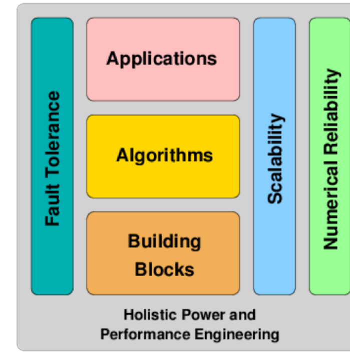
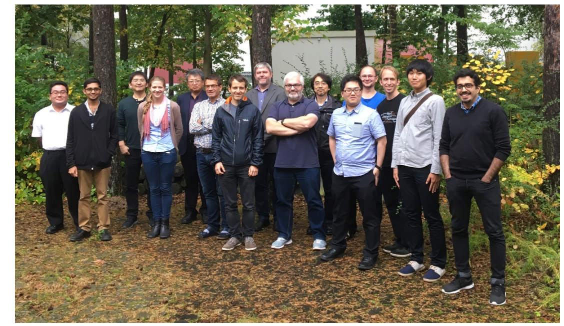
# ESSEX II – (Selected) Results and Perspectives



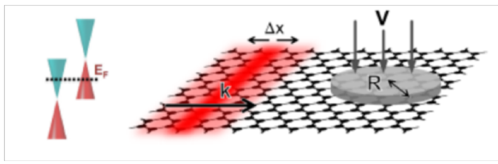




Gerhard Wellein — Computer Science, University Erlangen
Bruno Lang — Applied Computer Science, University Wuppertal
Achim Basermann — Simulation & SW Technology, German Aerospace
Holger Fehske — Institute for Physics, University Greifswald
Georg Hager — Erlangen Regional Computing Center
Tetsuya Sakurai — Applied Mathematics, University of Tsukuba
Kengo Nakajima — Computer Science, University of Tokyo


Paris, March 21st , 2019

# ESSEX project – background

## Quantum physics/information applications




Large, Sparse

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

and beyond....

