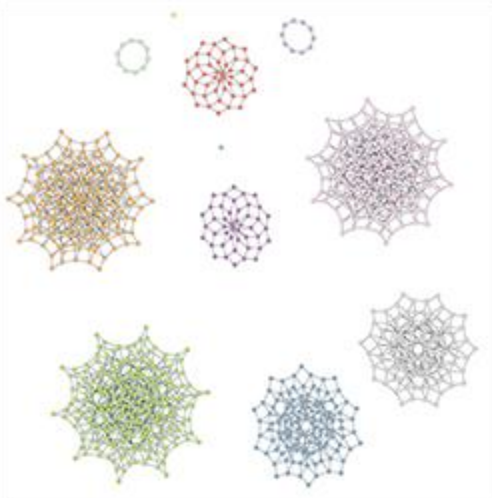
Software Technology and High Performance Computing for Simulations and Digital Twins

Dr.-Ing. Achim Basermann
Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR), Köln
Institut für Softwaretechnologie
Abteilungsleiter „High-Performance Computing“

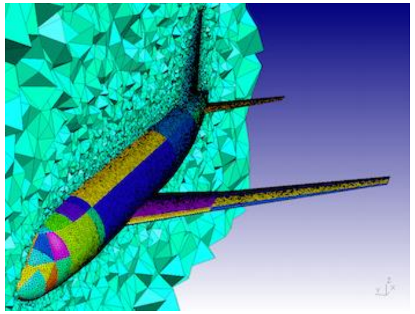
DLR

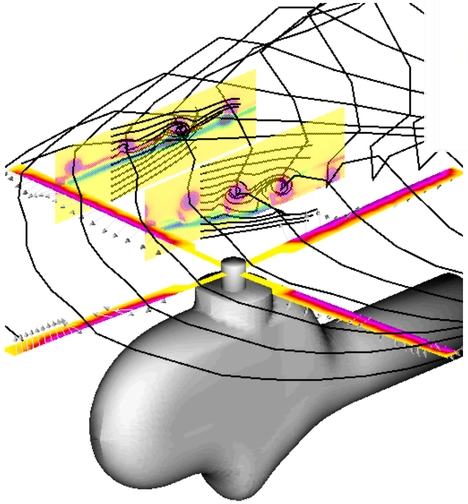
Contents



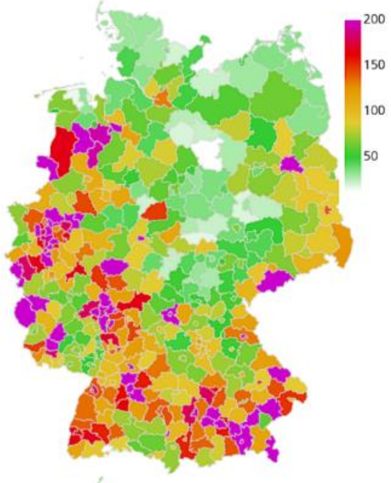
- **Survey of SC and SC-HPC**

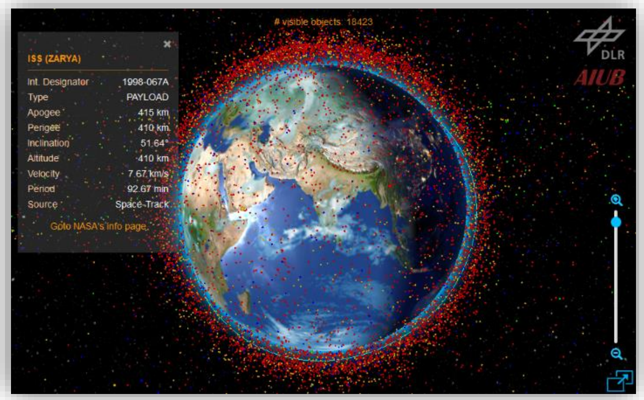
- **Solvers for Extreme Computing**



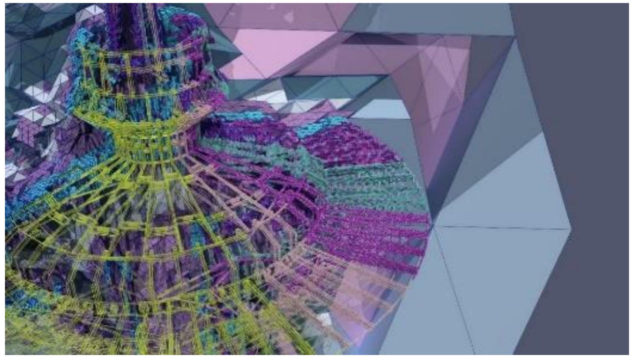


- **Parallel Frameworks**
    - Adaptive Meshes for Aeronautics and ESM
    - Helicopter Simulation
    - Pandemic Simulation



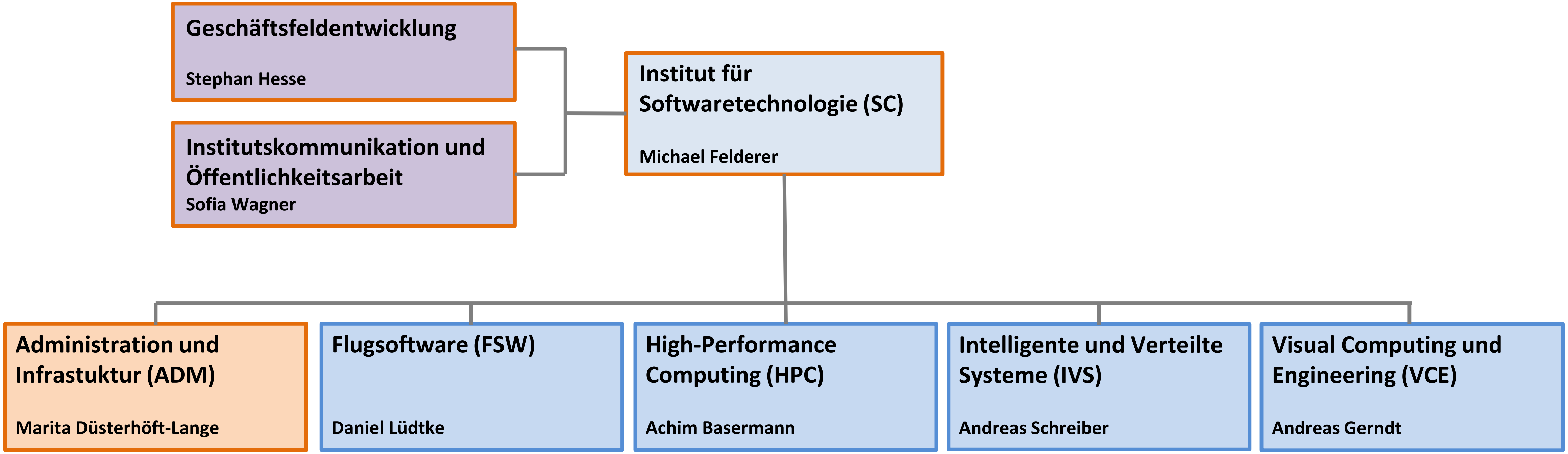


- **Space Debris**



- **Parallele Machine Learning**
    - Software Framework HeAT

Institutsstruktur



Hauptstandorte

Köln

Oberpfaffenhofen

Braunschweig

Institut für Softwaretechnologie (SC): unsere Rolle im DLR
Direktor: Prof. Dr. Michael Felderer, auch Universität zu Köln, seit 2023

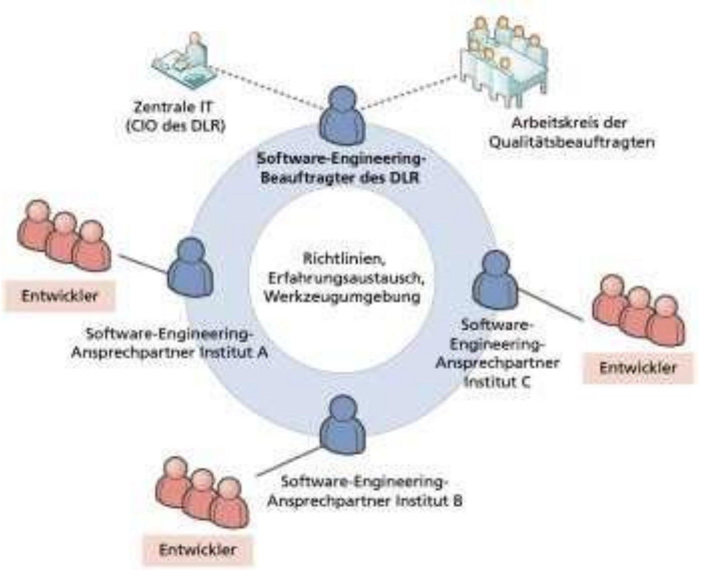
1. Erforschung neuer Softwaretechnologien für das DLR

Beispiel: Quantencomputing

2. Steigerung von Effektivität und Effizienz in DLR-Projekten durch moderne Softwaretechnologie

Beispiel: Visualisierung und Analyse hochaufgelöster Satellitendaten

3. Verbesserung der Softwaretechnik in den Instituten

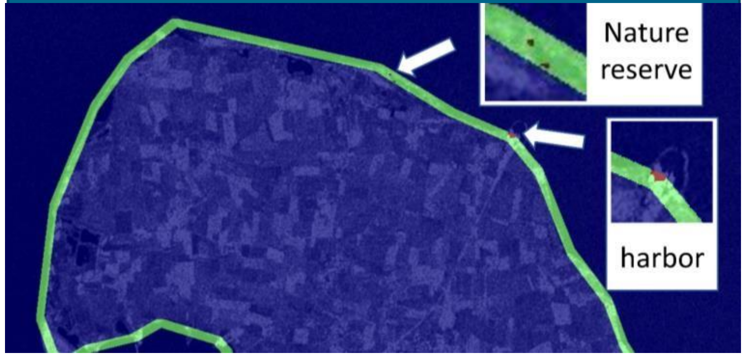


Beispiel: Erfahrungs-austausch-Workshops

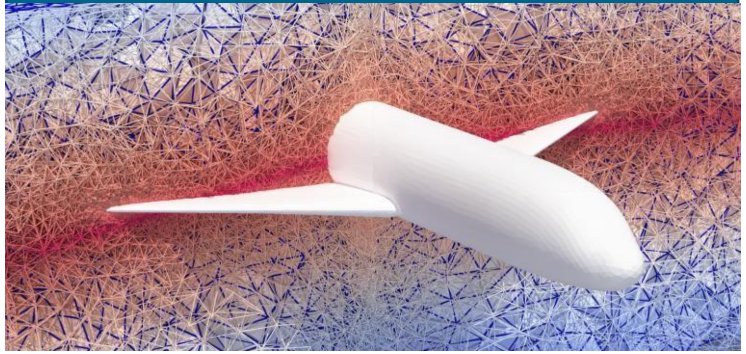
Topic Areas at the Institute of Software Technology (SC)

Dependable, Safe and Secure Software Systems

Artificial Intelligence



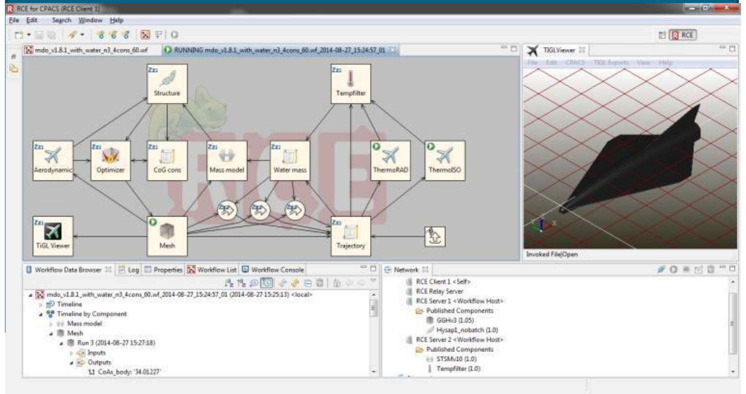
High Performance Computing and Quantum Computing

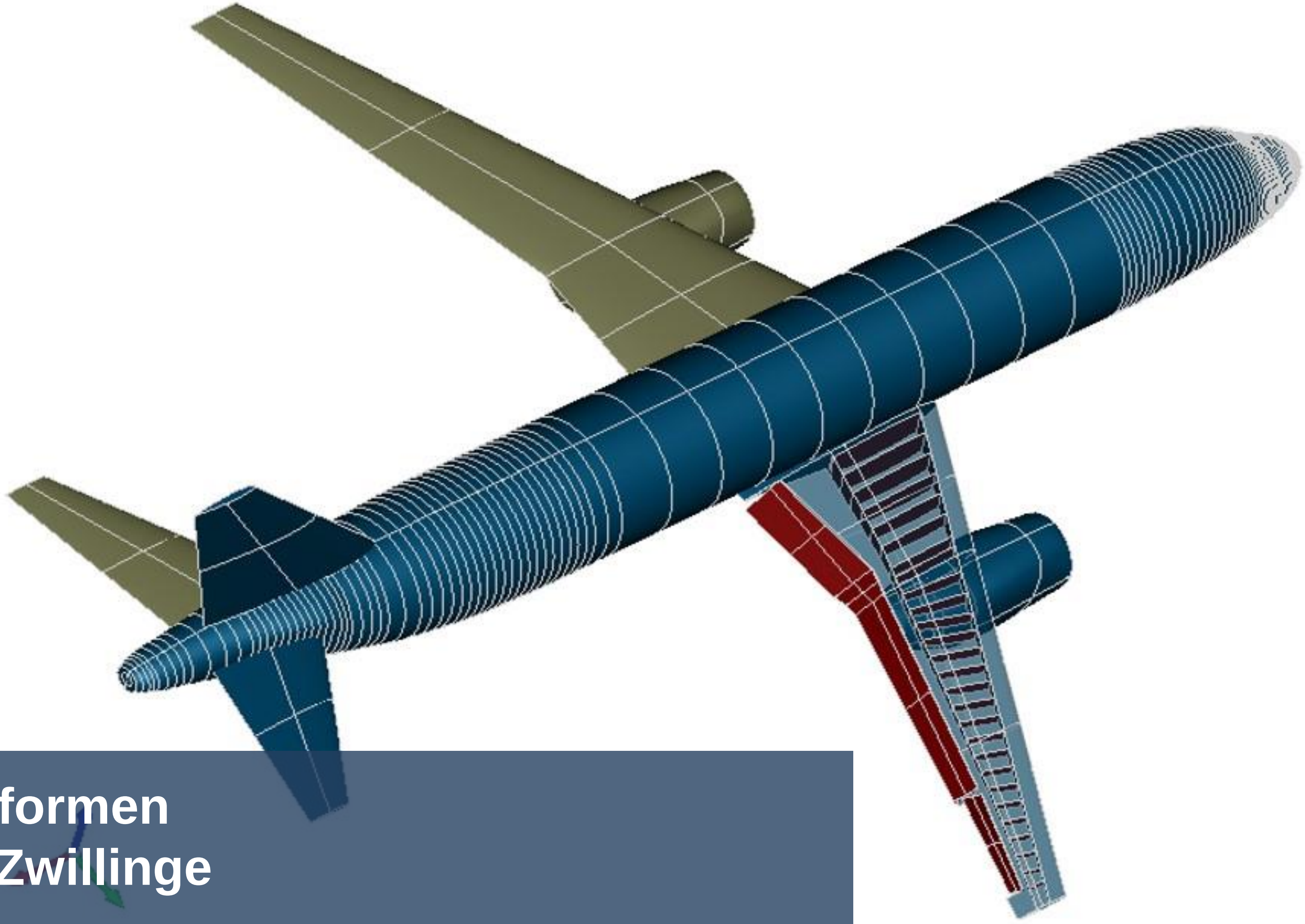


Human-System-Interaction and Visualisation

Software and Systems Engineering

Digital Platforms and Digital Twins





Digitale Plattformen und Digitale Zwillinge

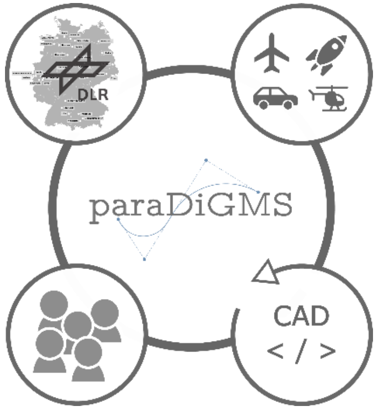
Digitale Plattformen und Digitale Zwillinge

Verteilte Integrationsumgebung für die Entwicklung von komplexen Systemen (Satelliten, Flugzeuge, Schiffe)

Parametrische Modellierung und Simulation für Luft- und Raumfahrtzeuge

Herstellung der technischen Infrastruktur für den Zugang zu den Quantencomputern der DLR QCI

RCE



CLIQUE

- Verteiltes kollaboratives Arbeiten mit Anbindung an Hochleistungsrechner oder Zugang zu den Quantencomputern des DLR
- Gehärtete Softwareinfrastrukturen für sichere Ingenieursanwendungen

