HPC Software Infrastructures at German Aerospace Center

Dr.-Ing. Achim Basermann
German Aerospace Center (DLR)
Simulation and Software Technology
Department Head “High-Performance Computing”
Survey

• DLR and SC-HPC

• An Exascale Software Infrastructure for Solving Quantum Physics Problems

• Parallel Coupling Frameworks
  • for Airplan Simulation
  • for Helicopter Simulation

• The Software Framework HeAT for Data Analytics with Parallel Machine Learning Methods

• Software for Quantum Computing
DLR

German Aerospace Center

• Research Institution
  • Research and development in aeronautics, space, energy, transportation, digitalization and security
  • National und international cooperations

• Space Agency
  • Planing and implementation of German space activities

• Project Management Agency
  • Research promotion
DLR Locations and Employees

Approx. 8000 employees across 40 institutes and facilities at 20 sites.

Simulation and Software Technology

• stands for innovative software engineering,

• develops challenging individual software solutions for DLR, and

• is partner in scientific projects in the area of simulation and software technology.
DLR Institute Simulation and Software Technology
Scientific Themes and Working Groups

Departments

- Intelligent and Distributed Systems
- High-Performance Computing
- Software for Space Systems and Interactive Visualization

Working Groups

- Distributed Software Systems
- Software Engineering
- Intelligent Systems
- Intelligent Algorithms and Optimization
- Parallel Numerics
- Onboard Software Systems
- Modeling and Simulation
- Scientific Visualization
- Virtual and Extended Reality
High Performance Computing Teams

**Department**
High Performance Computing
Head: Dr. Achim Basermann
Deputy: Dr. Margrit Klitz

- **Intelligent Algorithms & Optimization**
  Dr. Martin Siggel

- **Quantum Computing**

- **Parallel Numerics**
  Dr. Jonas Thies
An Exascale Software Infrastructure for Solving Quantum Physics Problems
ESSEX project – background

Quantum physics/information applications

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

and beyond....

\[ H \mathbf{x} = \mathbf{\lambda} \mathbf{x} \]

“Few” (1,...,100s) of eigenpairs

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

“Bulk” (100s,...,1000s) eigenpairs

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

→ Sparse eigenvalue solvers of broad applicability
Application, Algorithm and Performance: Kernel Polynomial Method (KPM) – A Holistic View

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

$$X(\omega) = \frac{1}{N} \text{tr}[\delta(\omega - H)X] = \frac{1}{N} \sum_{n=1}^{N} \delta(\omega - E_n) \langle \psi_n, X \psi_n \rangle$$
The Kernel Polynomial Method (KPM)

Optimal performance exploit knowledge from all software layers!

Basic algorithm – Compute Cheyshev polynomials/moments:

\[
\text{for } r = 0 \text{ to } R - 1 \text{ do } \\
|v\rangle \leftarrow |\text{rand()}\rangle \\
\text{Initialization steps and computation of } \eta_0, \eta_1 \\
\text{for } m = 1 \text{ to } M/2 \text{ do } \\
\text{swap}(|w\rangle, |v\rangle) \\
|u\rangle \leftarrow H|v\rangle \\
|u\rangle \leftarrow |u\rangle - b|v\rangle \\
|w\rangle \leftarrow -|w\rangle \\
|w\rangle \leftarrow |w\rangle + 2a|u\rangle \\
\eta_{2m} \leftarrow \langle v|v\rangle \\
\eta_{2m+1} \leftarrow \langle w|v\rangle \\
\text{end for} \\
\text{end for}
\]

Application:
Loop over random initial states

Building blocks:
(Sparse) linear algebra library

Algorithm:
Loop over moments

\[\uparrow \text{ spmv()} \quad \text{Sparse matrix vector multiply} \]
\[\uparrow \text{ axpy()} \quad \text{Scaled vector addition} \]
\[\uparrow \text{ scal()} \quad \text{Vector scale} \]
\[\uparrow \text{ axpy()} \quad \text{Scaled vector addition} \]
\[\uparrow \text{ nrm2()} \quad \text{Vector norm} \]
\[\uparrow \text{ dot()} \quad \text{Dot Product} \]
The Kernel Polynomial Method (KPM)

Optimal performance exploit knowledge from all software layers!