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

"Few" (1,…,100s) of eigenpairs

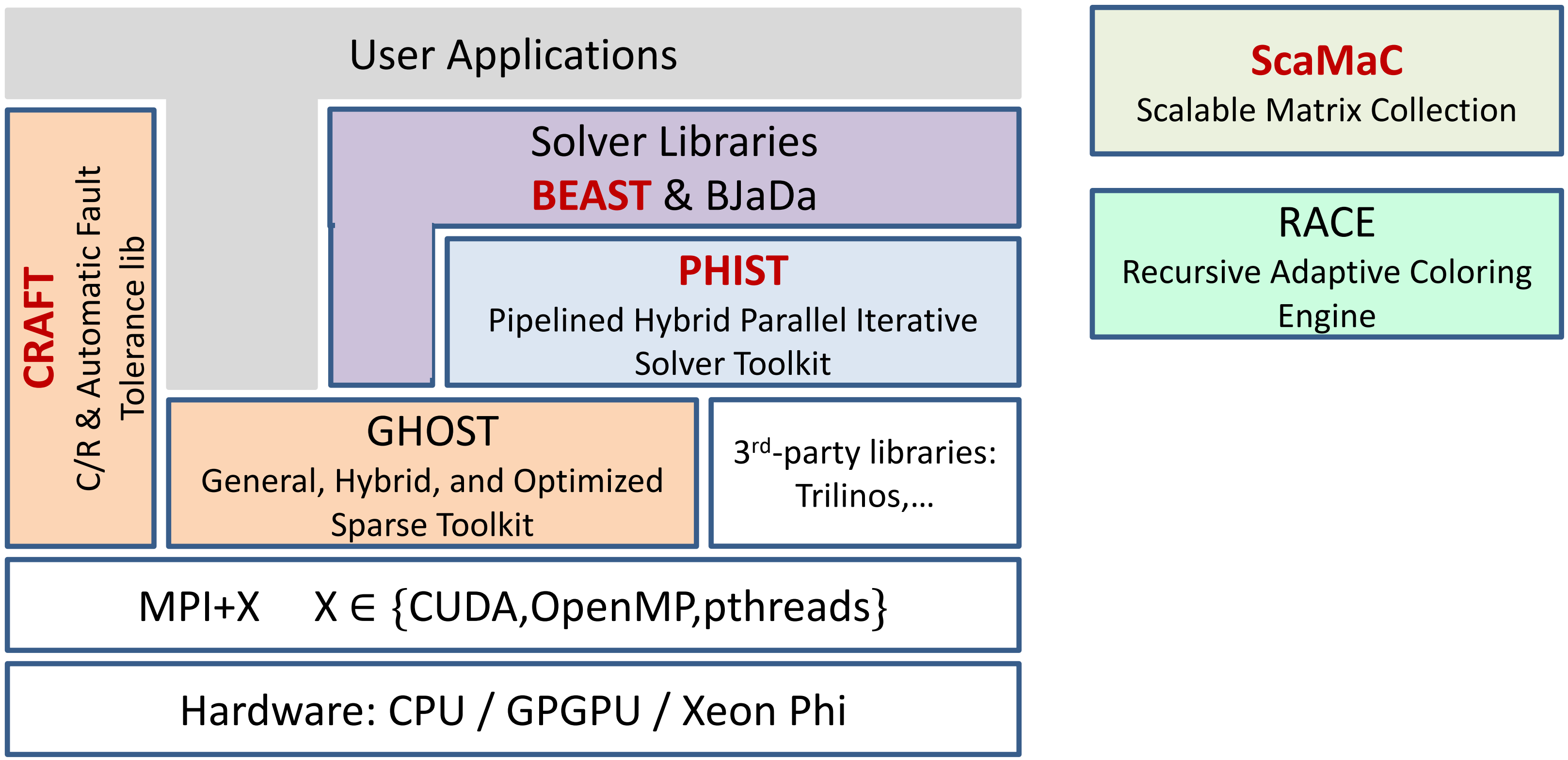
"Bulk" (100s,…,1000s) eigenpairs

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

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

→ Sparse eigenvalue solvers of broad applicability

# ESSEX-II: Software Packages

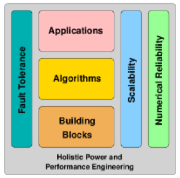



Links to open source repositories at **https://blogs.fau.de/essex/code**

Major goals in 2018/9:
Improve software quality, documentation and interoperability
Broaden Application Fields

# CRAFT
# Checkpoint/Restart & Fault-Tolerance

# CRAFT library: Application-level Checkpoint/Restart (C/R) & Automatic Fault Tolerance (AFT)




## CRAFT library [1] : Building blocks for fault tolerant (FT) applications

1. Application-level CR functionality with minimal code changes.
   C/R: Node-level CR using SCR, async., multi-stage & nested, signal based.

2. AFT functionality for automatic com. recovery in case of process failures.
   AFT based on MPI-ULFM

## Applications using CRAFT:

- ESSEX solver / applications
- FEM solver (Univ. of Tokyo) [2]
- EXASTEEL-II (SPPEXA)

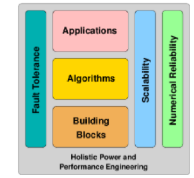
***https://bitbucket.org/essex/craft***

[1] Shahzad, Thies, Kreutzer, Zeiser, Hager, Wellein: *CRAFT: A library for easier application-level checkpoint/restart and automatic fault tolerance.* IEEE Transactions on Parallel and Distributed Systems (2018)

[2] Fukasawa, Shahzad, Nakajima, and Wellein. pFEM-CRAFT: A Library for Application-Level Fault-Resilience Based on the CRAFT Framework. *Poster at the SIAM PP18,* Tokyo, Japan.

# PHIST
# Flexible Iterative Solver Toolkit

# 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 (e.g. Block-ILU (Nakajima))
- C, C++, Fortran 2003 and Python bindings – Performance testing
- Backends (kernel libs) include GHOST, Tpetra, PETSc, Eigen, Fortran
- Can use Trilinos solvers Belos and Anasazi, independent of backend

- Tested on TOP10 systems
  - KNL (Oakforest)
  - Piz Daint (NVIDIA)
  - SuperMUC-NG (SKX) - ongoing

- Getting PHIST: https://bitbucket.org/essex/phist
- Cmake build system
- Availale via Spack https://github.com/spack/spack/