Abteilung Visual Computing und Engineering

Visual Engineering

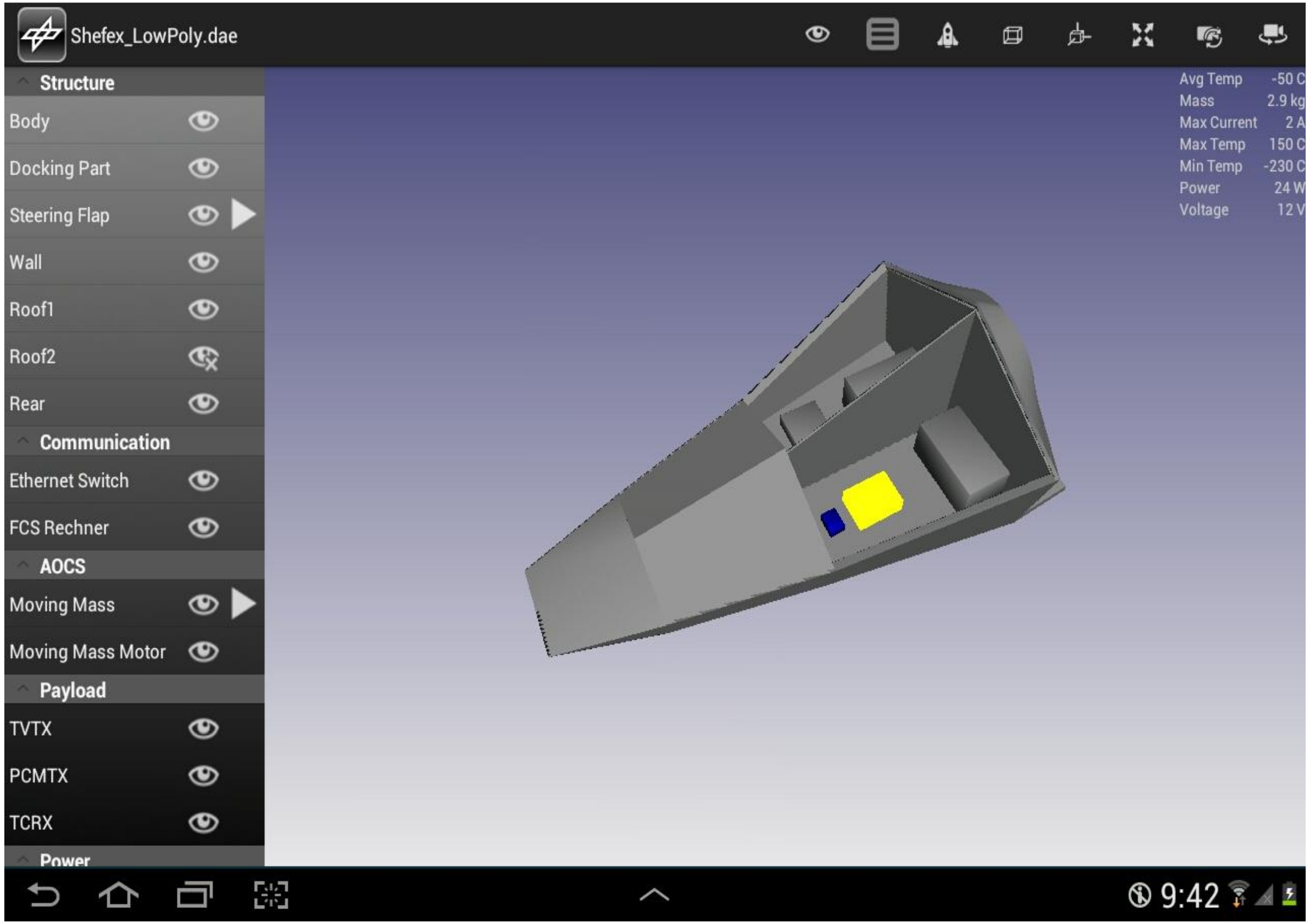
- interaktive Modellierung des Virtuellen Satelliten

Interaktive Dashboards

- für die Analyse von Emissionen

AR-basierte Analyse technischer Digitaler Zwillinge

- virtuelle Zusammenarbeit bei Wartungsaufgaben an Flugzeugen



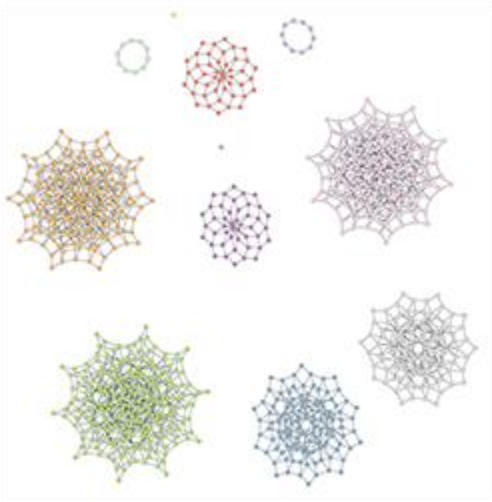
High Performance Computing
Teams

Abteilung
High-Performance Computing
Leiter: Dr.-Ing. Achim Basermann
Stellvertreter: Dr. Alexander Rüttgers

- **Simulation & Optimierung technischer Systeme** – Dr. Jan Kleinert
- **Skalierbare adaptive Gitter** – Dr. Johannes Holke
- **Prädiktive Simulationssoftware** – Dr. Martin Kühn
- **Performance Engineering für mathematische Software** – Dr.-Ing. Achim Basermann
- **Skalierbares Machine Learning** – Dr. Alexander Rüttgers
  - ↕ **Scientific Machine Learning** – Dr. Philipp Knechtges
- **Quantencomputing: Methoden & Implementierung** – Dr. Michael Epping
  - ↕ **Quantencomputing: Anwendungssoftware** – Dr. Elisabeth Lobe

Exascale computing and Performance Engineering

Solvers for Extreme Computing



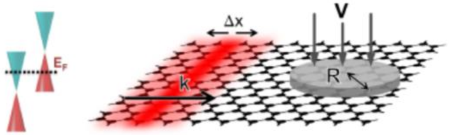
Graphvisualisierung der möglichen Zustandsänderungen des Heisenberg Spinkettenmodells

DLR

Motivation: Requirements for Exascale Computing

DFG Project ESSEX

Quantum physics/information applications



Large, Sparse

$$i\hbar \frac{\partial}{\partial t}\psi(\vec{r},t) = H\psi(\vec{r},t)$$

and beyond….

$$\boldsymbol{H\,x = \lambda\,x}$$

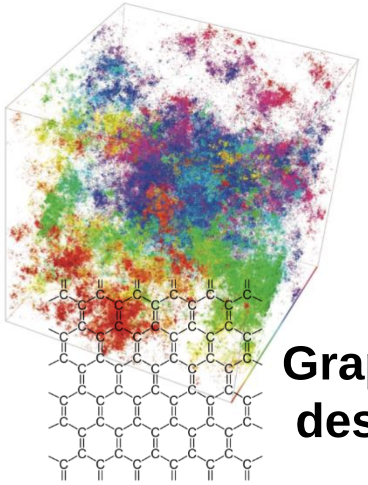
"Few" (1,…,100s) of eigenpairs

"**Bulk**" (100s,…,1000s) eigenpairs

$$\{\lambda_1, \lambda_2, \ldots, \ldots, \ldots, \ldots, \lambda_k, \ldots, \ldots, \ldots, \ldots, \lambda_{n-1}, \lambda_n\}$$

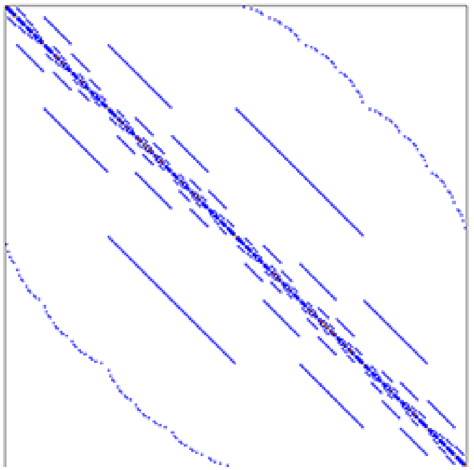
Good approximation to full spectrum (e.g. Density of States)

→ Sparse eigenvalue solvers of broad applicability



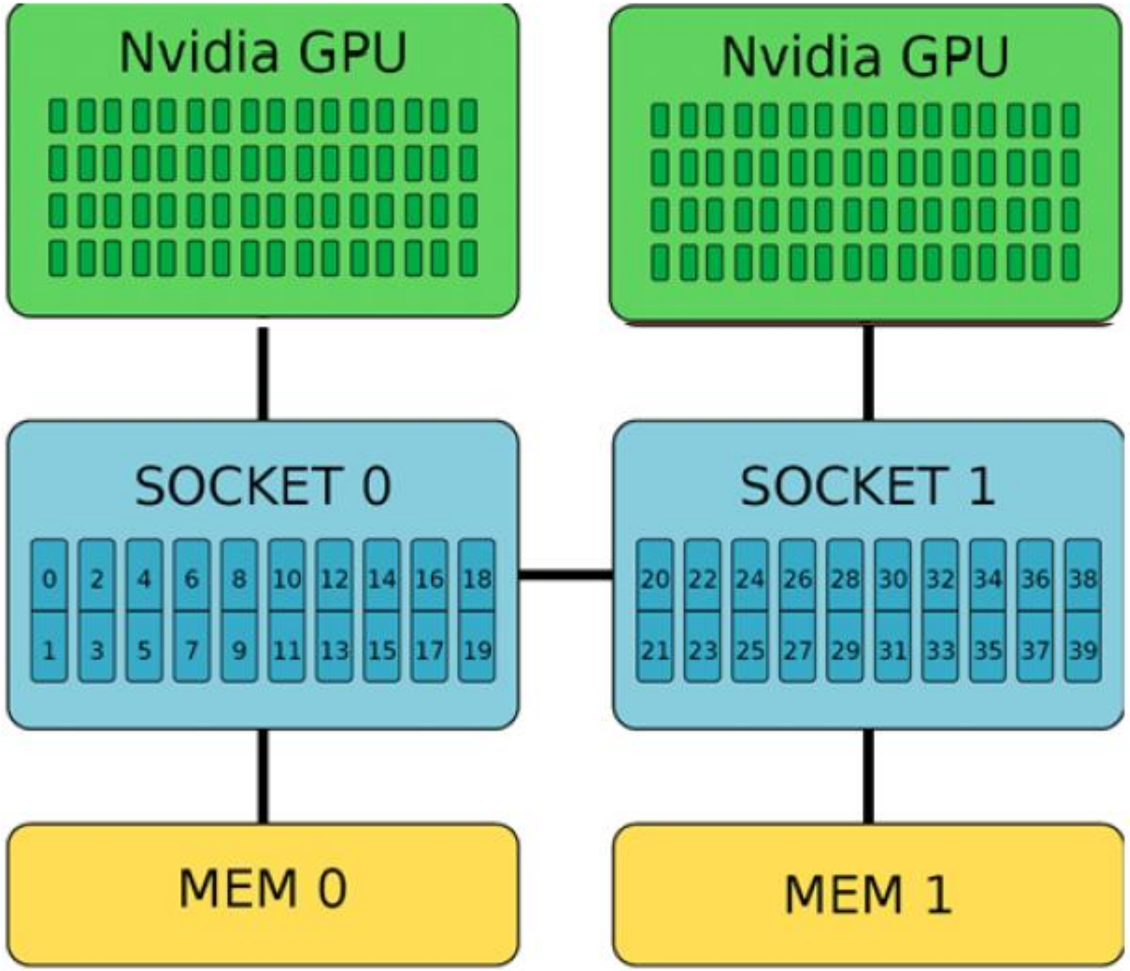
Graphen design

Sparse matrix



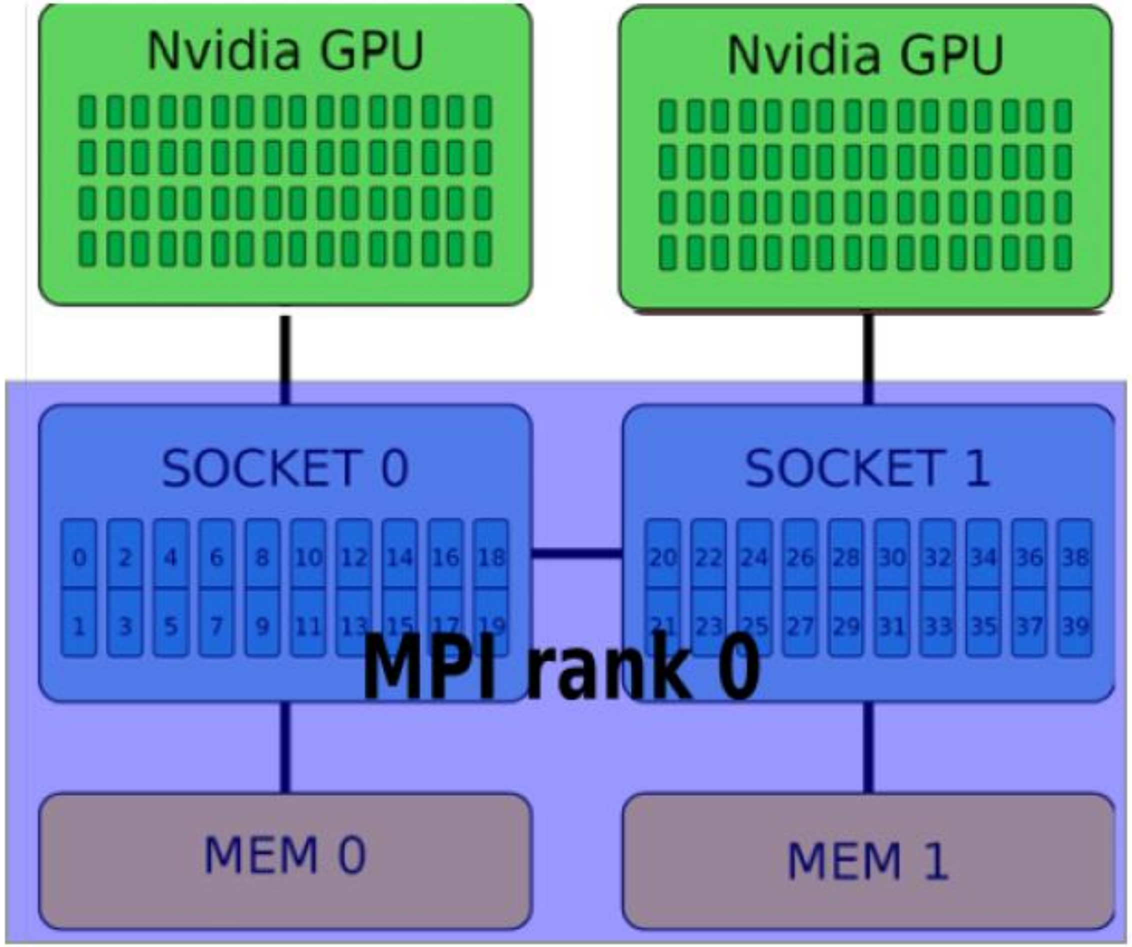
The ESSEX Software Infrastructure: MPI + X with GHOST

- System with multiple CPUs (NUMA domains) and GPUs



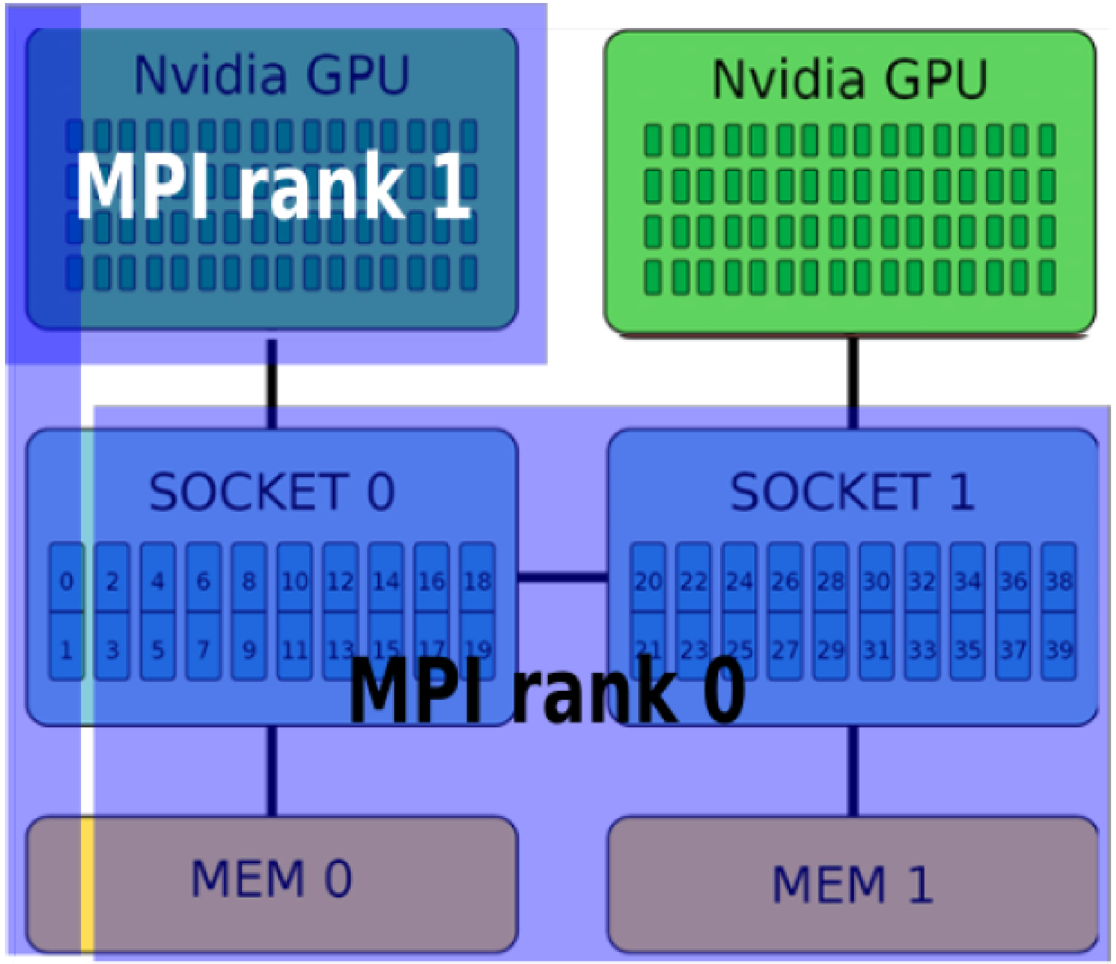
The ESSEX Software Infrastructure: MPI + X with GHOST