Basic algorithm – Compute Chebyshev polynomials/moments:

for $r = 0$ to $R-1$ do
    $|v\rangle \leftarrow |\text{rand}(\rangle$

    Initialization steps and computation of $\eta_0, \eta_1$

    for $m = 1$ to $M/2$ do
        swap$(|w\rangle, |v\rangle)$
        $|u\rangle \leftarrow H |v\rangle$
        $|u\rangle \leftarrow |u\rangle - |b\rangle |v\rangle$
        $|w\rangle \leftarrow -|w\rangle$
        $|w\rangle \leftarrow |w\rangle + 2a |u\rangle$
        $\eta_{2m} \leftarrow \langle v|v\rangle$
        $\eta_{2m+1} \leftarrow \langle w|v\rangle$
        spmv($)$
        axpy($)$
        scal($)$
        axpy($)$
        nrm2($)$
        dot($)$
    end for
end for

for $r = 0$ to $R-1$ do
    $|v\rangle \leftarrow |\text{rand}(\rangle$

    Initialization steps and computation of $\eta_0, \eta_1$

    for $m = 1$ to $M/2$ do
        swap$(|w\rangle, |v\rangle)$
        $|w\rangle = 2a (H - b 1 |v\rangle - |w\rangle)$
        $\eta_{2m} \leftarrow \langle v|v\rangle$
        $\eta_{2m+1} \leftarrow \langle w|v\rangle$
        aug_spvm($)$
    end for
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⟩ ← |rand(0)
    Initialization steps and computation of η₀, η₁
    for m = 1 to M/2 do
        swap(|w⟩, |v⟩)
        |w⟩ = 2a(H - b1)|v⟩ - |w⟩ &
        η₂m = ⟨v|v⟩ &
        η₂m+1 = ⟨w|v⟩
    end for