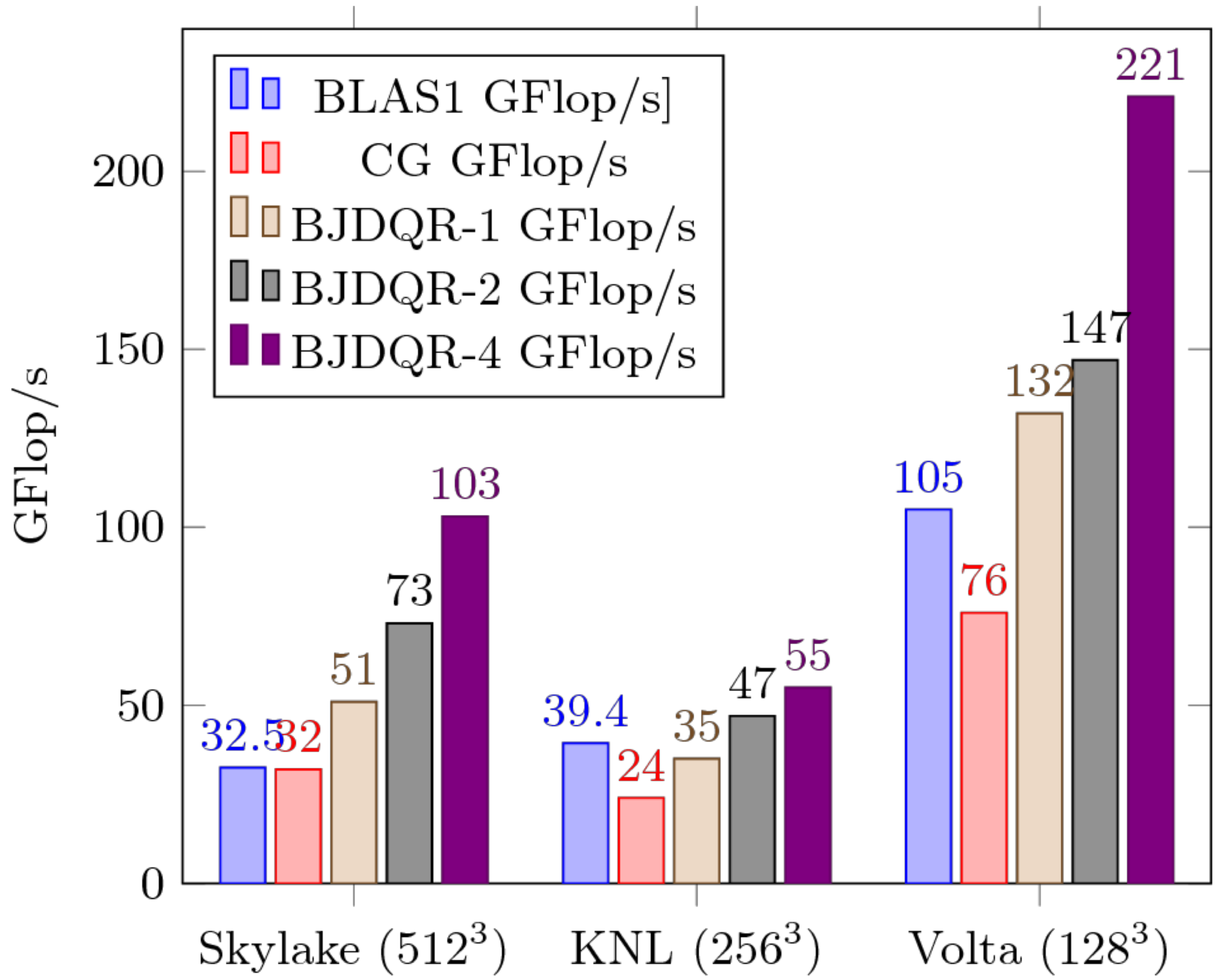
**xSDK num**

Packages

- AMReX
- deal.ii
- DTK
- hypre
- MAGMA
- MFEM
- Omega_h
- PETSc/TAO
- PHIST
- PLASMA
- PUMI
- SLEPc
- STRUMPACK
- SUNDIALS
- SuperLU
- Tasmanian
- Trilinos