- System with multiple CPUs (NUMA domains) and GPUs
- -np 1: use entire CPU



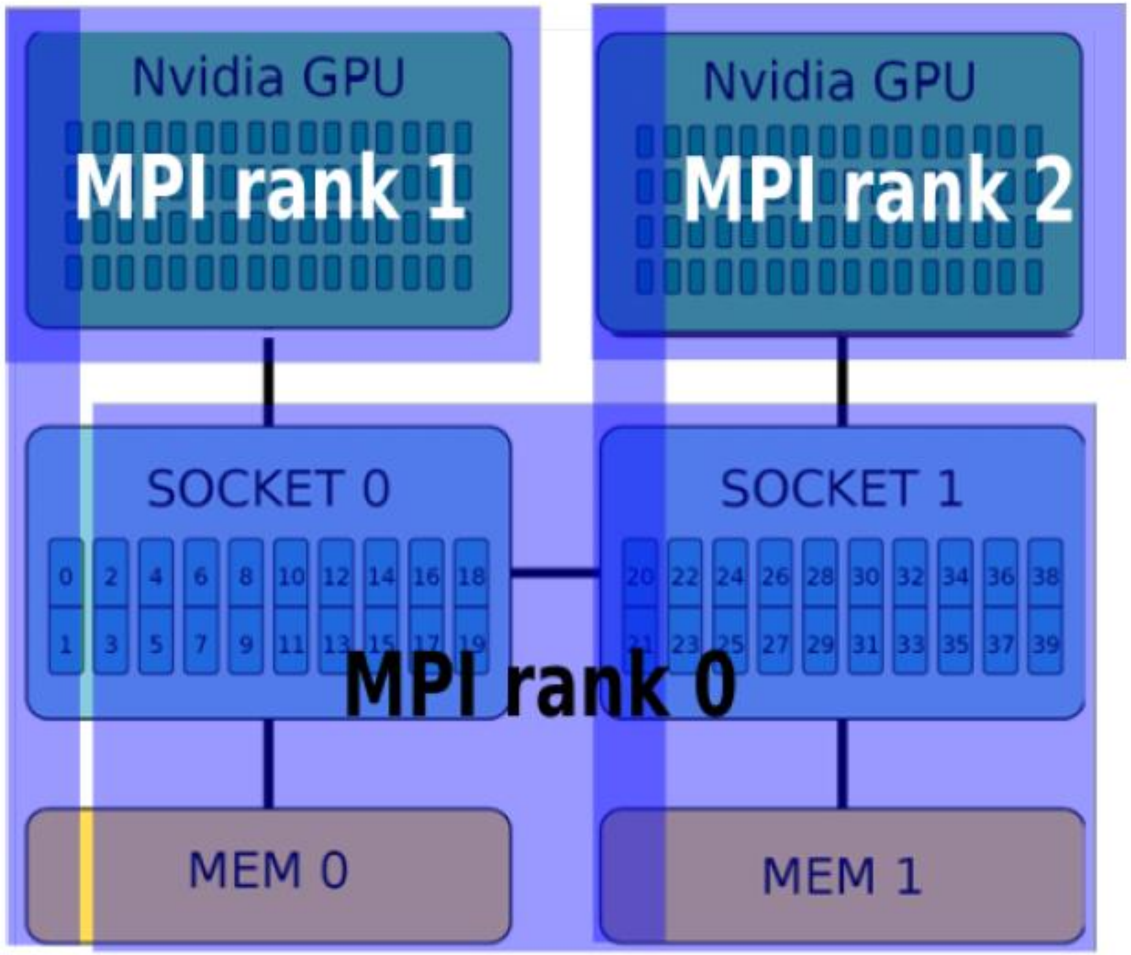
The ESSEX Software Infrastructure: MPI + X with GHOST

- System with multiple CPUs (NUMA domains) and GPUs
- -np 1: use entire CPU
- -np 2: use CPU and first GPU



The ESSEX Software Infrastructure: MPI + X with GHOST

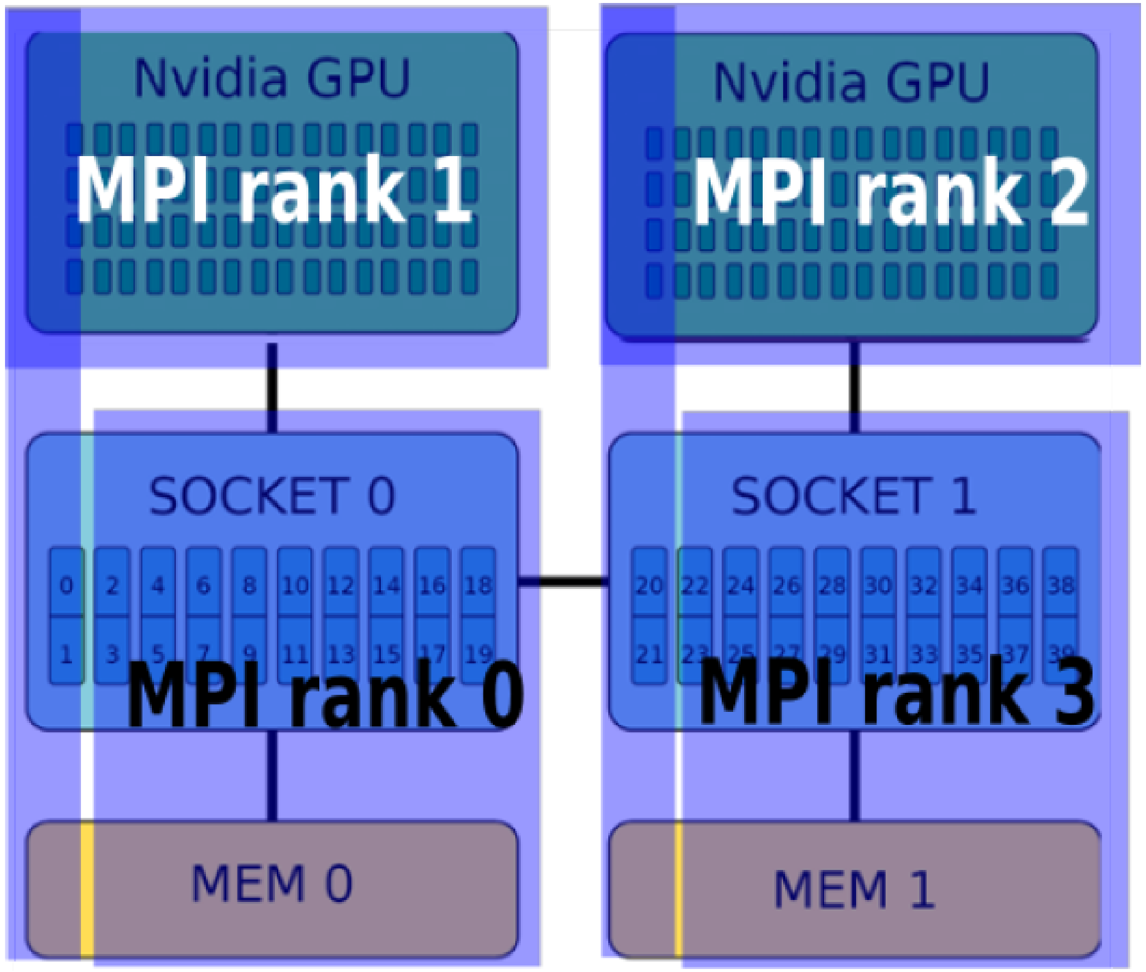
- System with multiple CPUs (NUMA domains) and GPUs
- -np 1: use entire CPU
- -np 2: use CPU and first GPU
- -np 3: use CPU and both GPUs



The ESSEX Software Infrastructure: MPI + X with GHOST

- System with multiple CPUs (NUMA domains) and GPUs
- -np 1: use entire CPU
- -np 2: use CPU and first GPU
- -np 3: use CPU and both GPUs
- -np 4: use one process per socket and one for each GPU

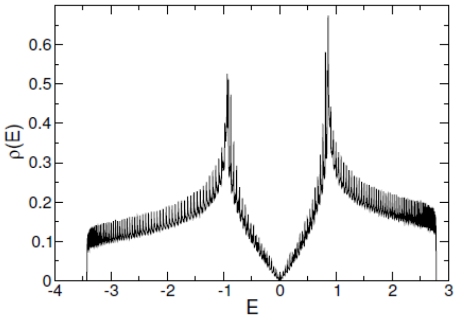
Option: distribute problem according to memory bandwidth measured

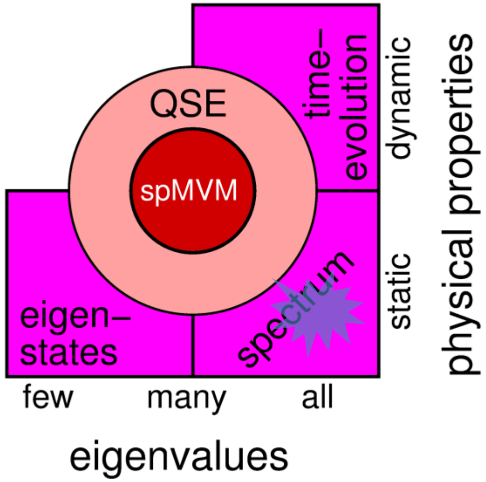


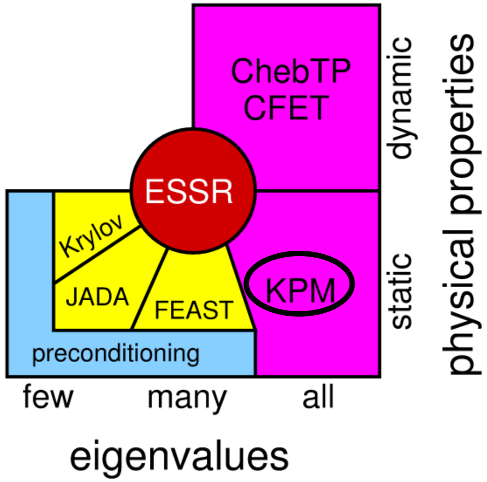
Application, Algorithm and Performance: Kernel Polynomial Method (KPM) – A Holistic View

- Compute **approximation to the complete eigenvalue spectrum** of large sparse matrix A (with $X = I$)

$$X(\omega) = \frac{1}{N}\mathrm{tr}[\delta(\omega - H)X] = \frac{1}{N}\sum_{n=1}^{N}\delta(\omega - E_n)\langle\psi_n, X\psi_n\rangle$$







The Kernel Polynomial Method (KPM)

Optimal performance exploit knowledge from all software layers!

Basic algorithm – Compute Cheyshev polynomials/moments:

for $r = 0$ to $R-1$ **do**
  $|v\rangle \leftarrow |\text{rand}()\rangle$
  Initialization steps and computation of η_0, η_1
  for $m = 1$ to $M/2$ **do**
    $\text{swap}(|w\rangle, |v\rangle)$
    $|u\rangle \leftarrow H|v\rangle$ ▷ spmv() Sparse matrix vector multiply
    $|u\rangle \leftarrow |u\rangle - b|v\rangle$ ▷ axpy() Scaled vector addition
    $|w\rangle \leftarrow -|w\rangle$ ▷ scal() Vector scale
    $|w\rangle \leftarrow |w\rangle + 2a|u\rangle$ ▷ axpy() Scaled vector addition
    $\eta_{2m} \leftarrow \langle v|v\rangle$ ▷ nrm2() Vector norm
    $\eta_{2m+1} \leftarrow \langle w|v\rangle$ ▷ dot() Dot Product
  end for
end for

Application: Loop over random initial states

Algorithm: Loop over moments

Building blocks: (Sparse) linear algebra library

The Kernel Polynomial Method (KPM)

Optimal performance exploit knowledge from all software layers!

Basic algorithm – Compute Cheyshev polynomials/moments:

for $r = 0$ to $R-1$ **do**
  $|v\rangle \leftarrow |\text{rand}()\rangle$
  Initialization steps and computation of η_0, η_1
  for $m = 1$ to $M/2$ **do**
    $\text{swap}(|w\rangle, |v\rangle)$
    $|u\rangle \leftarrow H|v\rangle$ ▷ spmv()
    $|u\rangle \leftarrow |u\rangle - b|v\rangle$ ▷ axpy()
    $|w\rangle \leftarrow -|w\rangle$ ▷ scal()
    $|w\rangle \leftarrow |w\rangle + 2a|u\rangle$ ▷ axpy()
    $\eta_{2m} \leftarrow \langle v|v\rangle$ ▷ nrm2()
    $\eta_{2m+1} \leftarrow \langle w|v\rangle$ ▷ dot()
  end for
end for

for $r = 0$ to $R-1$ **do**
  $|v\rangle \leftarrow |\text{rand}()\rangle$
  Initialization steps and computation of η_0, η_1
  for $m = 1$ to $M/2$ **do**
    $\text{swap}(|w\rangle, |v\rangle)$
    $|w\rangle = 2a(H - b\mathbb{1})|v\rangle - |w\rangle$ &
    $\eta_{2m} = \langle v|v\rangle$ &
    $\eta_{2m+1} = \langle w|v\rangle$ ▷ aug_spmv()
  end for

Augmented Sparse Matrix Vector Multiply

The Kernel Polynomial Method (KPM)

Optimal performance exploit knowledge from all software layers!

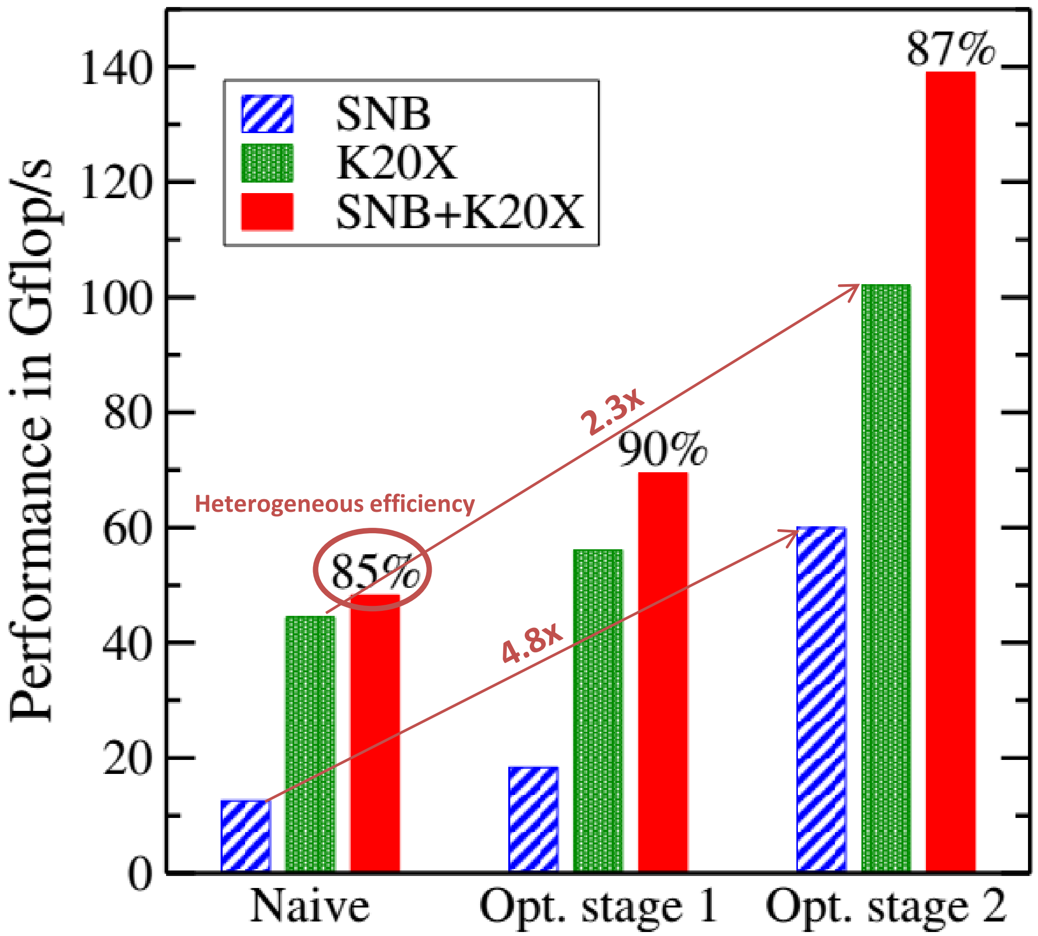
Basic algorithm – Compute Cheyshev polynomials/moments:

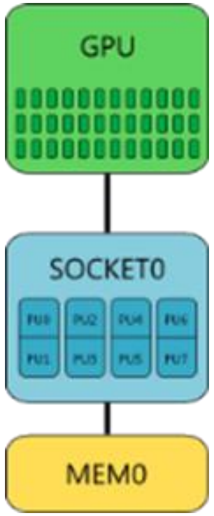
for $r = 0$ to $R-1$ **do**
  $|v\rangle \leftarrow |\text{rand}()\rangle$
  Initialization steps and computation of η_0, η_1
  for $m = 1$ to $M/2$ **do**
    $\text{swap}(|w\rangle, |v\rangle)$
    $|w\rangle = 2a(H - b\mathbb{1})|v\rangle - |w\rangle$ &
      $\eta_{2m} = \langle v|v\rangle$ &
      $\eta_{2m+1} = \langle w|v\rangle$ ▷ aug_spmv()
  end for

$|V\rangle := |v\rangle_{0..R-1}$ ▷ Assemble vector blocks
$|W\rangle := |w\rangle_{0..R-1}$
$|V\rangle \leftarrow |\text{rand}()\rangle$
Initialization steps and computation of μ_0, μ_1
for $m = 1$ to $M/2$ **do**
  $\text{swap}(|W\rangle, |V\rangle)$
  $|W\rangle = 2a(H - b\mathbb{1})|V\rangle - |W\rangle$ &
    $\eta_{2m}[:] = \langle V|V\rangle$ &
    $\eta_{2m+1}[:] = \langle W|V\rangle$ ▷ aug_spmmv()
end for