end for
```

```
|V⟩ := |v⟩₀..R−1
|W⟩ := |w⟩₀..R−1
|V⟩ ← |rand(0)
Initialization steps and computation of μ₀, μ₁
for m = 1 to M/2 do
    swap(|W⟩, |V⟩)
    |W⟩ = 2a(H - b1)|V⟩ - |W⟩ &
    η₂m[⟨] = ⟨V|V⟩ &
    η₂m+1[⟨] = ⟨W|V⟩
end for
```

▷ aug_spmmv()  
▷ Assemble vector blocks

Augmented Sparse Matrix  
Multiple Vector Multiply
KPM: Heterogenous Node Performance

- Topological Insulator Application
- Double complex computations
- Data parallel static workload distribution
**KPM: Large Scale Heterogenous Node Performance**

![Performance diagram showing the scalability of the Kernel Polynomial Method on large-scale CPU-GPU systems.](image)

- **CRA Y XC30 – PizDaint**
  - 5272 nodes
  - Peak: 7.8 PF/s
  - LINPACK: 6.3 PF/s
  - Largest system in Europe

---


*Thanks to CSCS/T. Schulthess for granting access and compute time*
ESSEX: Software Packages

User Applications

- PHIST
  Pipelined Hybrid Parallel Iterative Solver Toolkit

- GHOST
  General, Hybrid, and Optimized Sparse Toolkit

- 3rd-party libraries: Trilinos, ...

Hardware: CPU / GPGPU / Xeon Phi

User Applications

- MPI+X $X \in \{\text{CUDA, OpenMP, pthreads}\}$

Links to open source repositories at https://blogs.fau.de/essex/code
GHOST Library

- **Hybrid MPI+X** execution mode  
  (X=OpenMP, CUDA)

- **Algorithm specific kernels**: SIMD Intrinsics (KNL) and CUDA (NVIDIA)  
  $\rightarrow$ 2x – 5x speed-up vs. Optimized general building block libraries

- **Tall & skinny matrix-matrix kernels** (block orthogonalization)  
  $\rightarrow$ 2x – 10x speed-up vs. Optimized general building block libraries

- **SELL-C-σ** sparse matrix format

- Open Source code & example applications: [https://bitbucket.org/essex/ghost](https://bitbucket.org/essex/ghost)
A Portable and Interoperable Eigensolver Library

**PHIST** (Pipelined Hybrid Parallel Iterative Solver Toolkit) sparse solver framework
- General-purpose block Jacobi-Davidson Eigensolver, Krylov methods
- Preconditioning interface
- C, C++, Fortran 2003 and Python bindings
- Backends (*kernel libs*) include **GHOST**, Tpetra, PETSc, Eigen, Fortran
- Can use **Trilinos solvers Belos** and **Anasazi**, independent of backend

Getting PHIST and GHOST
- [https://bitbucket.org/essex/[ghost,phist]](https://bitbucket.org/essex/[ghost,phist])
- Cmake build system
- Available via Spack
- [https://github.com/spack/spack/](https://github.com/spack/spack/)
- PHIST will join Extreme-Scale Development Kit, [https://xSDK.info/](https://xSDK.info/)
PHIST & GHOST – Interoperability & Performance

• **Anasazi** Block Krylov-Schur solver on **Intel Skylake CPU**

• Matrix: non-sym. 7-pt stencil, \( N = 128^3 \) (var. coeff. reaction/convection/diffusion)

![](chart.png)

- **Anasazi’s** kernel interface mostly a subset of PHIST → extends PHIST by e.g. BKS and LOBPCG
- **Trilinos** not optimized for block vectors in **row-major storage**

Anasazi: https://trilinos.org/packages/anasazi/
Tpetra: https://trilinos.org/packages/tpetra/
PHIST & GHOST – Interoperability & Performance

- **Anasazi** Block Krylov-Schur solver on **Nvidia VOLTA GPGPU**
- Matrix: non-symmetric 7-point stencil, $N = 128^3$ (var. coeff. reaction/convection/diffusion)

- **Anasazi’s** kernel interface mostly a subset of PHIST $\rightarrow$ extends PHIST by e.g. BKS and LOBPCG
- Trilinos not optimized for block vectors in **row-major storage**

Anasazi: https://trilinos.org/packages/anasazi/
Tpetra: https://trilinos.org/packages/tpetra/
CRAFT: Application-level Checkpoint/Restart & Automatic Fault Tolerance

Application-level Checkpoint/Restart (CR):
• Simple & extendable interface to integrate CR functionality with minimal code changes
• Node-level CR using SCR, asyn. CP., Multi-stage & Nested CPs, signal based CP

Automatic Fault Tolerance (AFT) using CR
• Define ‘AFT-zones’ for automatic communication recovery in case of process failures.
• Detection and recovery methods from User-level Failure Mitigation (ULFM) MPI-ULFM.

Goal: Low programming & performance overhead

Tested Applications:
• GHOST & PHIST applications from ESSEX
• pFEM-CRAFT [Nakajima (U.Tokyo)]

https://bitbucket.org/essex/craft
ScaMaC: Scalable Matrix Collection

**Goal:** Collection of parametrized sparse matrices for eigenvalue computations from (quantum) physics

**Features:**

- "Scalable" matrix generator instead of fixed-size matrices
- Compatible with PETSc, Trilinos, GHOST, PHIST ...
- "Real World" (quantum) physics matrices, e.g.
  - wave & advection-diffusion eqs.,
  - correlated systems,
  - graphene & topological insulators,
  - quantum optics, (c)QED, optomechanics,...
- Real & complex, symmetric & non-symmetric, easy & hard to solve matrices
- Generating matrices of dimension $10^{11}$ in less than 30s on full scale OFP (0.5 Mcores)
Recursive Algebraic Coloring Engine (RACE)

Graph coloring: RACE uses recursive BFS level based method for “distance-k coloring” of symmetric matrices

Objectives
• Preserve data locality
• Generate sufficient parallelism
• Reduce synchronization
• Simple data format like CRS