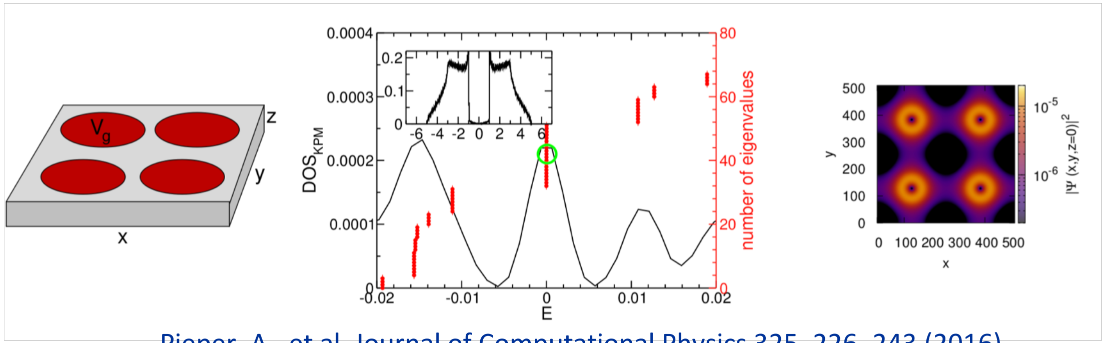
**Extreme-Scale Development Kit**
https://xSDK.info/

# PHIST/GHOST: Node-Level Performance of Jacobi-Davidson (symm. PDE problem)



# BEAST

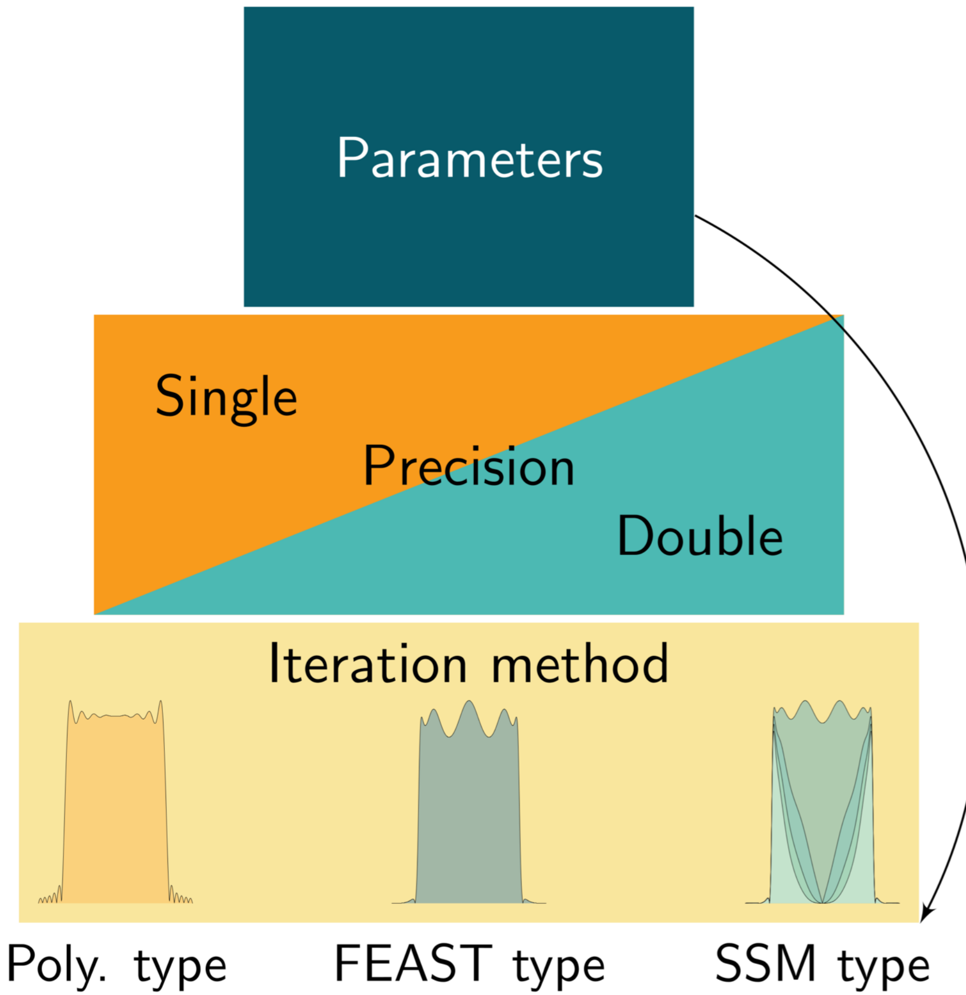
# Iterative, adaptive algorithms to compute interior eigenvalues