Augmented Sparse Matrix
Multiple Vector Multiply

KPM: Heterogenous Node Performance



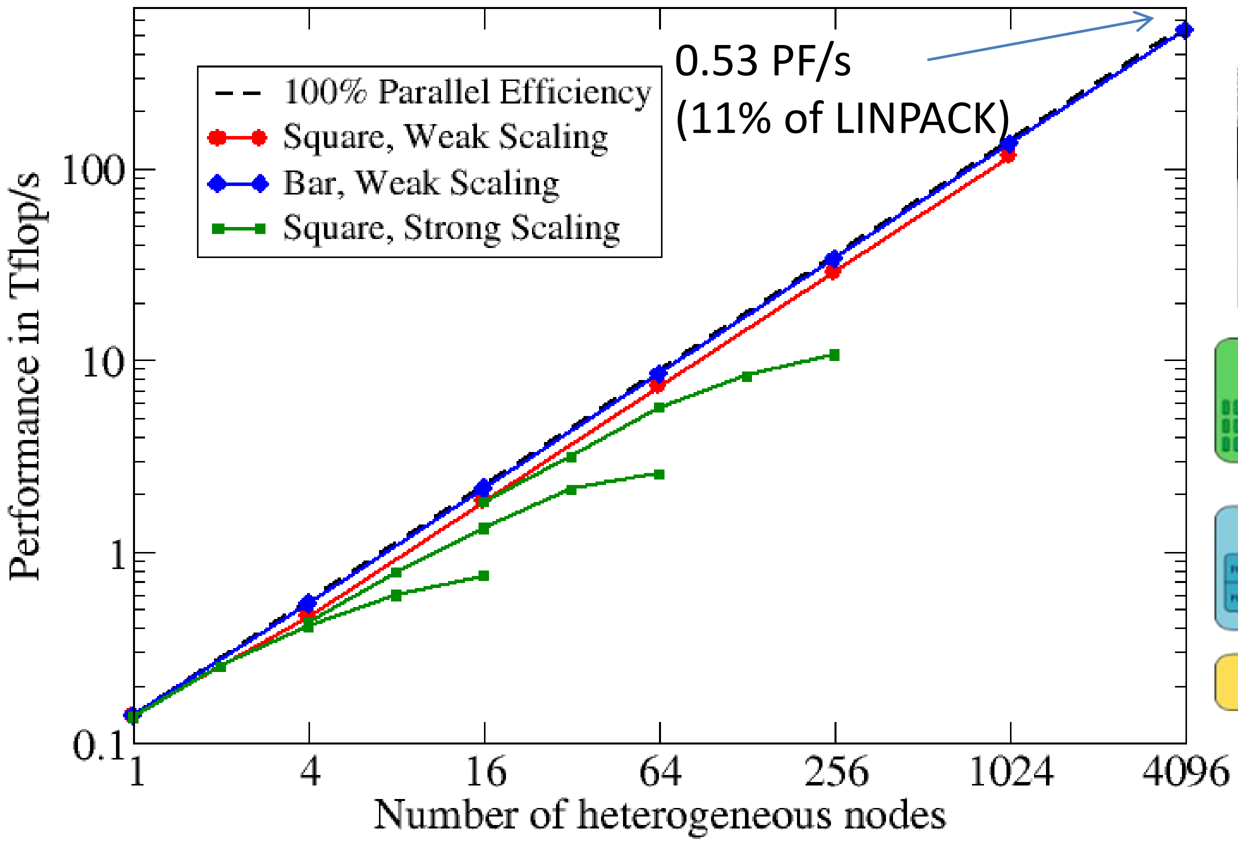


NVIDIDA K20X

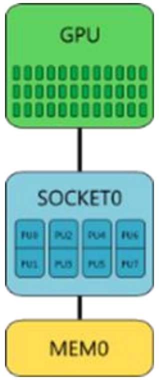
Intel
Xeon E5-2670 (SNB)

- Topological Insulator Application
- Double complex computations
- Data parallel static workload distribution

KPM: Large Scale Heterogenous Node Performance



CRAY XC30 – PizDaint*



- 5272 nodes
- Peak: 7.8 PF/s
- LINPACK: 6.3 PF/s
- Largest system in Europe

Performance Engineering of the Kernel Polynomial Method on Large-Scale CPU-GPU Systems
M. Kreutzer, A. Pieper, G. Hager, A. Alvermann, G. Wellein and H. Fehske, IEEE IPDPS 2015

*Thanks to CSCS/T. Schulthess for granting access and compute time

How to ensure the quality of the ESSEX software: Basics

- **Git** for distributed software developement
    - **Merge-request workflow** for code review; changes only in branches
    - Visualization of git repository development

- Own MPI extension for **Google Test**

- Realization of **continuous-integration** with Jenkins server

Towards common standards and community software for extreme-scale computing

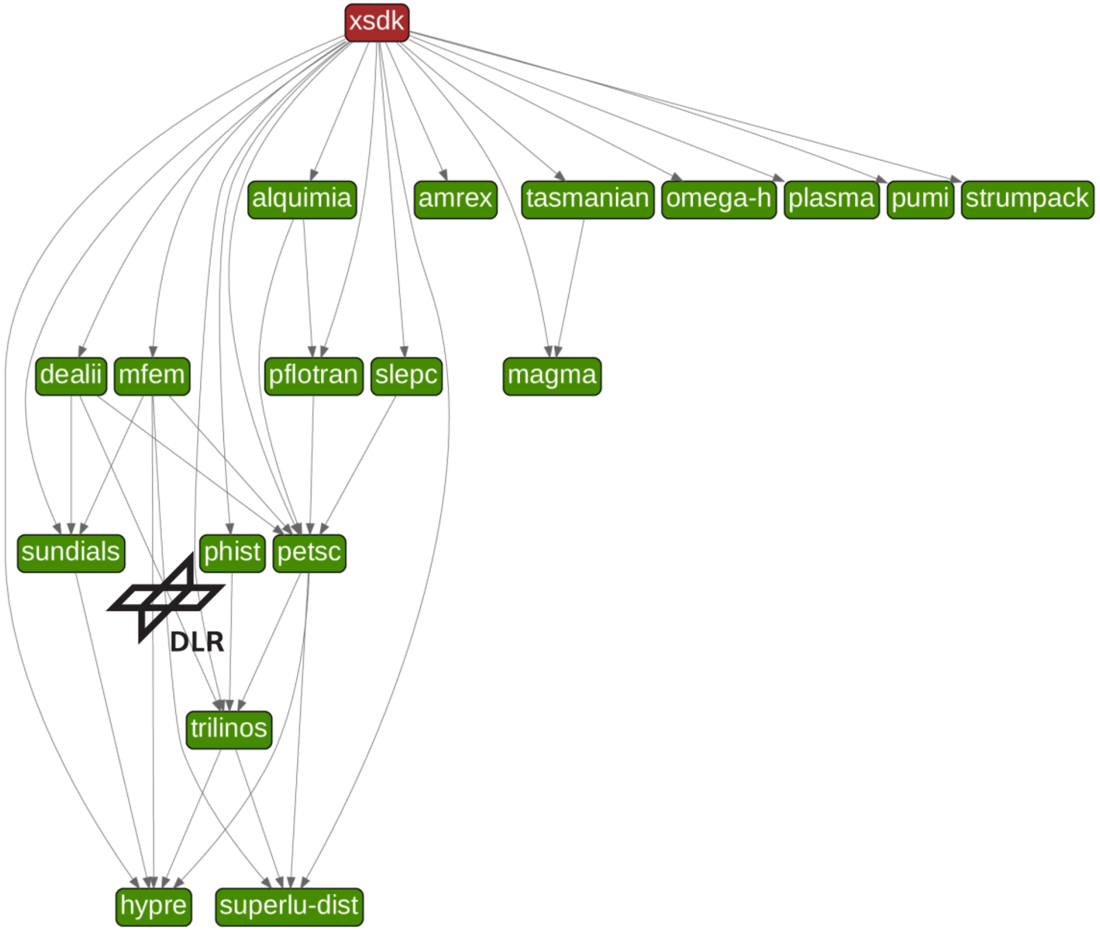
As we approach the Exa-scale, requirements on robustness, portability, scalability and interoperability of scientific software are rapidly increasing

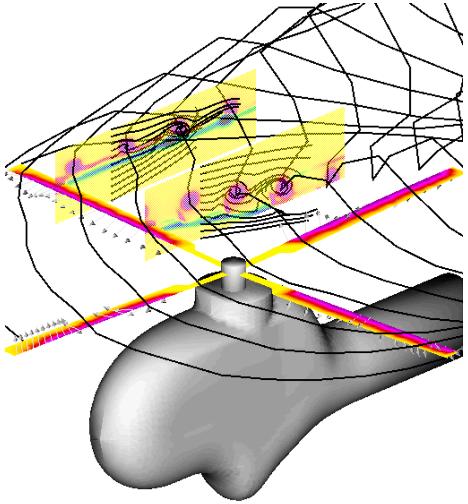
xSDK: Extreme-scale Scientific Software Development Kit

- Joint open-source effort of DOE labs and other international teams (https://xsdk.info/)

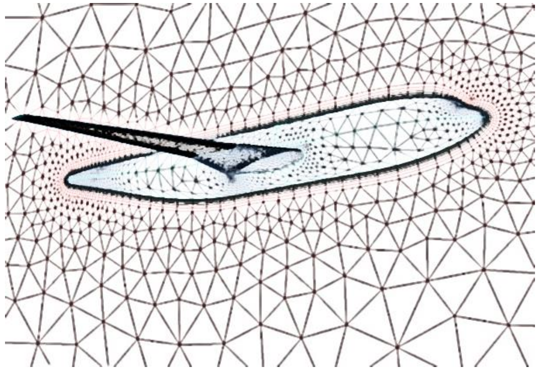
- DLR contributes a hybrid-parallel library for solving sparse eigenvalue problems on heterogenous supercomputers
- (https://bitbucket.org/essex/phist/)

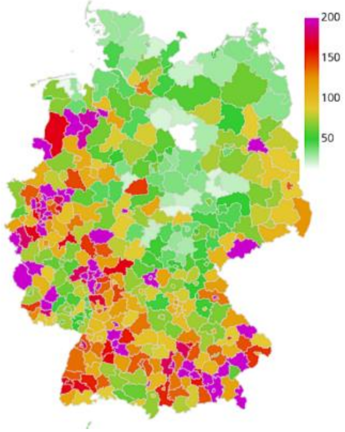
Towards common standards and community software for extreme-scale computing

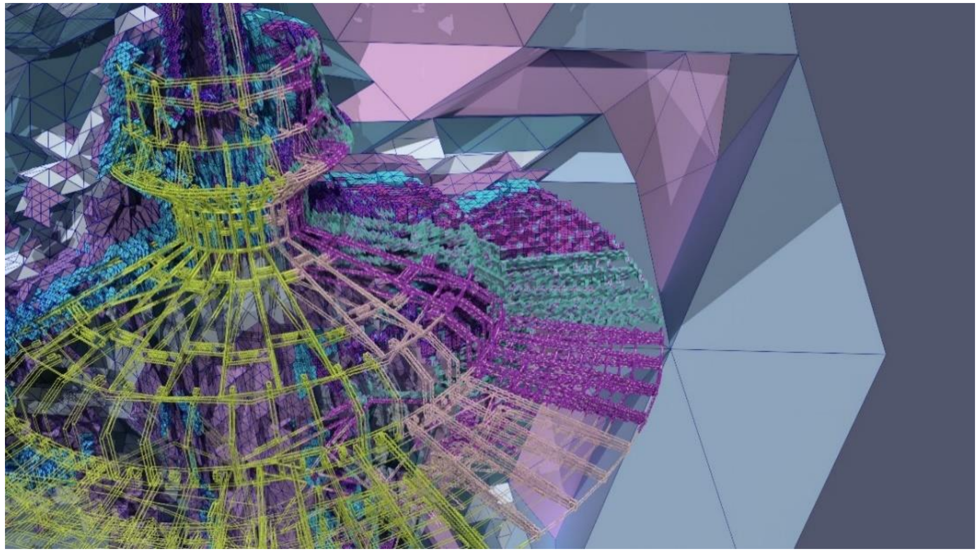




Parallel Frameworks





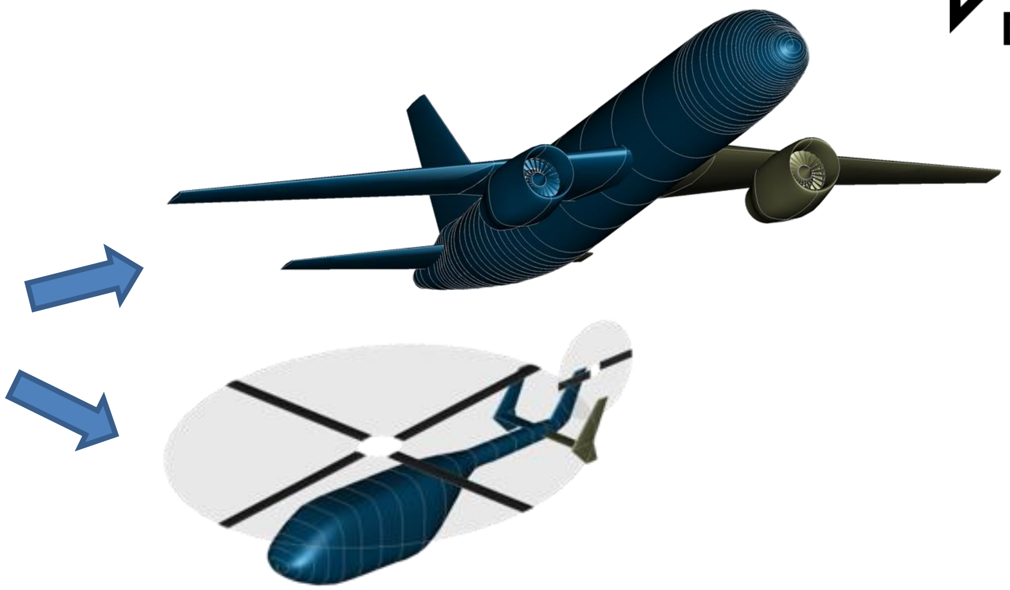


TiGL- Overview

(TiGL Geometry Library)

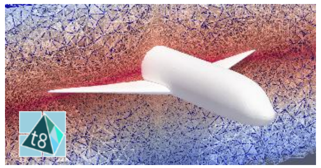
- Parametric aircraft geometry modeler implementing the XML- based CPACS Schema

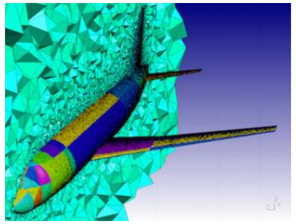
cpacs
A Common Language for Aircraft Design

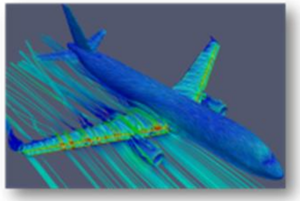


- Established for preliminary design of aircrafts in DLR, universities and industry

- Provides “single source of truth” geometry model for MDAO processes







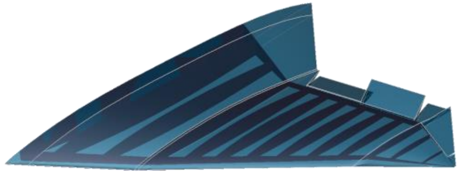
- Open source and joint development

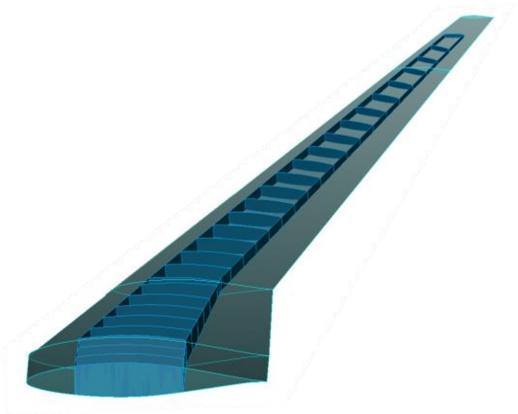
DLR
AIRBUS DEFENCE & SPACE
RISC Software GmbH

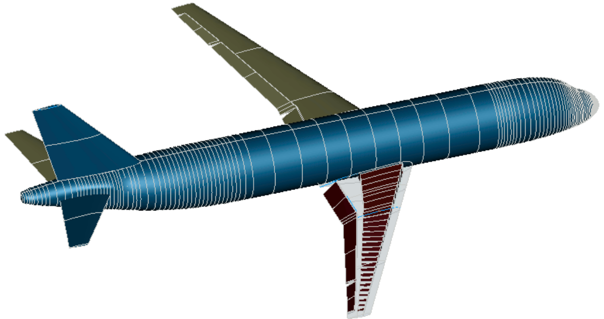
- Based on over 15 years expertise in geometry modeling

