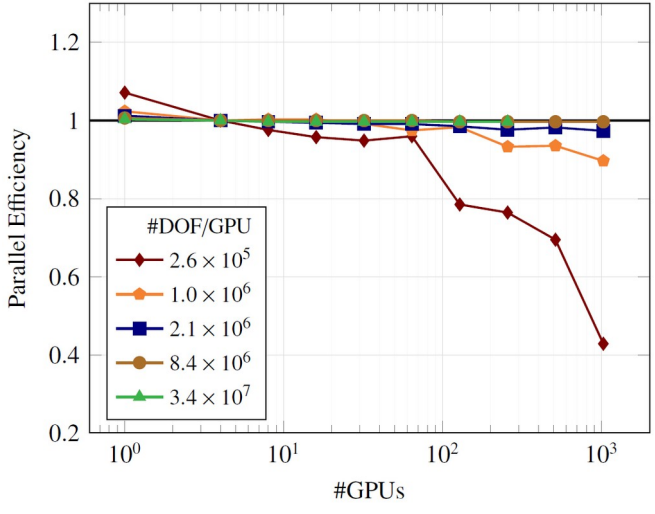


GALÆXI v1.0: Performance Evaluation

- Performance index: $$\mathrm{PID} = \frac{\text{Walltime} \times \#\text{Ranks}}{\#\text{RK-stages} \times \#\text{DOF}}$$
- PID describes the walltime required by a single rank to advance a single DOF for one stage of the explicit Runge–Kutta time-stepping







GALÆXI v2.0

GALÆXI v2.0

- Retain design principles from GALÆXI v1.0
- Allow computation on ALL architectures (including CPU)
  - Rewrite all CUDA Fortran kernels in C.
- Governing philosophy of multi-backend approach
  - Fortran = host code
  - C = device code
- Host code (Fortran) calls device code (C) through interfaces
  - Interfaces wrap CUDA/HIP API calls and Kernels
  - Hides GPU code for those who don’t want/need to interact with it
  - Easily extendable to multiple backends/vendors

```
ALLOCATE(U(  PP_nVar,0:PP_N,0:PP_N,0:PP_NZ,nElems))
CALL AllocateDeviceMemory("d_U", SIZE_C_DOUBLE, SIZE(U))

CALL CopyToDevice("d_U", U, SIZE(U))

CALL CopyFromDevice(U, "d_U", SIZE(U))
```

```
void AllocateDeviceMemory(char* dVarKey, size_t typeSize_bytes, int arraySize)
{
    void* d_arr;
#if (USE_ACCEL == ACCEL_CUDA)
    DEVICE_ERR_CHECK( cudaMalloc(&d_arr, typeSize_bytes*arraySize) );
#elif (USE_ACCEL == ACCEL_HIP)
    DEVICE_ERR_CHECK( hipMalloc(&d_arr, typeSize_bytes*arraySize) );
#elif (USE_ACCEL == ACCEL_HYBRID)
    d_arr = NULL;
#endif

    DeviceVars[dVarKey] = d_arr;
}
```

GALÆXI v2.0: Development Plan

- Phase 0 (Finished)
  - Updates to supporting function surrounding code (CI/CD, unit testing, build, etc.)
  - Incorporated device memory management methods
- Phase 1 (In Progress - Due 13.12.2024)
  - Port core kernels from CUDA Fortran to CUDA/HIP C++.
  - Code able to run inviscid, incompressible test problem on a single GPU (all architectures)
- Current Status
  - Core kernels rewritten in C
  - Computation of inviscid test problem is possible on a single NVIDIA GPU with new kernels
  - No support for AMD
- Phase 2 and Phase 3 (Planned)
  - Reintroduce multi-GPU support
  - Support for more physics and features (viscous flow, shock capturing, WMLES, etc.)


GALÆXI
Goals For Hackathon

LUMI Hackathon Goals

- Easy
  - Extend device support code (memory management, kernel launches, etc) to support HIP
  - Build with HIP compiler
  - Familiarize with AMD profiling tools
- Better
  - Complete adjustments to kernels (if needed)
  - Code can run test problem on LUMI-G nodes
- Best
  - Tuning and optimization
  - Start looking at hybrid architectures

University of Stuttgart
Institute of Aerodynamics and Gas Dynamics

THANK YOU!

NRG

GALAEXI