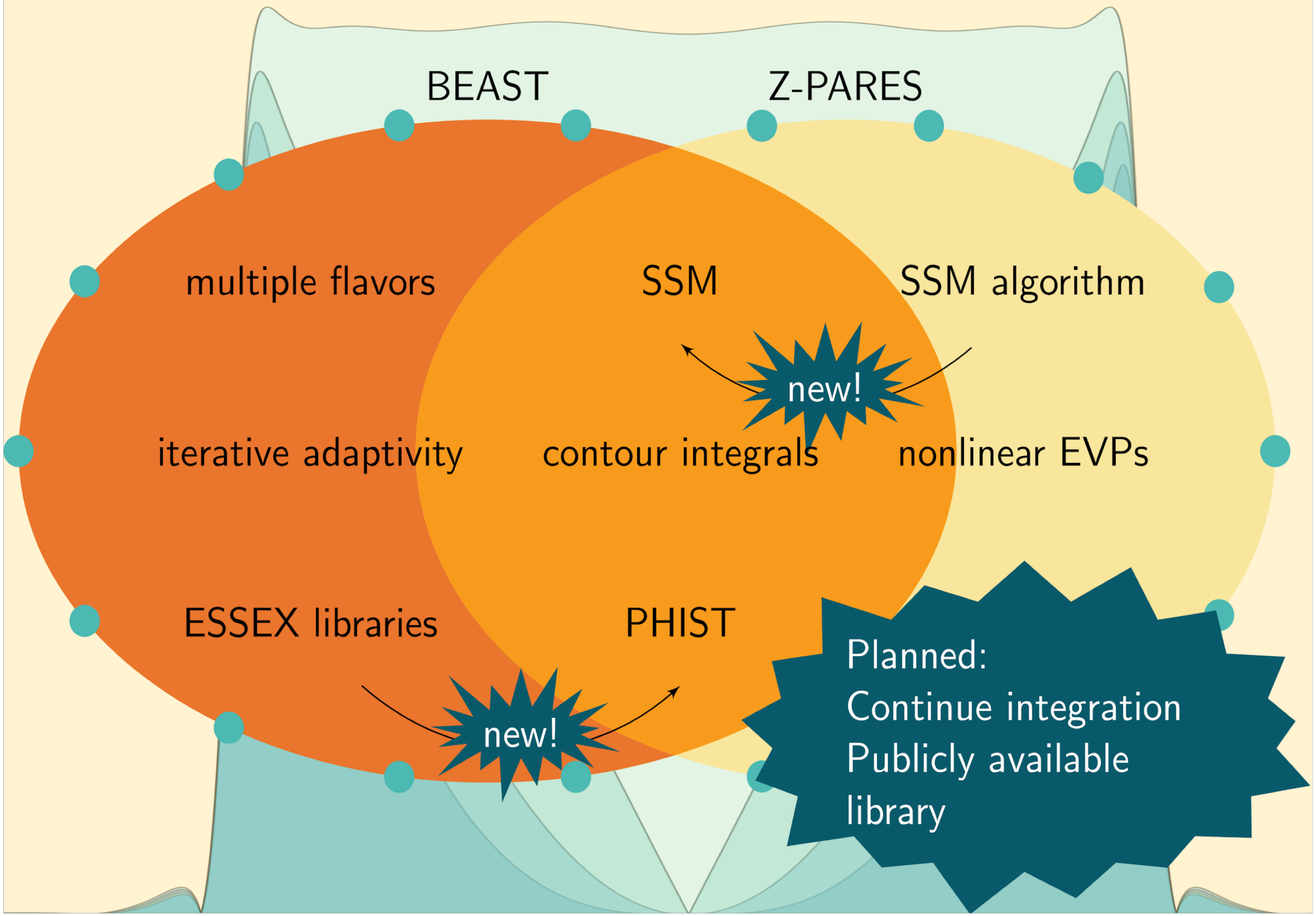

Pieper, A., et al. Journal of Computational Physics 325, 226–243 (2016)