DLR

TiGL- Features



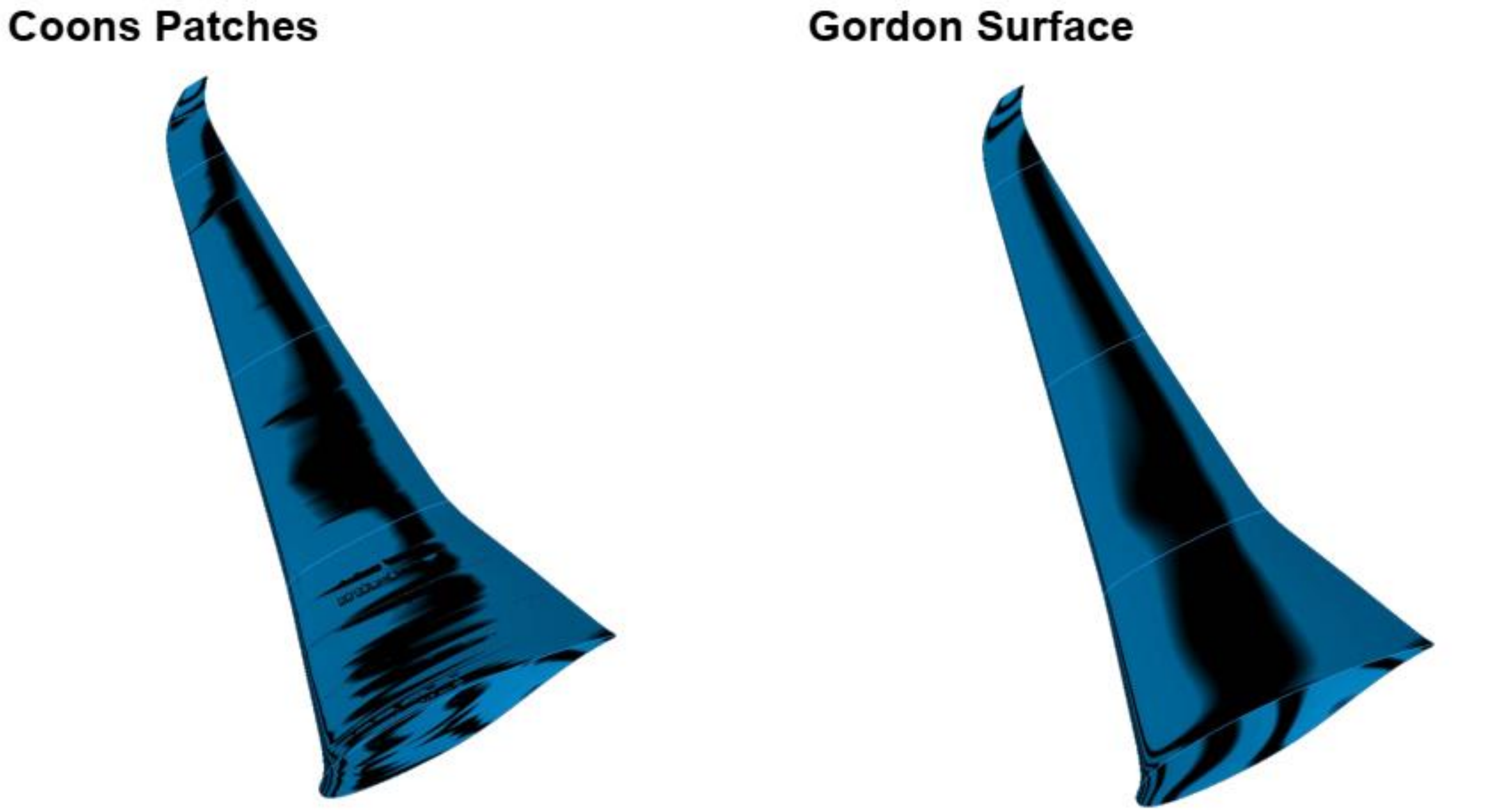




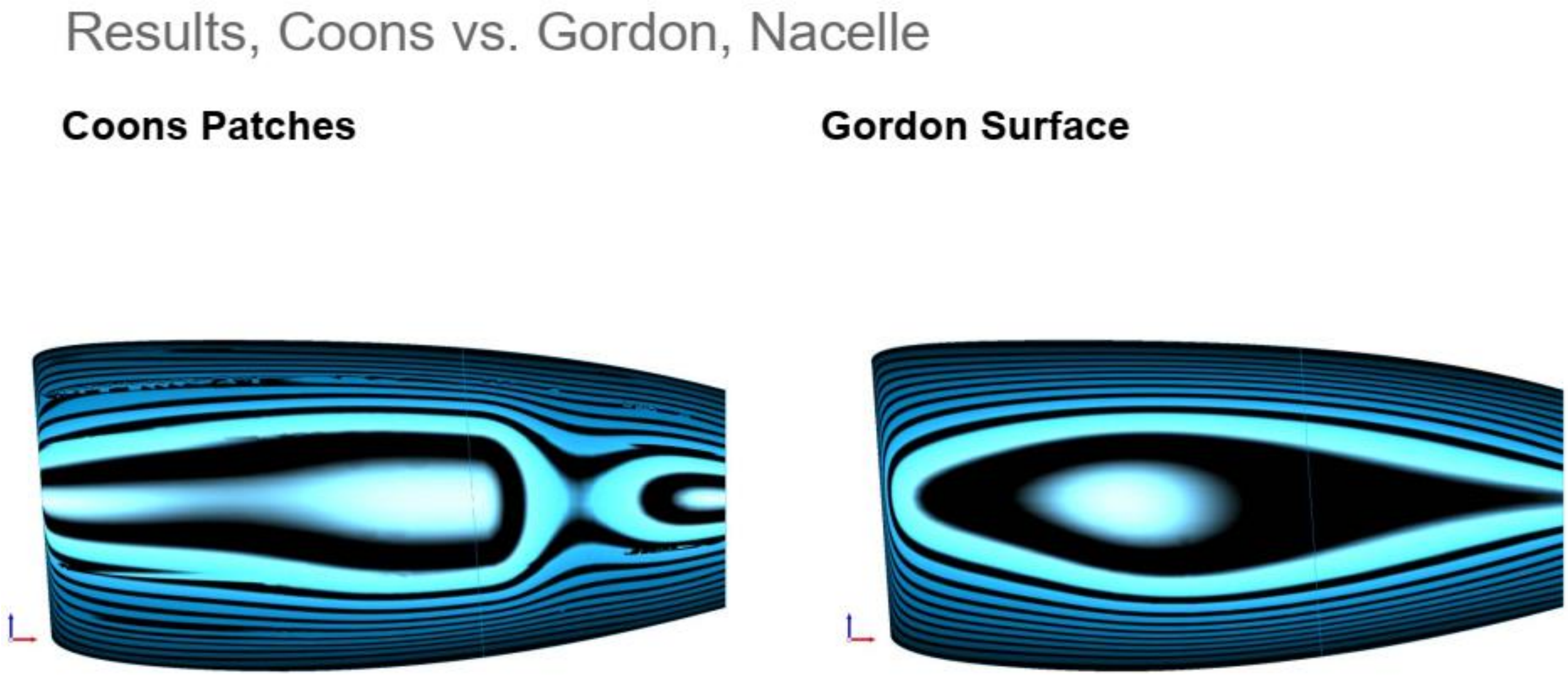
- Advanced NURBS-based modelling algorithms for wings, fuselages, flaps, nacelles, e.t.c.
- Geometry export to common CAD formats
- Language bindings to C, C++, Matlab, Java and Python
- A 3D-Viewer based on OpenGL

Gordon Surfaces → better quality than Coons Patches

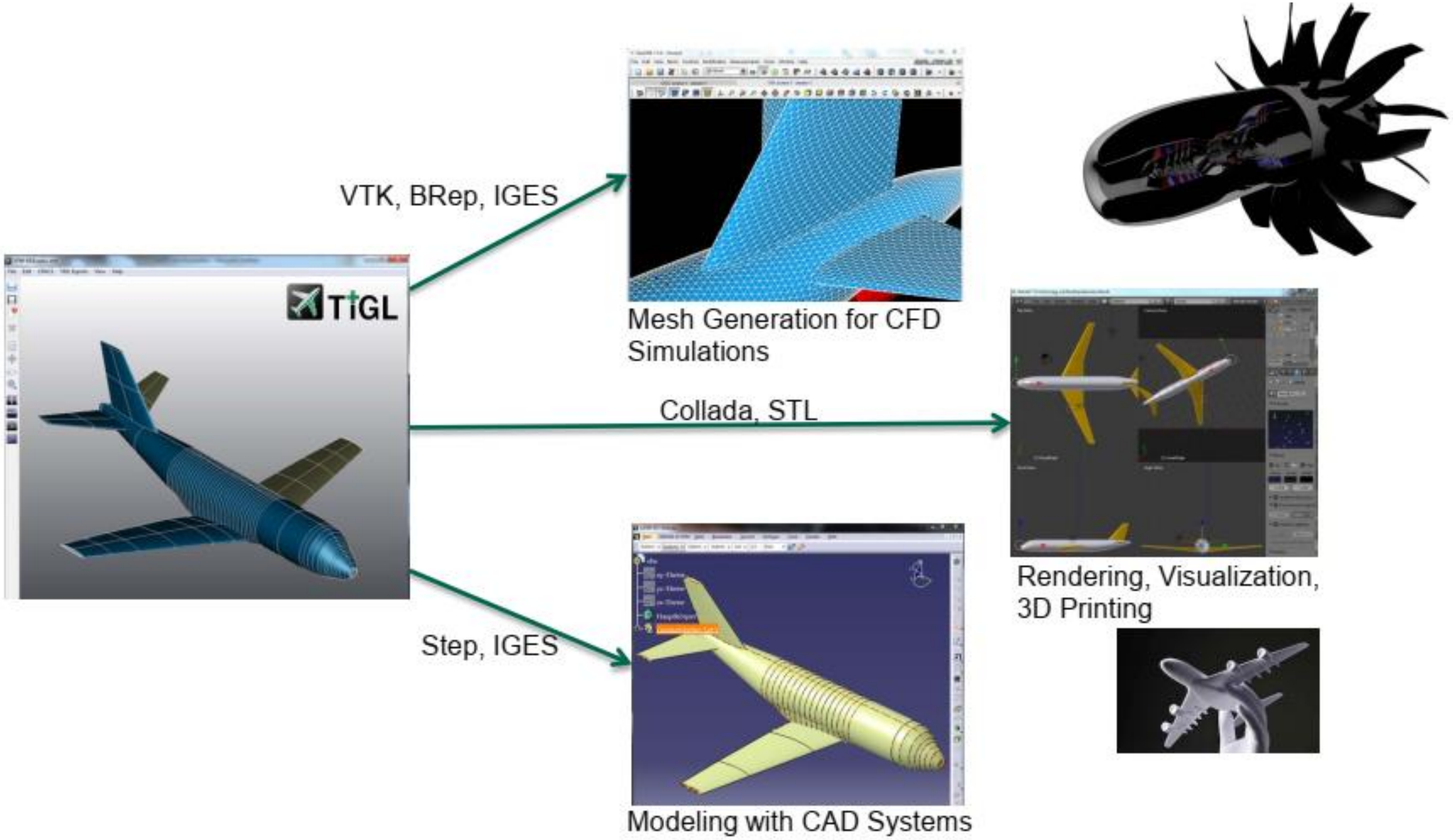
Results, Coons vs. Gordon, Wing



Gordon Surfaces → better quality than Coons Patches

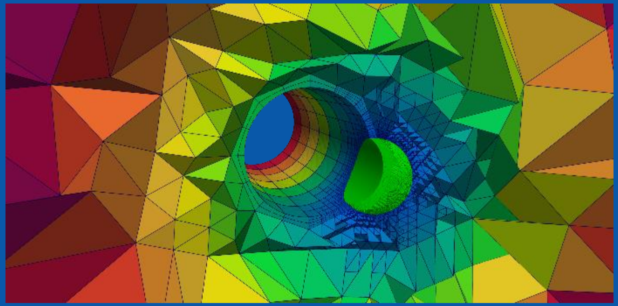


TiGL: How it is used



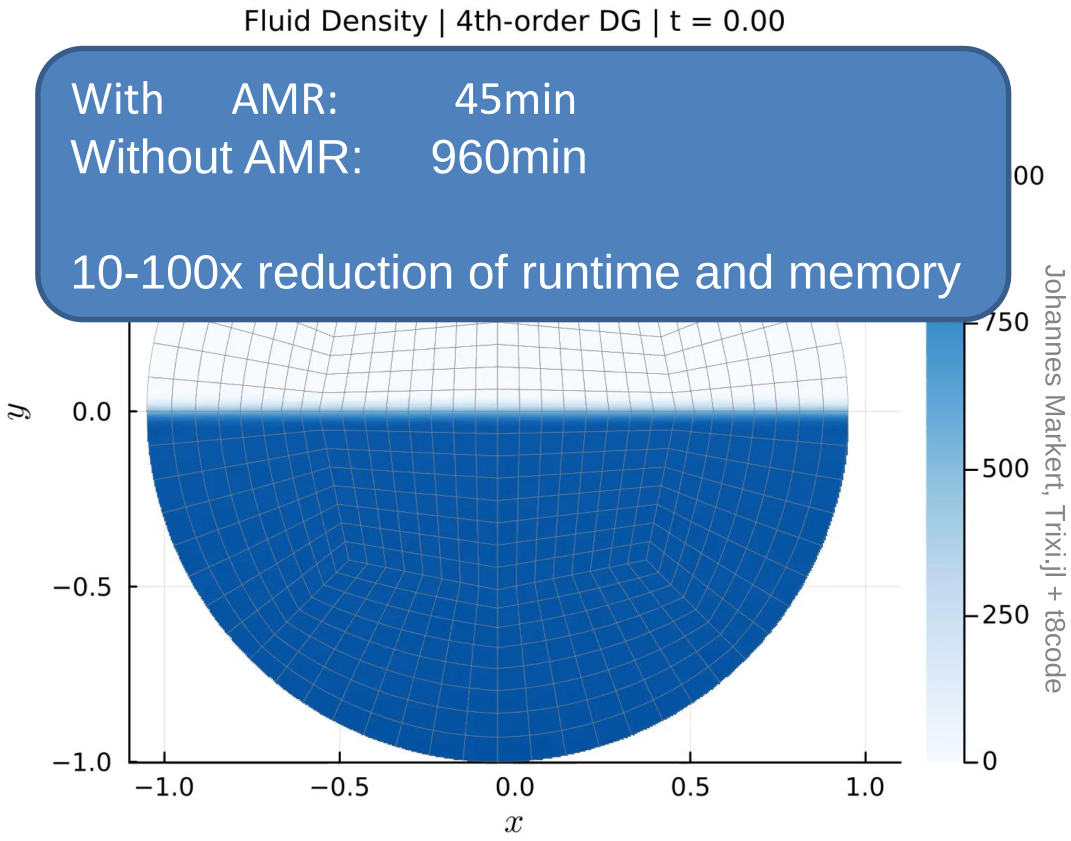
t8code
Dynamic adaptive mesh refinement (AMR)

- Enables solvers to use AMR
- Refine, Coarsen, Load-balance, Ghost, …
- 1D, 2D, 3D
- Tetrahedra, Hexahedra, Prisms, Pyramids, …
- Extremely efficient and low memory footprint
- Scales to 1 Trillion elements
- Scales to 1 Million MPI processes



Better results at lower cost

Hydrogen tank sloshing - HYTAZER



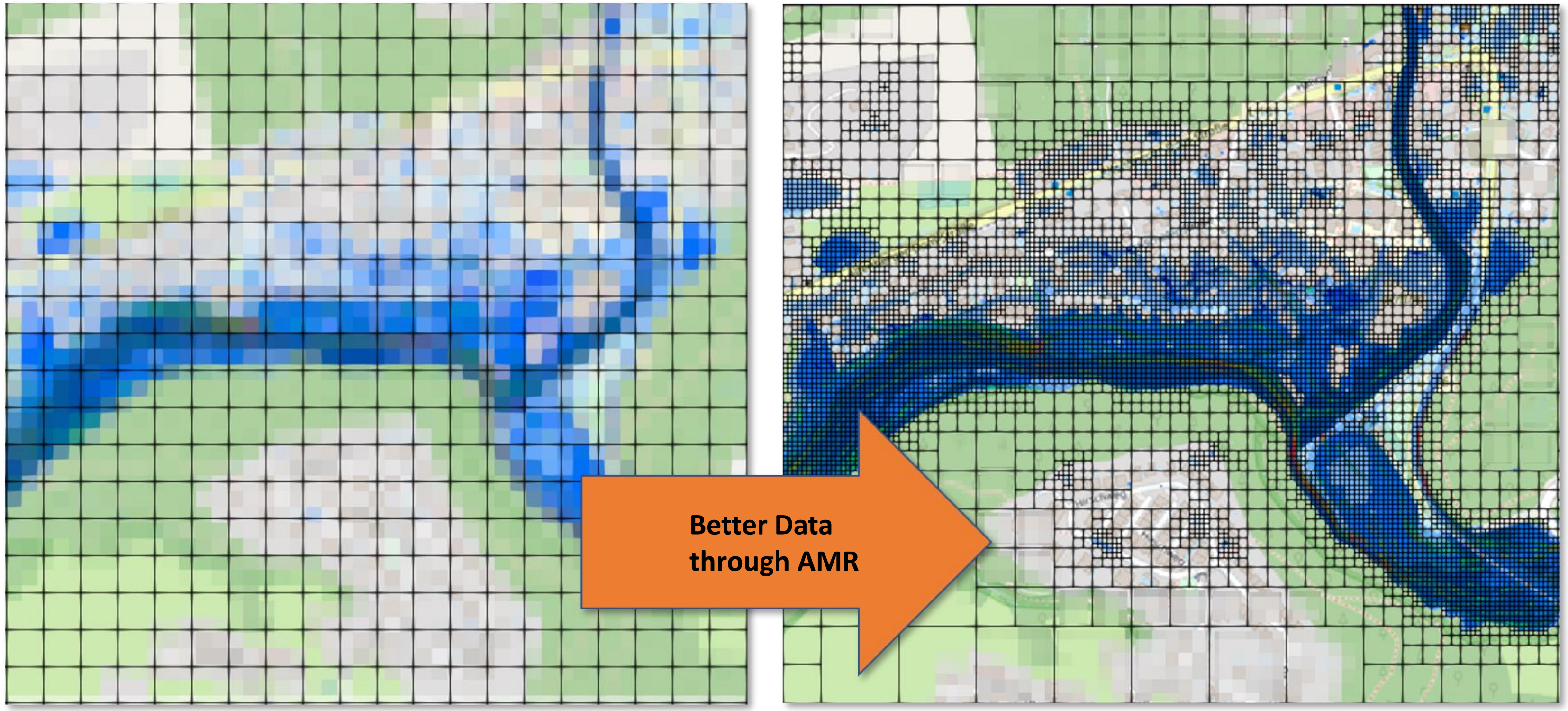
With AMR: 45min
Without AMR: 960min

10-100x reduction of runtime and memory

Pilot Lab Exascale Earth System Modelling

- Data management with space filling curves, e.g. for adaptive grid refinement
- Machine learning in model parametrizations and transfer functions