Applications – Parallelization of
• iterative solvers, e.g. Gauß-Seidel & Kaczmarz
• sparse kernels with dependencies, e.g. symmetric spMVM

Example: Node-level parallelization of symmetric spMVM (distance-2)

Intel Skylake (20 cores)

Compare with
• Intel MKL
• RSB (data format)
• Multicoloring
Recursive Algebraic Coloring Engine (RACE)

**Graph coloring:** RACE uses recursive BFS level based method for “distance-k coloring” of symmetric matrices

**Objectives**
- Preserve data locality
- Generate sufficient parallelism
- Reduce synchronization
- Simple data format like CRS

**Applications – Parallelization of**
- iterative solvers, e.g. Gauß-Seidel & Kaczmarz
- sparse kernels with dependencies, e.g. symmetric spMVM

Example: Node-level parallelization of symmetric spMVM (distance-2)

Compare with
- Intel MKL
- RSB (data format)
- Multicoloring
Scalability on Oakforest-PACS since 6 / 2018 number 12 of Top 500

| Cores: | 556,104 |
| Memory: | 919,296 GB |
| Processor: | Intel Xeon Phi 7250 68C 1.4GHz (KNL) |
| Interconnect: | Intel Omni-Path |

| Linpack Performance (Rmax) | 13.554 PFlop/s |
| Theoretical Peak (Rpeak) | 24.913 PFlop/s |

| Nmax HPCG [TFlop/s] | 9,938,880 385.479 |

Impression of the Oakforest-PACS supercomputer at the Japanese Joint Center for Advanced HPC (JCAHPC)
Weak Scaling: Jacobi-Davidson Method for Computing Extreme Eigenvalues

- Up to 0.5M cores
- Percentage indicates the parallel efficiency compared to the first measurement (smallest node count).
- Symmetric PDE problem with the largest matrix size \( N = 4\,0963 \),
- The best performance was obtained with a block size of 4.
Large Scale Performance – Weak Scaling

Computing 100 inner eigenvalues on matrices up to $n = 4 \times 10^9$

Typical Application[1]:
Topological Insulator

How to ensure the quality of the ESSEX software: Basics

- **Git** for distributed software development

- **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
Parallel Coupling Frameworks
FlowSimulator

- HPC environment for integration of multiple parallel components into a process chain

- Jointly developed by Airbus, DLR, ONERA, universities, ...

- Components of simulation process chain ("Plug-ins") integrated via
  - Python control interface
  - FSDM data interface
FlowSimulator DataManager (FSDM)

- FSDM reads/writes data (mesh, solution, log-data) from/to files
- FSDM decomposes data and distributes it over the different MPI domains
- FSDM stores data in container classes (e.g. FSMesh, FSDataset)
- FSDM offers an interface (Python and C++) to container classes
- FSDM for us means unstructured meshes, can handle structured meshes as well
- FSDM is Open Source
the Flexible Unstructured CFD Software

- The “next generation” flow solver currently developed at DLR by the Institute for Aerodynamics and Flow Technology

- Solves the Euler-equations, the Navier-Stokes equations, or the RANS equations

- Two discretizations
  - Second-order Finite-Volume
  - Discontinuous Galerkin

- Flucs is designed as an FS plug-in in order to facilitate multi-disciplinary simulations

- Consequently, development of FSDM and Flucs has to go hand in hand
Versatile Aeromechanic Simulation Tool (VAST)

- Multi-physics and multi-model simulation tool for the description of the aeromechanics of helicopters

- **Goal:** specification of the dynamic behavior, e.g., for the description of helicopters in free flight

- **Generic approach:** framework, numerical methods, user interface designed for general multi-model simulations

- **Modeling** as coupled ODEs: independent development of models for, e.g., rigid structures (MBS), flexible beams, rotor blade dynamics ...
VAST – Generic State Space Model

- **Config data:**
  - XSD-description of a model’s config parameters
  - Automatic generation of read-in method → TiXi

- **Model factory:** sets up and configures the model

- **Model definition:** defines the model to the outside:
  - States, inputs, outputs

- **Attributes:** Model parameters processed by the factory needed for computation

- **Optional Methods:** Gradients and linearization can be provided, otherwise they are computed by the solver when needed

- **Simulation Methods:**
  - $\dot{x} = f(u, x, t)$: Compute first time derivatives of the states
  - $y = g(u, x, t)$: Compute outputs

- **Config data:**
  - $x_i(t)$: Model internal States
  - $u_i(t)$: Model specific inputs

- **Output and Simulation Methods:**
  - $y_i(t)$: Outputs of the system
  - $\dot{x}_i(t)$: Time derivatives of the model internal states
**DLR Software Free-Wake**

- Developed from 1994 to 1996 by the DLR Institute of Flight Systems, Department Rotorcraft (FT-HS)
  - Implemented in Fortran
  - MPI-parallel

- Used by the FT-HS rotor simulation code S4

- Simulates the flow around a helicopter’s rotor
  - Vortex-Lattice method
  - Discretizes complex wake structures with a set of vortex elements
  - Based on experimental data from the international HART program 1995

- **Our task:** hybrid Free-Wake parallelization for CPUs and GPGPUs