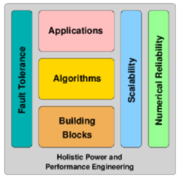
# BEAST framework

- Subspace iterative eigensolver
  - Change projector/extraction method between iterations
- Adaptive choices
  - Switching between methods
  - Precision
  - Algorithmic parameters
- Efficient convergence
  - Early iterations less expensive
  - High accuracy still possible
- PHIST implementation
- Use highly optimized GHOST kernels (e.g. Chebyshev-filter)
- Ongoing collaboration with T. Sakurai (SSM / z-Pares)



# BEAST and Z-PARES: shared tools for large EVPs



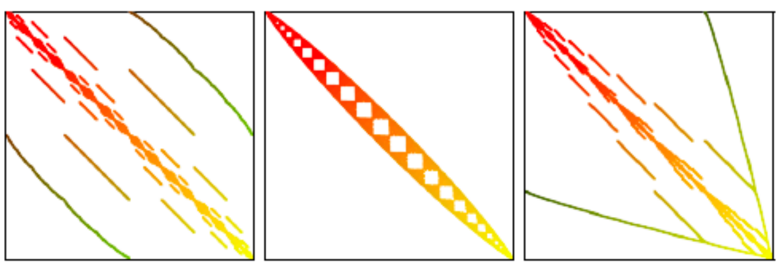
# Scalable Matrix Collection (**ScaMac**) & Quantum Physics Application

# ScaMaC: Scalable Matrix Collection

## Collection of parametrized sparse matrices for eigenvalue computations

- scalable matrix generators instead of fixed-size matrices
  distributed matrix generation compatible with PETSc, Trilinos, GHOST, PHIST ...
- "real world" matrices mainly from classical & quantum physics
  wave & advection-diffusion eqs., correlated (fermion, boson, spin) systems,
  graphene&topological insulators, quantum optics, (c)QED, optomechanics, ...
- real & complex, symmetric & non-symmetric, easy & hard matrices

```
=== Example: FreeFermionChain        ===
=== parameter: n_fermions, n_sites ===
```



Hubbard-N24
( < 1 min @ OakforestPACS/1024 nodes)
DIM = 9'658'958'400, NNZ = 158'682'888'000

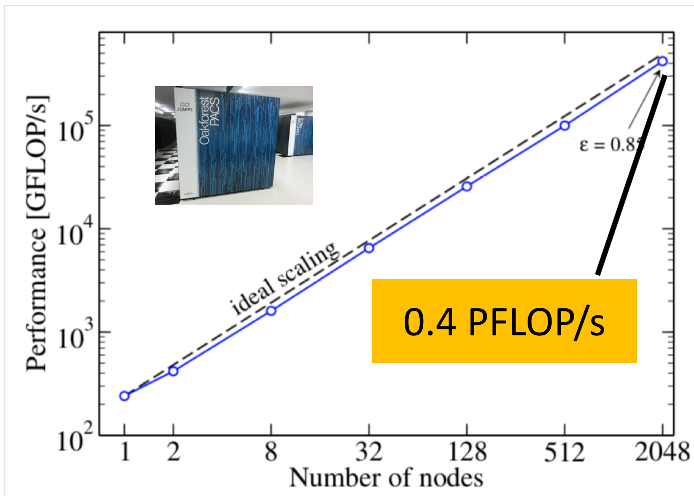
| n_fermions | n_sites | dimension |
|---|---|---|
| 5 | 10 | 252 |
| 6 | 12 | 924 |
| 7 | 14 | 3 432 |
| 8 | 16 | 12 870 |
| 9 | 18 | 48 620 |
| 10 | 20 | 184 756 |
| 11 | 22 | 705 432 |
| 12 | 24 | 2 704 156 |
| 13 | 26 | 10 400 600 |
| 14 | 28 | 40 116 600 |
| 15 | 30 | 155 117 520 |
| 16 | 32 | 601 080 390 |
| 17 | 34 | 2 333 606 220 |
| 18 | 36 | 9 075 135 300 |
| 19 | 38 | 35 345 263 800 |
| 20 | 40 | 137 846 528 820 |