Flood analysis and prediction → operational forecasting system



Hydrotec GmbH

- Simulations of rainfall and local flooding
- Consulting for cities and communities on flood prevention measures

Coupling with t8code (DLR-SC)

- Parallelisation
- Increase resolution and decrease runtime
- Increase simulation areas (whole regions vs. cities)
- 200k€ support by DLR Technology Marketing
- Will result in commercial licence for t8code

Multibody AEeromechanic COomprehensive simulation– MAECOsim

Comprehensive Aeromechanics Code from DLR

- Simulation of the complete rotorcraft
  → Core tool for helicopter design
- Spans multiple disciplines
  - Structural mechanics & aerodynamics
  - Rotor dynamics & aero-elastics
  - Flight mechanics

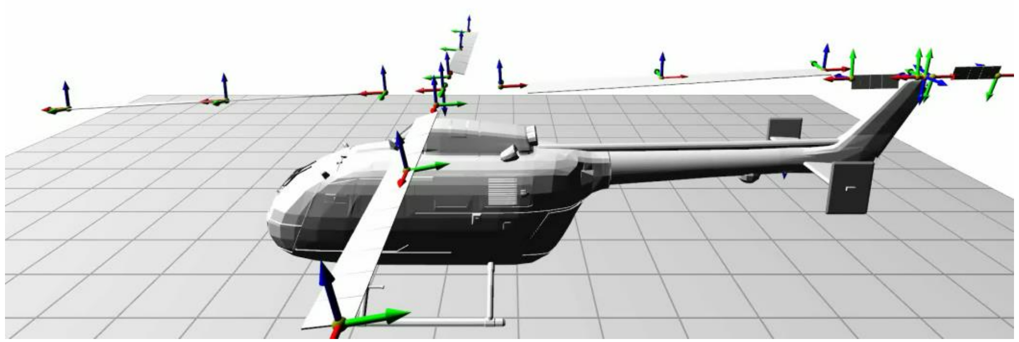
- Restrictions in other software:
  (commercial / proprietary)
  - Severe modeling limitations
  - Difficulty to extend third-party codes
  - Very costly, US-only, license restrictions

Aerodynamic forces (simplified)

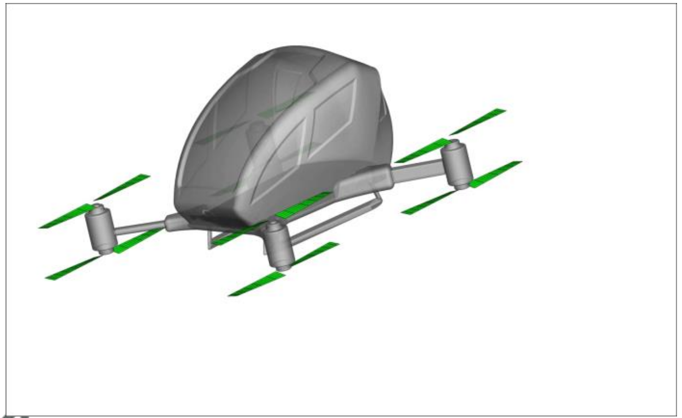
Flexible & complex mechanical parts

Multibody AEeromechanic COomprehensive simulation– MAECOsim

Collaboration and role of DLR-SC



Visualization in the VAST-GUI



Configuration with multiple rotors

- Development with DLR-FT since 2016
  - Currently ~9 core developers
  - Part of various projects (CHASER, ARCADE, ROME, LuFo eVOLve, …)
  - Modular and generic design
  - Medium fidelity models integrated (flexible multibody dynamics, simple aerodynamics)
  - High fidelity → coupling with other DLR- / commercial software (CODA, SIMPACK)

- Contribution of DLR-SC:
  - Software quality (Automatic tests, +++)
  - Tailored numerical methods (flexible rotor blades, trim, solver for coupling disciplines)

Multibody AEeromechanic COomprehensive simulation– MAECOsim

- Aktuelle Entwicklung: Anbindung an AVES
- AVES (Bild rechts; im DLR in Braunschweig):
  Flug-Simulator mit beweglichem Cockpit
- Simulation in Echtzeit:
  - Inputs für uns: Steuerbewegungen des Piloten
  - Outputs von uns: Bewegung des Cockpits
- Ziel bspw. Training und Erprobung des Flugverhaltens von neuen Hubschraubern / speziellen Konfigurationen

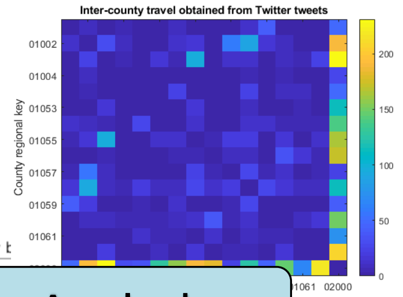
Pandemic Simulation

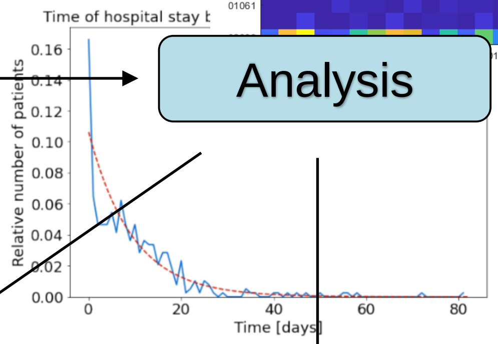
- Start of research 03/2020
- Extension of epidemiological models by hardware-efficient software and high-performance computing
- Integration of geographic and demographic heterogeneity
- Applicable to other infectious diseases
- Modular open-source software
- Cooperation partners:
  HZI, LEOSS (University of Cologne), EO Data Science
  HGF project LOCI: HZI, FZJ, CISPA, UFZ, RKI; a mobile phone provider

ROBERT KOCH INSTITUT …

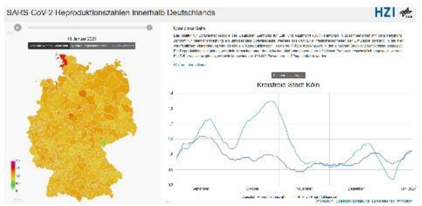
LEOSS

Data → Analysis → Real-time observation

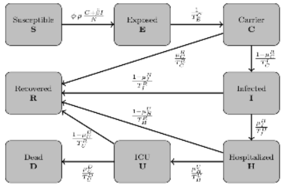


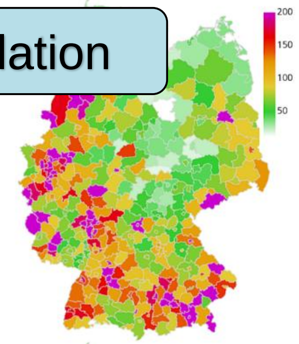


hpcvscorona.dlr.de

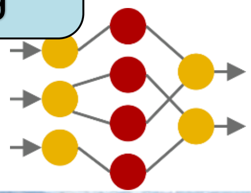


Modelling → Simulation





Machine Learning → Predictive Frontend



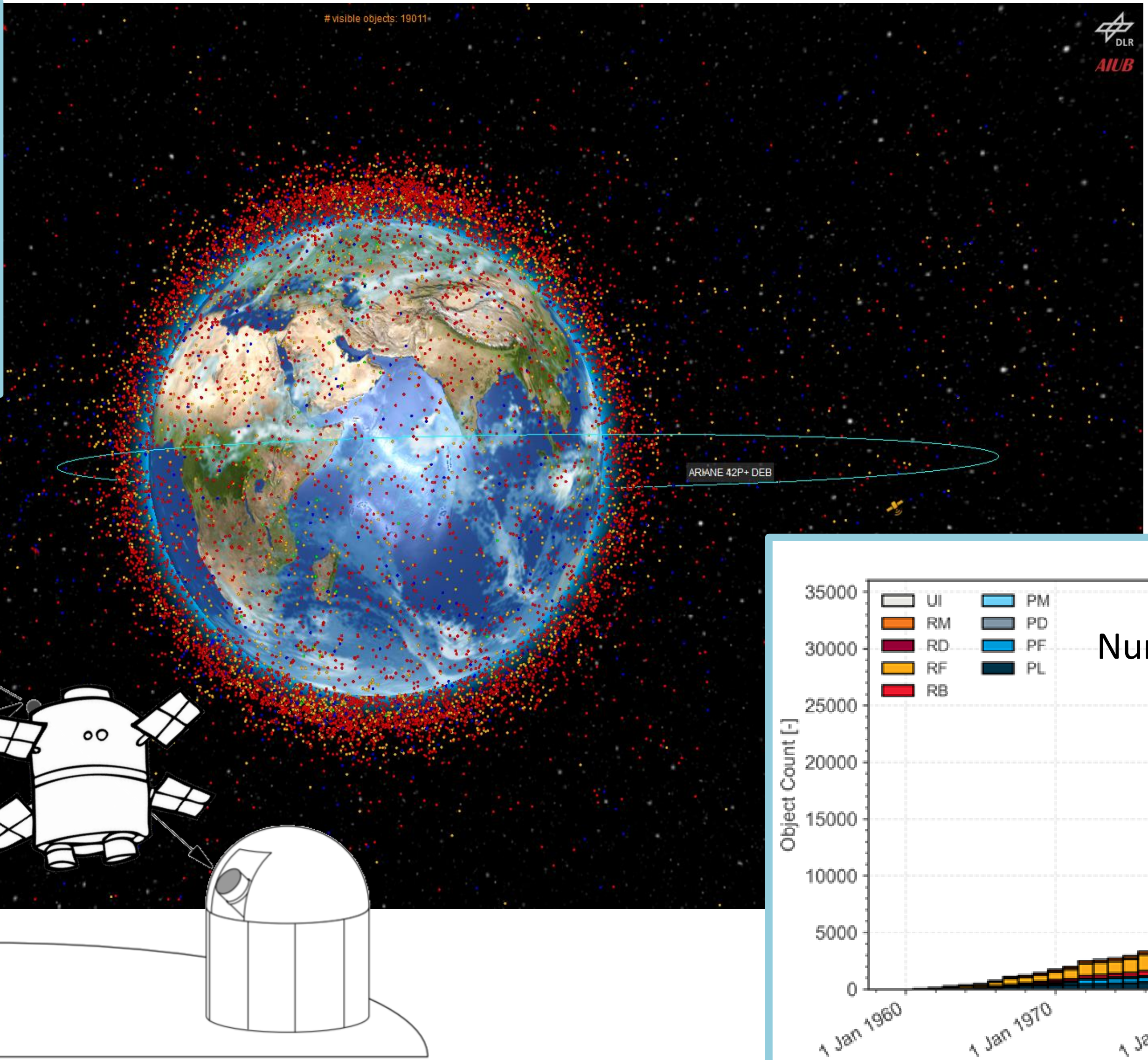
Weltraumschrott

Picture from
http://kidsnews.hu/2018/03/az-urszemetrol/

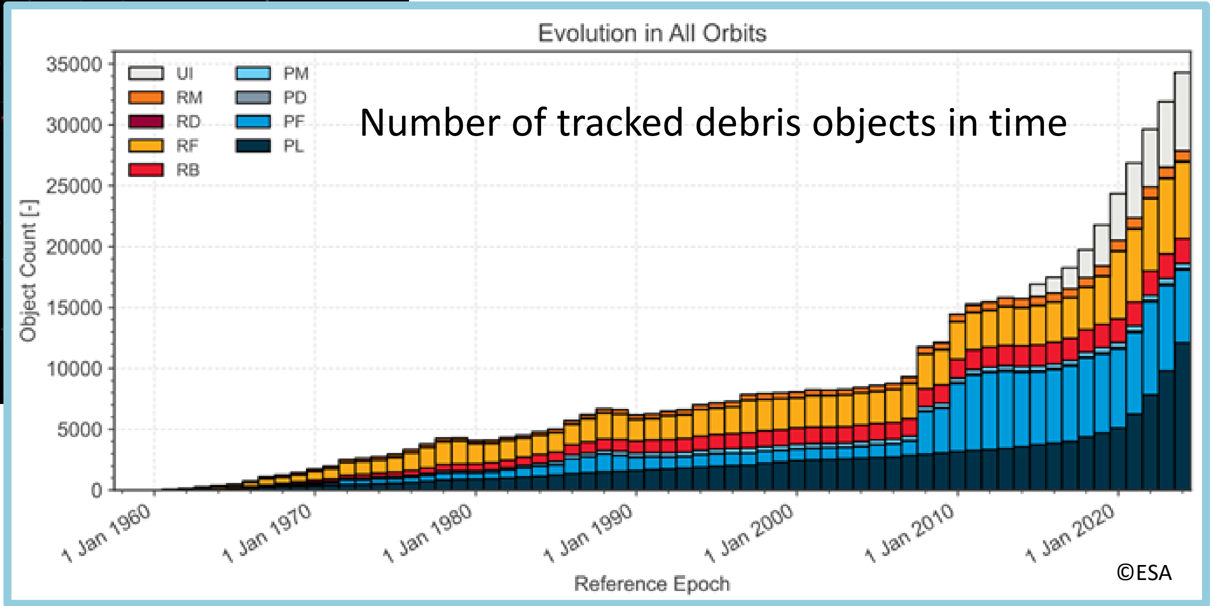
DLR

Knowledge for Tomorrow

BACARDI – Backbone Catalogue of Relational Debris Information



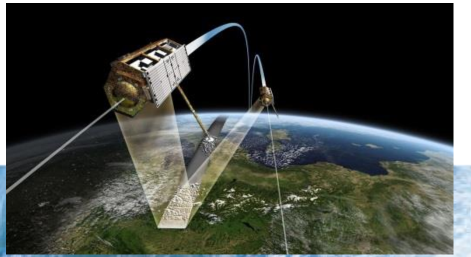
- Catalogue of space objects
- **Detection of new objects**
- Collision avoidance
- Re-entry prediction

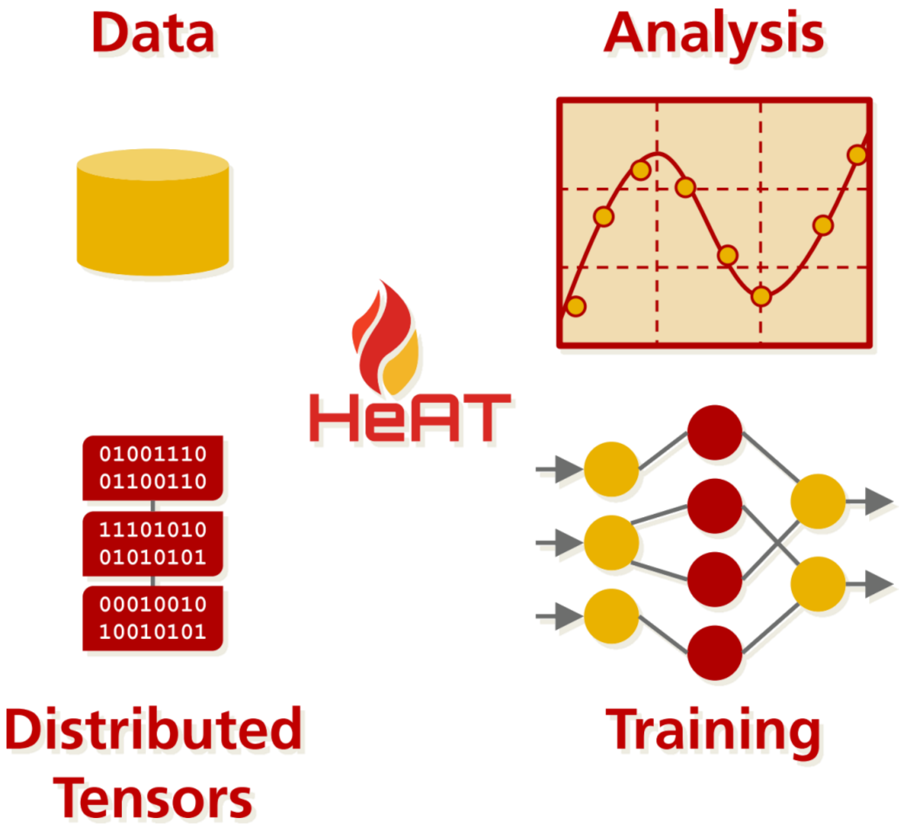


Paralleles Maschinelles Lernen

Big Data & High-Performance Machine Learning

- **Software framework HeAT = Helmholtz Analytics Toolkit**
- Python framework for **parallel**, **distributed** data analytics and machine learning
- Developed within the Helmholtz Analytics Framework Project since 2018
- AIM: Bridge data analytics and **high-performance computing**
- **Space research:**
  - Rocket engine combustion
  - Space debris analysis
  - Satellite data analysis