Port to GPGPU and Modernization

- Successfully ported Free-Wake simulation to GPUs using OpenACC
  - original numerical method not modified
  - results verified in SP and DP on CPU and GPU
  - “time to solution” improved significantly (for SP and CPU+GPU)
- Porting complex algorithms to GPUs is difficult
  - branches in loops hurt (much more than for CPUs)
- Loop restructuring may also improve the CPU performance
  - SIMD vectorization on modern CPUs
- Stumbled upon several OpenACC PGI-compiler bugs (all fixed by now)
- Freewake on a workstation with reasonable cycle times → Goal achieved!
VAST + Free-Wake

“Trim” of a free-flying, multi-rotor helicopter with rigid blades
The Software Framework HeAT for Data Analytics with Parallel Machine Learning Methods
Helmholtz Analytics Framework (HAF)

- Joint effort of all 6 Helmholtz centers to foster data analytics within Helmholtz.

- Scope: Systematic development of domain-specific data analysis techniques in a co-design approach between domain scientists and information experts.
HAF Use Cases

Earth System Modelling

Research with Photons

Aeronautics and Aerodynamics

Structural Biology

Neuroscience
Greatest Common Denominator?

Machine Learning

= 

Lots of Data

+ 

Numerical Linear Algebra

Sounds like something HPC has been doing anyway.

https://xkcd.com/1838/
Which technology stack to use for an analytics toolkit?

- We do not want to reinvent the wheel, and there are already plenty of machine learning frameworks available.

- Common denominator: all frameworks provide Python as a front end language.
  - We also chose Python.

- Which framework to use as a basis?
  - It is better to measure than to guess. Benchmark!
Technology Benchmarks

- Testing several Machine Learning Frameworks
  - by implementing
    - K-means
    - Self-Organizing Maps (SOM)
    - Artificial Neural Networks (ANN)
    - Support Vector Machines (SVM)
  - Evaluation criteria:
    - Feature completeness
    - Ease of implementation
    - Performance (on-going effort)
Evaluation

- Feature completeness:

<table>
<thead>
<tr>
<th>Framework</th>
<th>GPU</th>
<th>MPI</th>
<th>AD</th>
<th>LA</th>
<th>nD Tensors</th>
<th>SVD</th>
<th>Dist. tens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pytorch</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>TensorFlow</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>MXNet</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Arrayfire</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>

- Ease of implementation and usage: Pytorch and MXNet
- Performance: Pytorch and MXNet

- Since we need MPI for distributed parallelism, we have chosen Pytorch as a basis for our Machine Learning toolkit: HeAT
What is HeAT?

- **HeAT** = Helmholtz Analytics Toolkit
- The hot new machine learning / data analytics framework to come.
- Developed in the open:
  Available on https://github.com/helmholtz-analytics and https://pypi.org/project/heat
- Liberally licensed: MIT
Facilitating HAF Use Cases in their work

Bringing HPC and Machine Learning / Data Analytics closer together

Ease of use

HeAT

- k-means
- SVM
- NN

PyTorch

- Tensor Linear Algebra
- Automatic Differentiation
- Pythonic numpy-like interface
- GPU support
- Distributed Parallelism (MPI)
What has been done so far?

• The core technology has been identified

• Implementation of a distributed parallel tensor class has begun

• A first implementation of the k-means algorithm is available
Transparent development process

Github for code review, issue tracking, sprint planning

Travis for continuous integration

Mattermost for discussions

Feel free to join us there!
Quantencomputing
Software for Quantum Computers

Challenges:

• Quantum computer interfaces are close to hardware

• Superior quantum algorithms still under investigation

Embedding for Quantum Annealing

Compiling of Quantum Circuits

Google QC Chip
Our Approach

- Develop **software and algorithms** for the whole stack from application mapping down to compiling
- **Keep it flexible**: Be able to react to fast changing hardware developments
- **Open and collaborative**
  - Joined software development with NASA Ames QC group (in progress)
  - Use of open-source tools and abstraction layers to be vendor agnostic
  - Planned to be released as Open Source

- DLR Quantum Annealing Library
  - Python based, high quality software (test coverage, continuous integration)
DLR Quantum Annealing Library

- Binary encoding
- Constraint substitution
- Embedding strategies
- Parameter scanning
- Performance evaluation
- Storage & plotting
- Classical (pre-) solving
Applications

Flight Gate Assignment

Anomaly Detection in Satellite Telemetry Data

Air Traffic Management

Earth Observation Satellite Mission Planning
Many thanks for your attention!

Questions?

Dr.-Ing. Achim Basermann
German Aerospace Center (DLR)
Simulation and Software Technology
Department Head High-Performance Computing
Achim.Basermann@dlr.de
http://www.DLR.de/sc

We are hiring: http://www.dlr.de/jobs/