# Scalability

# Large Scale Chebyshev Filter Diagonalization

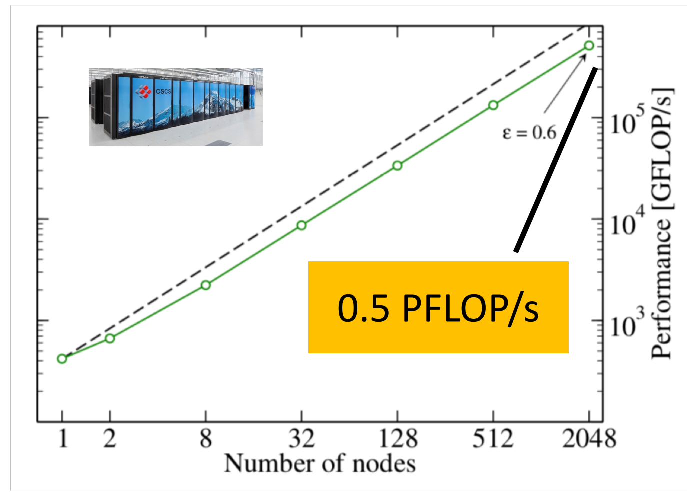
Computing 100 inner eigenvalues on matrices up to $n = 4\times10^9$

$$\frac{n}{node} = 2.1\times10^6$$
$$n_p = 500;\ n_s = 128$$




8208 nodes (Intel Xeon **Phi7250**)
Intel OmniPath (full fat tree)
#9 – TOP500 11/2017
$R_{peak}$= 24.9 PF/s
$R_{max}$ = 13.5 PF/s

Oakforest-PACS




5320 nodes (**1xP100** + 1x Xeon E5..v3)
Cray Dragonfly network
#3 – TOP500 11/2017
$R_{peak}$= 25.3 PF/s
$R_{max}$ = 19.6 PF/s

Piz Daint

# Perspectives

# Eigenvalue Problems in unsupervised ML

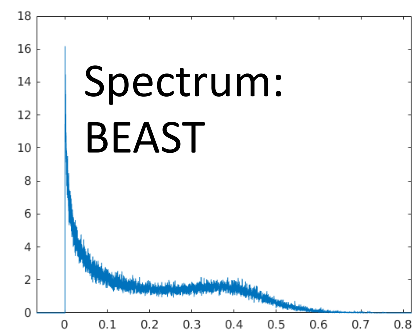
Thanks to Marina Meila (UW)

- Examples:
  - PCA (Covariance matrix)
  - spectral clustering (stochastic matrix/Laplacian matrix)
  - Manifold learning (real symmetric matrix, or Laplacian matrix)
- In all methods, data mapped to principal eigenvectors of matrix A :

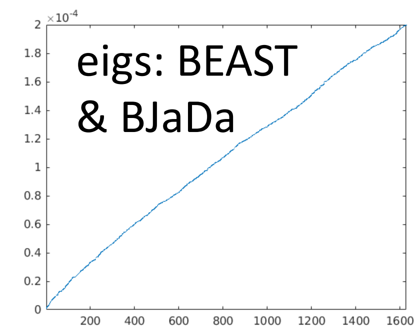
$$[\lambda, V] = \text{eigs}(A, m)$$

- Choice of m be? → data dependent
- How can eigenvalues help?
  - eigenvalues inform on the geometry of the problem and therefore on the choice of m
  - PCA: covariance explained
  - spectral clustering: cluster separation

→ detecting large eigengaps for geometric and numeric stability

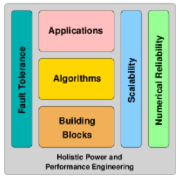



0,5 M rows Spectrum & 1600 eigenvals




Collaboration: J. Thies (DLR) B. Lang (WUP) & M. Meila (U. of Washington)

# Supervised ML on quantum physics problems




- Idea:
  - Use classic quantum physics methods (**ED**, DMRG, QMC,...) to generate data sets for supervised ML
  - Predict properties by trained network (which are not accessible by classic methods)