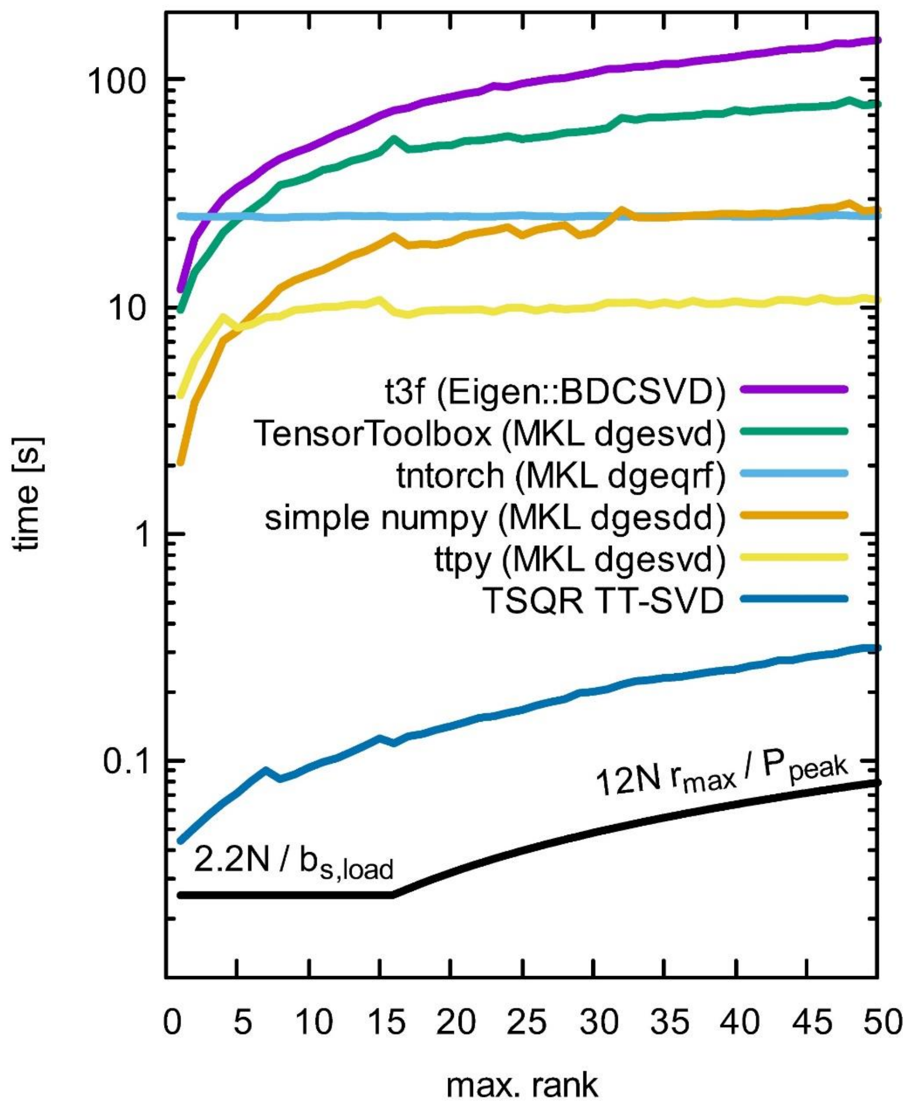
Performance of Tensor Methods (HDS-LEE)

pipeline passed | coverage 96.80%

C++20 Shell Python

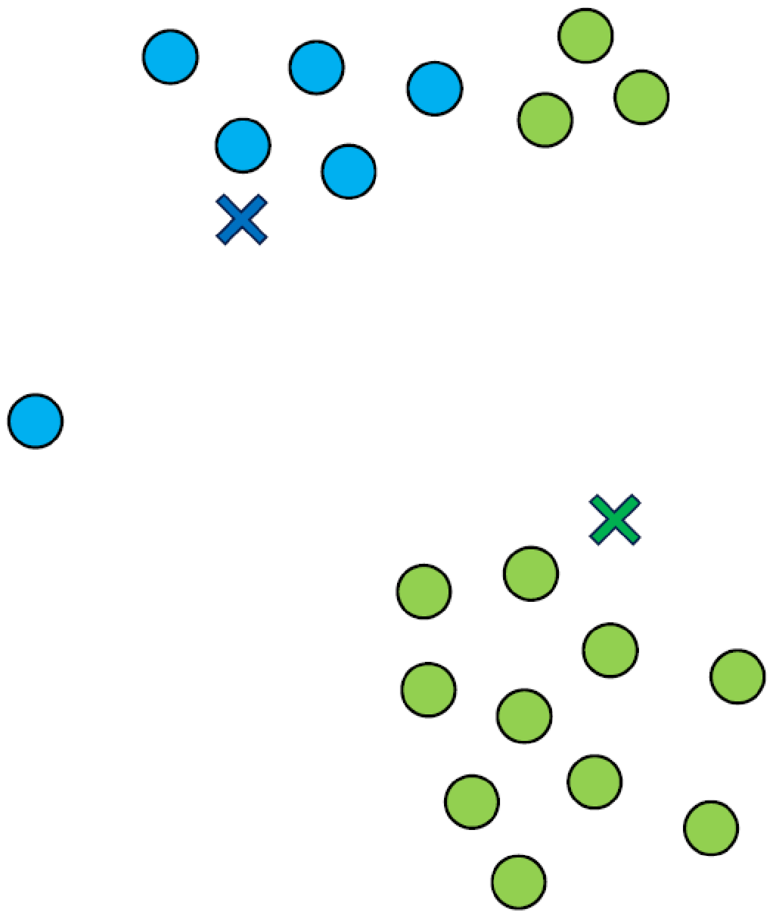
- **Tensor-Train (TT) - Format**:
  - Erweiterung der Singulärwertzerlegung (SVD) auf $\mathbb{R}^d$
  - verlustbehaftete Kompression
- Anwendungen:
  - Data-Science
  - Simulation mit Unsicherheiten
  - Simulation von Quantensystemen
- Grundoperationen und parallele Algorithmen
  - **TT-SVD: große Daten mit TT approximieren**
  - Addition, Rekompression, lineare Gleichungslöser, …
- Performance-Modellierung aller relevanten Operationen
  → detaillierter Vergleich Modell vs. Messung (für jede Funktion)
- Komplexes Zusammenspiel von Genauigkeit und Performance

TT-SVD von 2^{27}-Tensor



https://gitlab.dlr.de/sc/hpc-open/pitts (LOC: 22k + 19k unit tests)

Example: k-means



Numpy vs. HeAT

Compute **new centroid positions** by averaging

```
>>> matching_centroids.shape
(18, 1, 1)

>>> data.shape
(18, 2, 1)
```

NumPy

```
>>> for i in range(self.n_clusters):
>>>     new_centroids[:, :, i:i+1] = ((data*selection).sum(axis=0, keepdims=True) /
                                       selction.sum(axis=0).clip(1.0, sys.maxsize))
```

HeAT

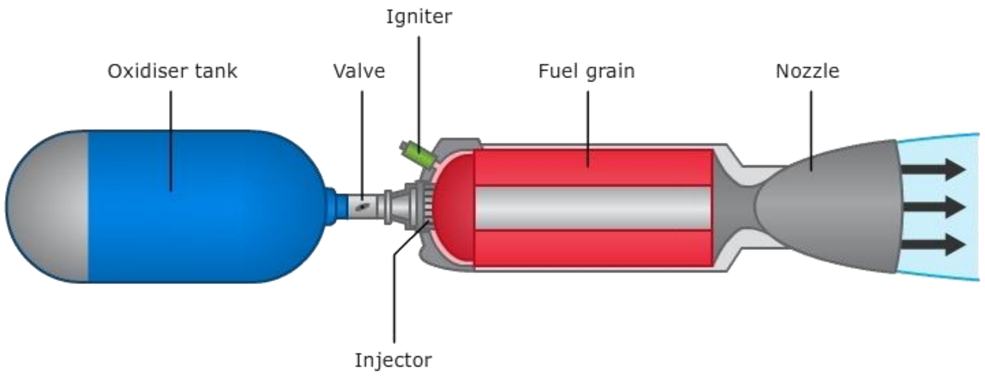
```
>>> for i in range(self.n_clusters):
>>>     new_centroids[:, :, i:i+1] = ((data*selection).sum(axis=0) /
                                       selction.sum(axis=0).clip(1.0, sys.maxsize))
```

```
>>> new_centroids.shape
(1, 2, 2)
```

HeAT hides parallelism, looks like sequential NumPy code.

A real world example:
Rocket engine combustion analysis

- **Goal**: Cost reduction of rocket engines, be competitive with e.g. Space-X



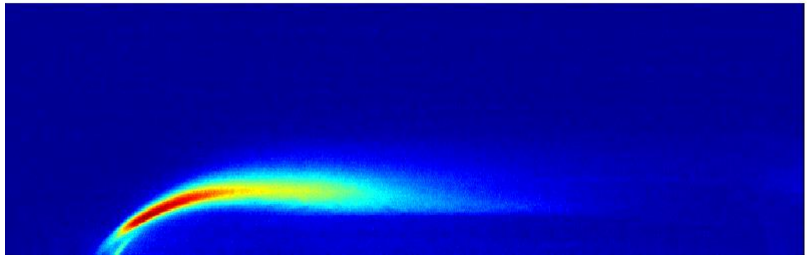
Hybrid rocket engine

- Pressurized fluid oxidizer
- Solid fuel
- A valve controls, how much oxidizer gets into the combustion chamber

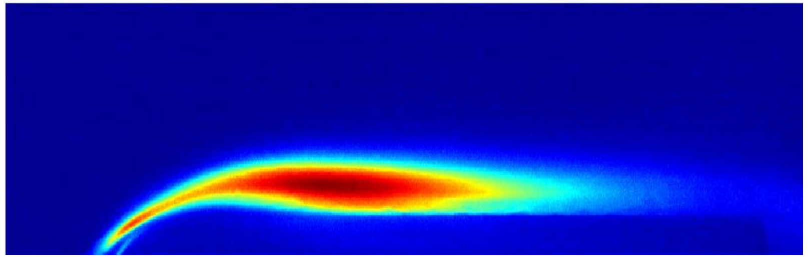
- Advantages
  - Cheap
  - Controllable

A real world example:
Resulting Clusters, k = 7

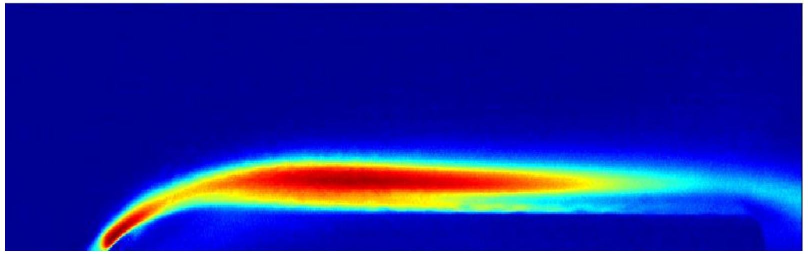
Centroid 0



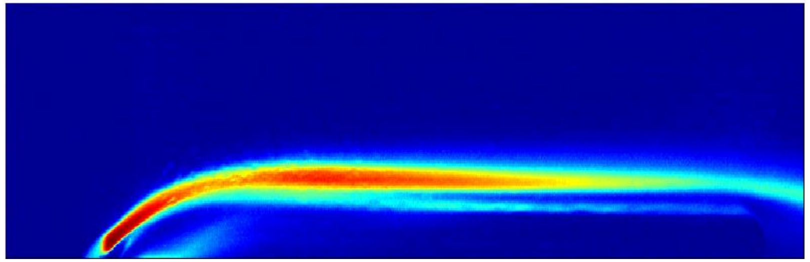
Centroid 1



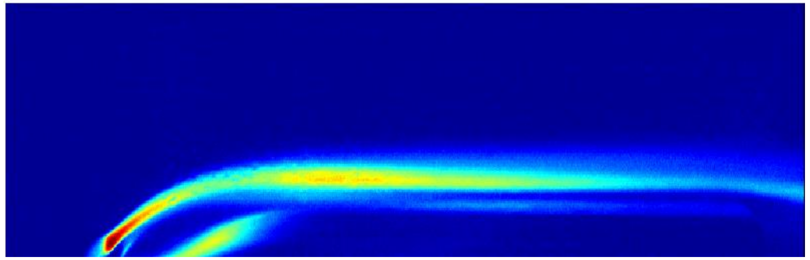
Centroid 2



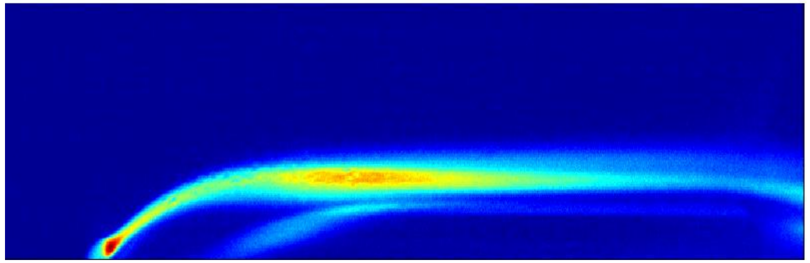
Centroid 3



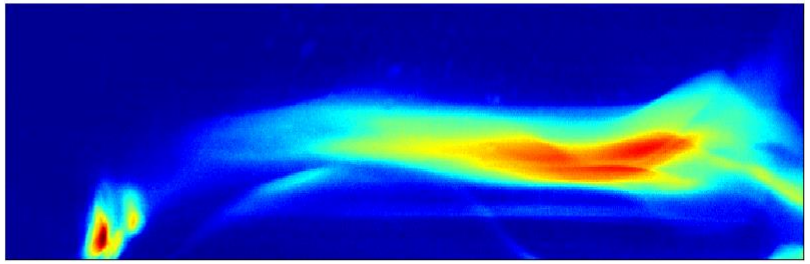
Centroid 4



Centroid 5

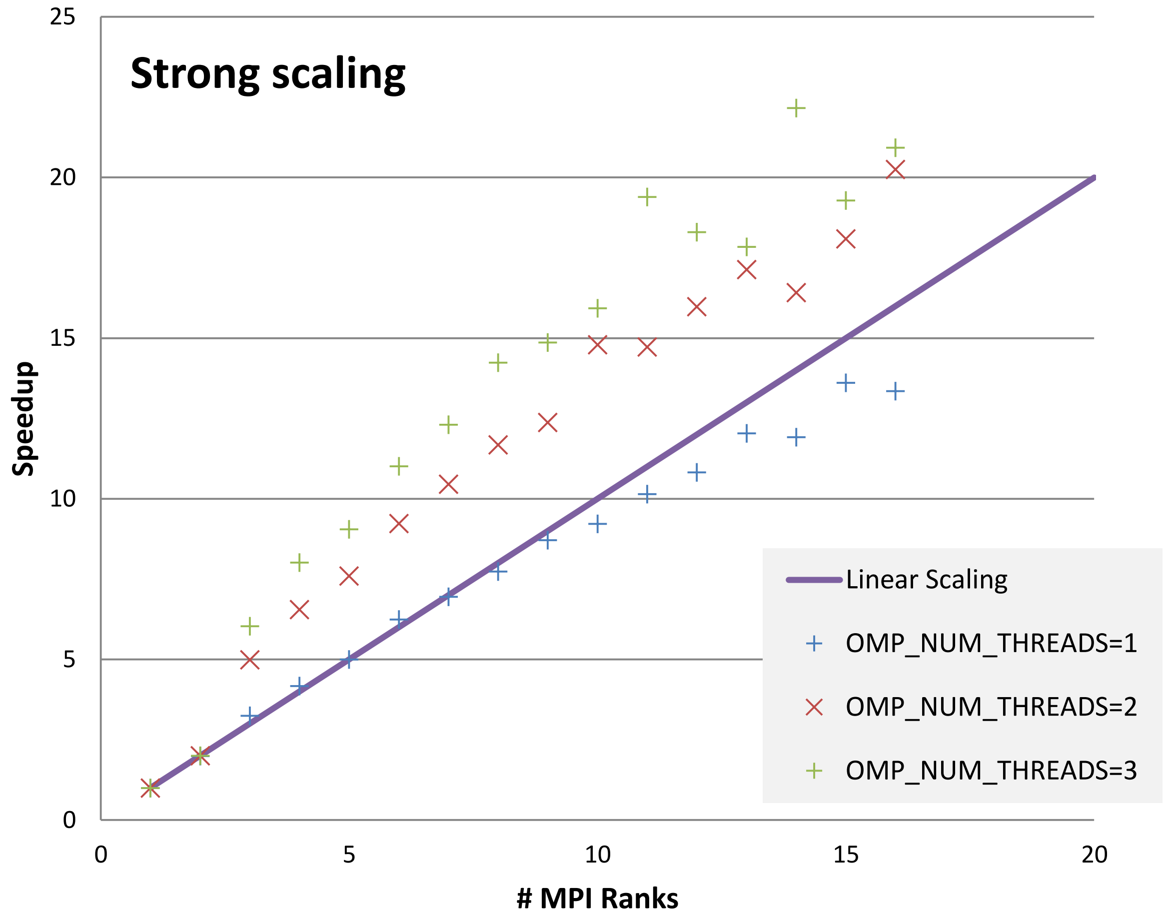


Centroid 6



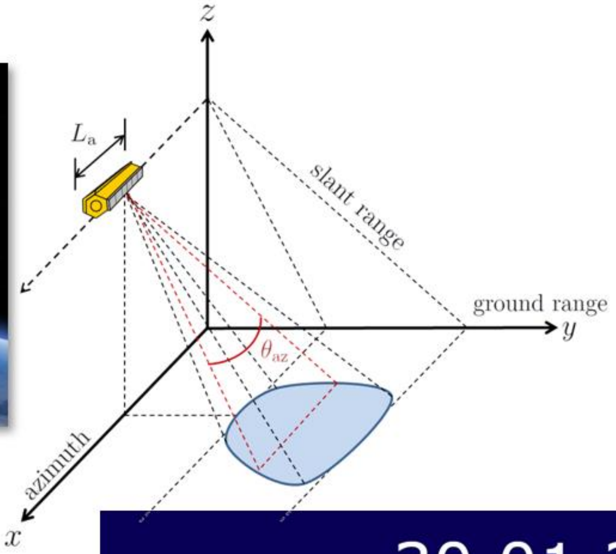
A real world example:
Computational Performance

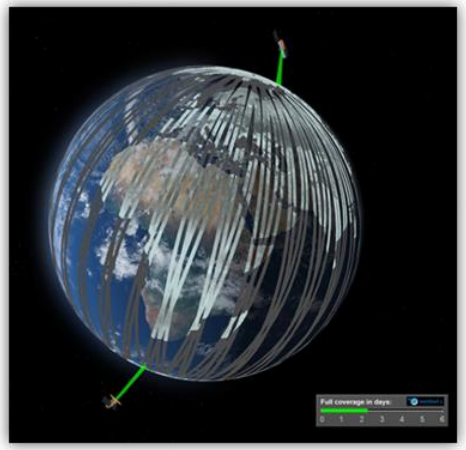
- Hybrid shared memory + distributed memory setting
- CPU only
- Variation of 1 … 16 MPI total ranks
- Variation of 1 … 3 local threads per process
- Strong scaling analysis: How does the computing time reduce with number of ranks?
- Results look promising, testing on larger systems + distributed GPUs also successful



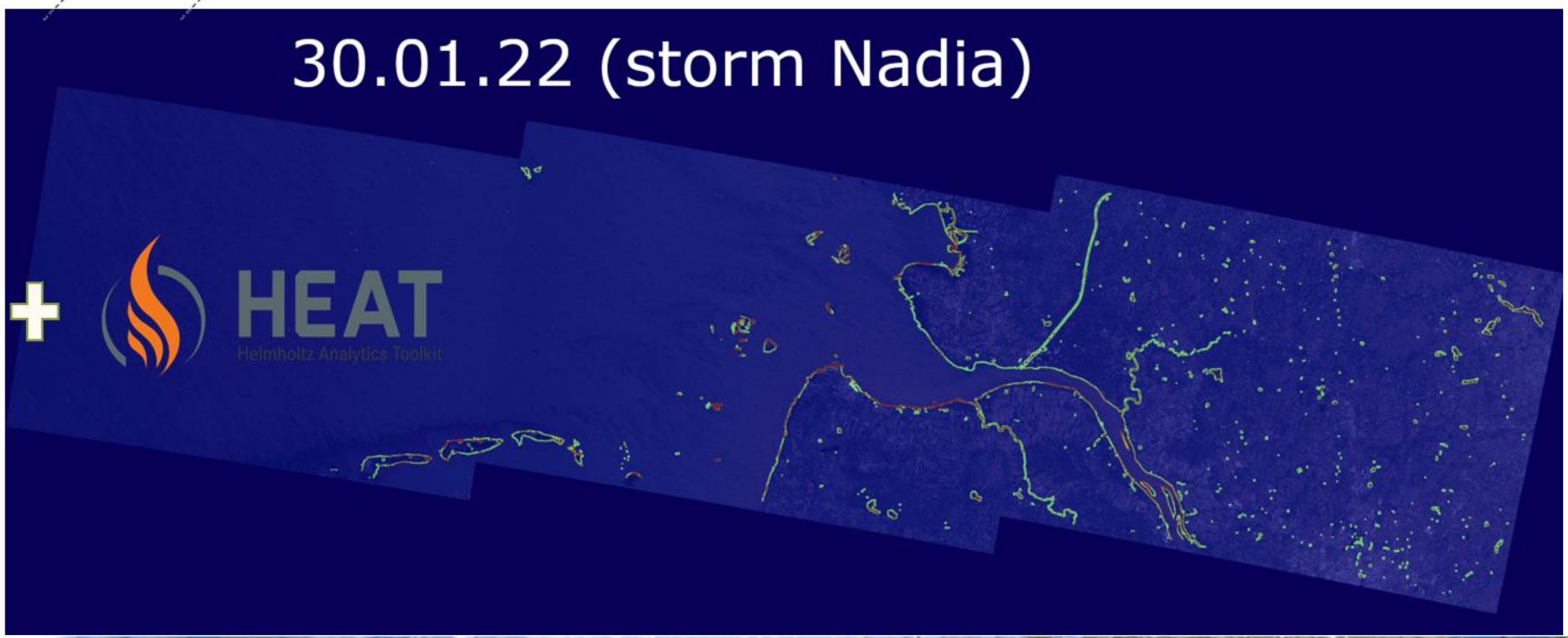
RESIKOAST - Large-scale anomaly detection on North Sea coast

Sentinel-1

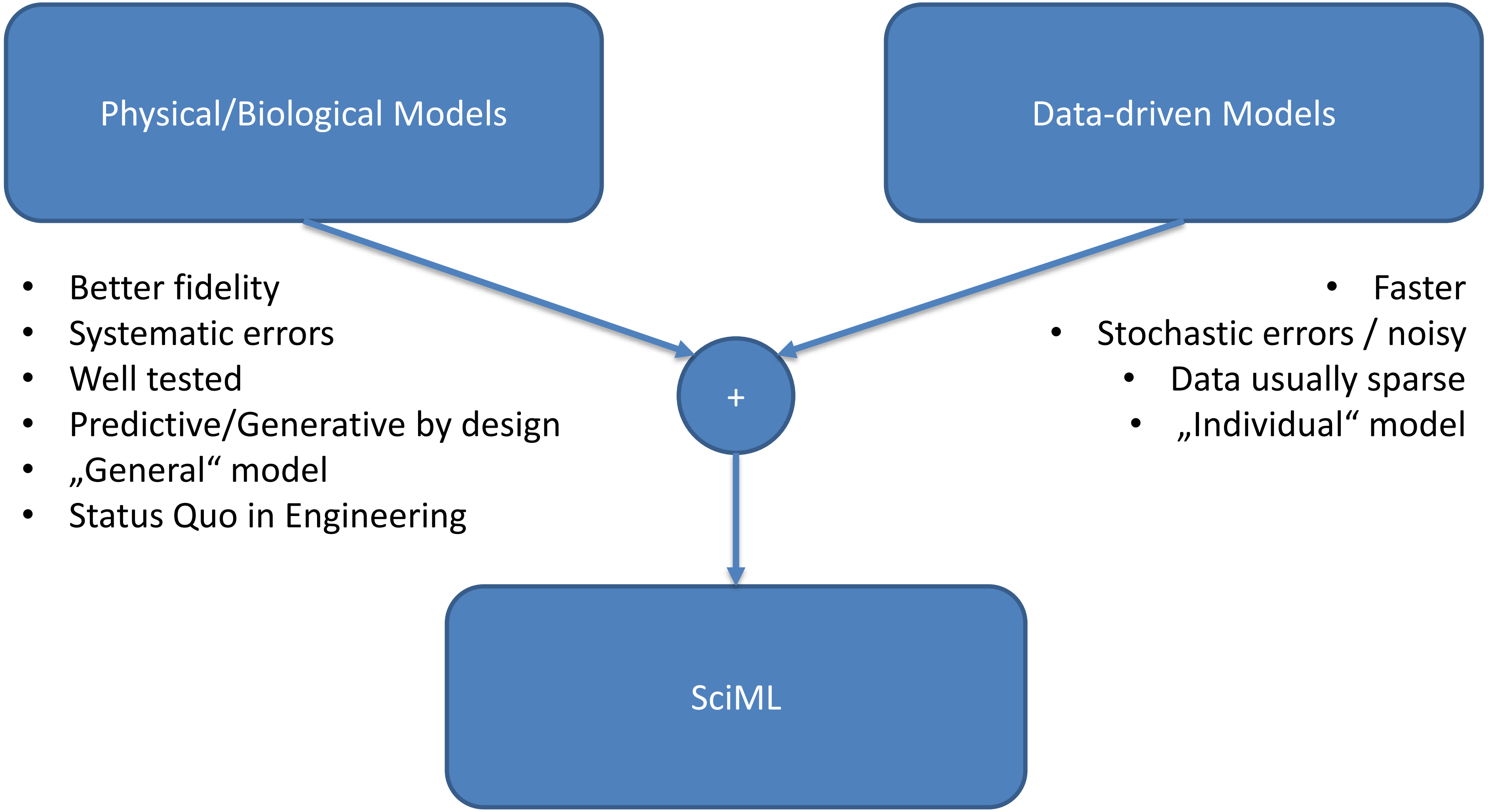




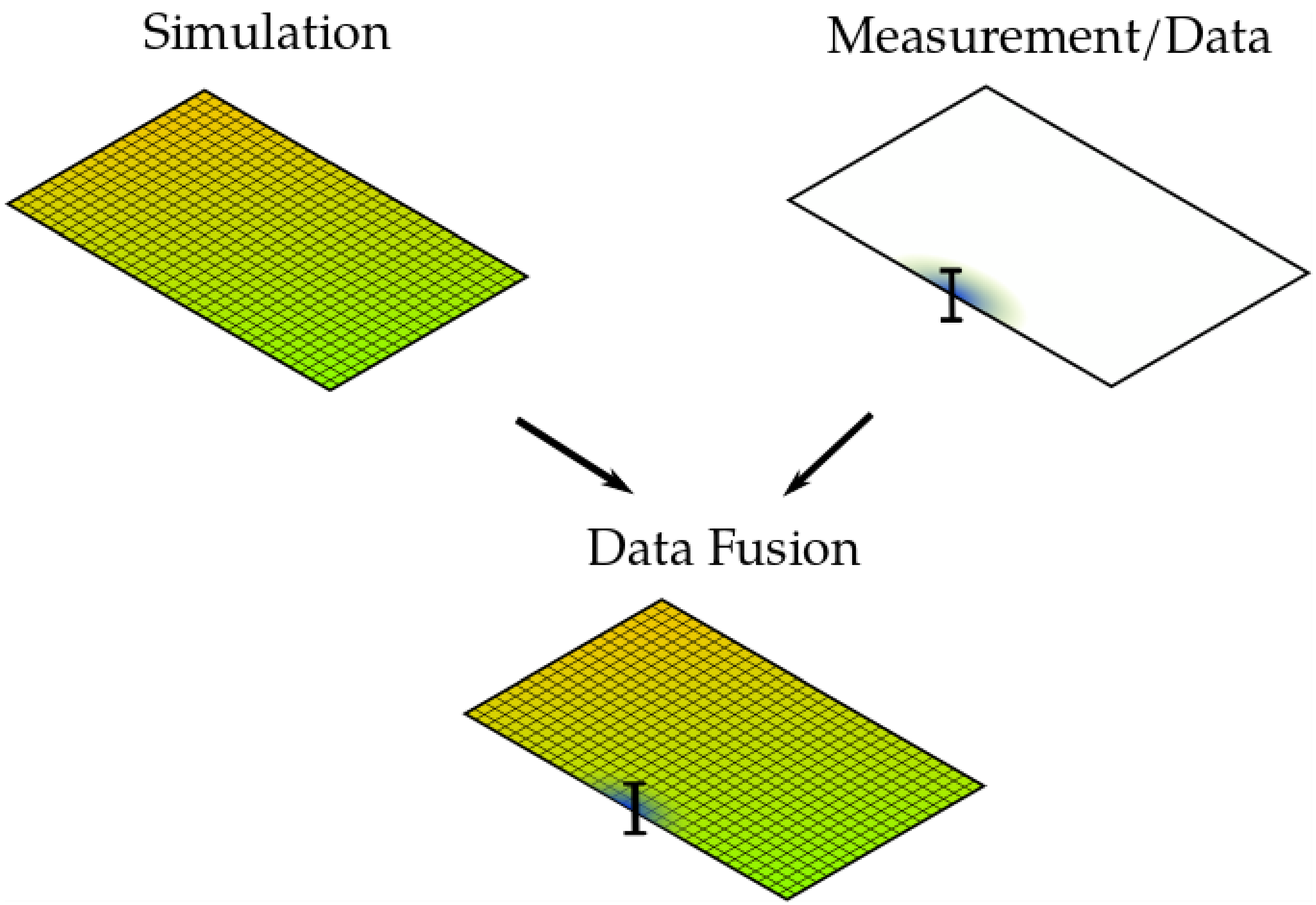
- SAR- Data
- Water-Mask
- ML-Methods:
  - Local-Outlier-Factor
  - Autoencoder
- HPC



SciML Modellbildung



Uncertainty Quantification



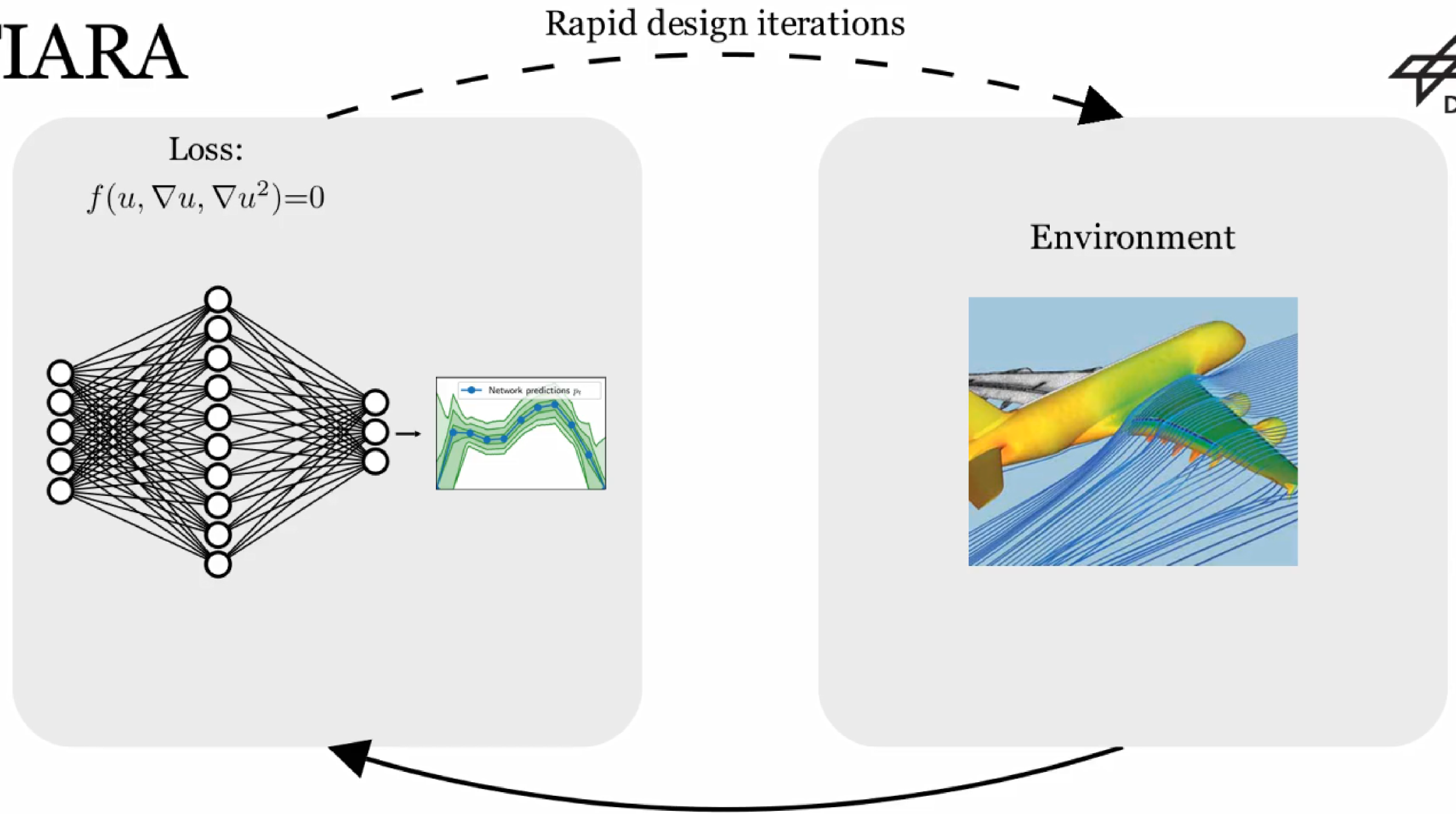
TIARA - Trustworthy physics Informed Ai foR Aerospace and transportation

- Modeling of technical systems is often complex and expensive.

→ Need for **trustworthy** surrogate models.

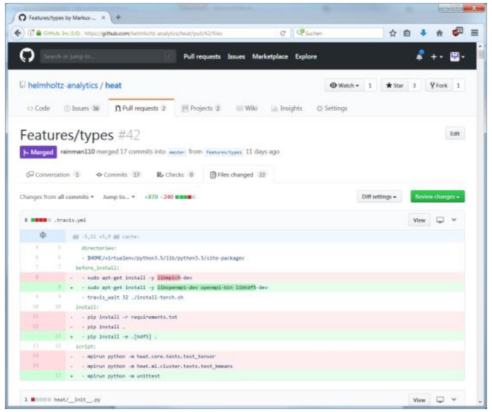
DALL-E / ChatGPT

TIARA

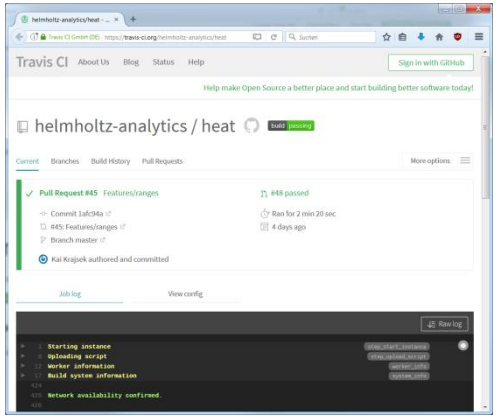

Rapid design iterations
Loss:
$f(u, \nabla u, \nabla u^2)=0$
Network predictions
Environment

Transparent development process

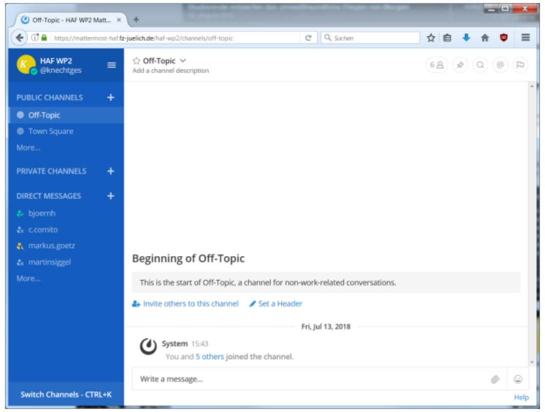
Github for code review, issue tracking, sprint planning



Travis for continuous integration



Mattermost for discussions



https://github.com/helmholtz-analytics

Scan me

Join us there!

[Mission ATEK: Vom hohen Norden ins All]

Many thanks for your attention!

Questions?

Dr.-Ing. Achim Basermann

German Aerospace Center (DLR)

Institute of Software Technology

Department Head High-Performance Computing

Achim.Basermann@dlr.de

http://www.DLR.de/sc