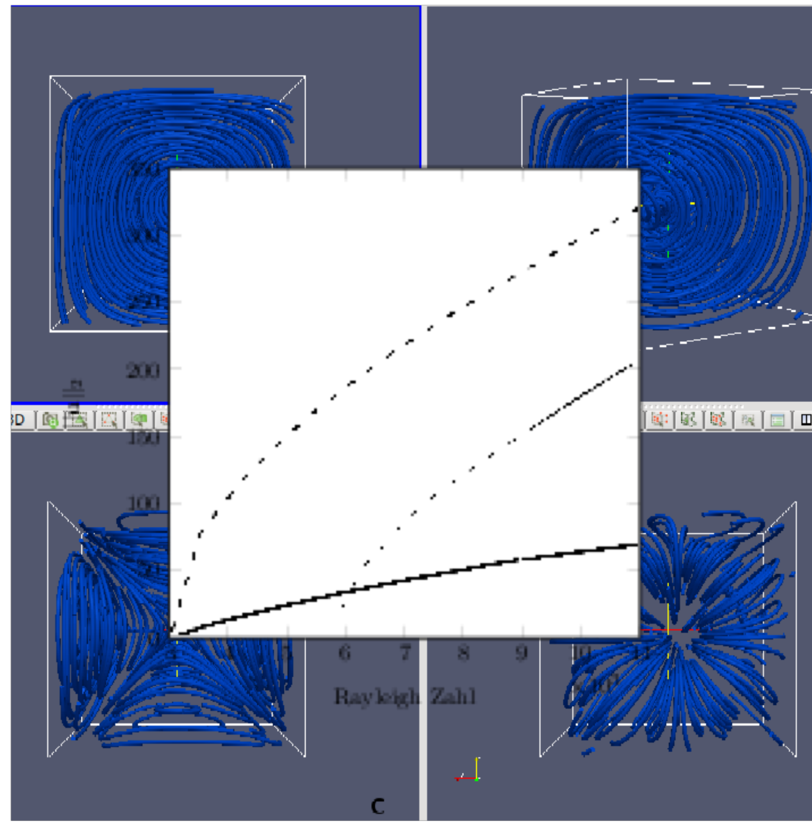
- Many ongoing research directions, e.g.
  - Learn topological phases for finite systems in real space: arXiv:1901.01963
  - ML on static functions to characterize photoexcited states (in 1D EHM) see K. Shinjo, S. Sota, S. Yunoki, T. Tohyama, arXiv:1901.07900

    Upcoming Collaboration:
    H. Fehske (HGW) K. Nakajima (RIKEN) & S. Yunoki (RIKEN)

# Employing **PHIST** for studying dynamical systems

- Continuation of steady states
  - Using Trilinos libraries LOCA+Epetra
  - Linear stability analysis using **Jacobi-Davidson** (PHIST)
  - Multi-Level ILU preconditioner HYMLS
  - (https://github.com/nlesc-smcm/hymls)

- Work in progress:
  - Automatic exploration of parameter space using PyNCT+PHIST
  - Applications: climate modelling, non-Newtonian fluids




Bifurcation diagram for 3D Rayleigh-Bénard convection (dashed lines: unstable states)

Ongoing collaboration: J. Thies (DLR) & W. Vanroose (U. Antwerp)

# Thank you

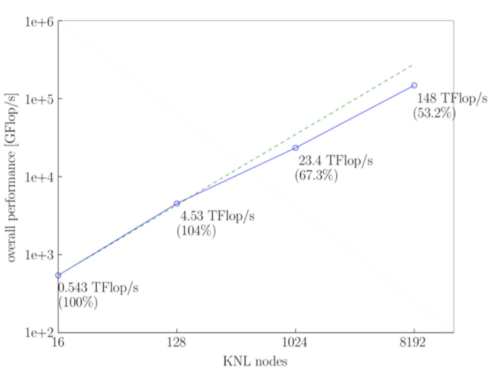
# Backup

# Oakforest PACS HPC Challenge

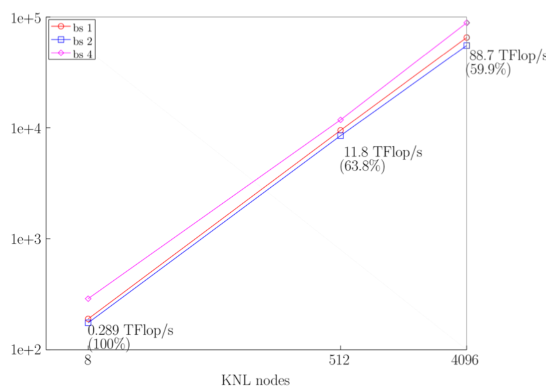

- Large-Scale HPC Challenge
- Oakforest-PACS (Tokyo, Japan)
- 8208 nodes, 24 hours
- Results from various ESSEX packages

JaDa weak scaling

Symmetric PDE



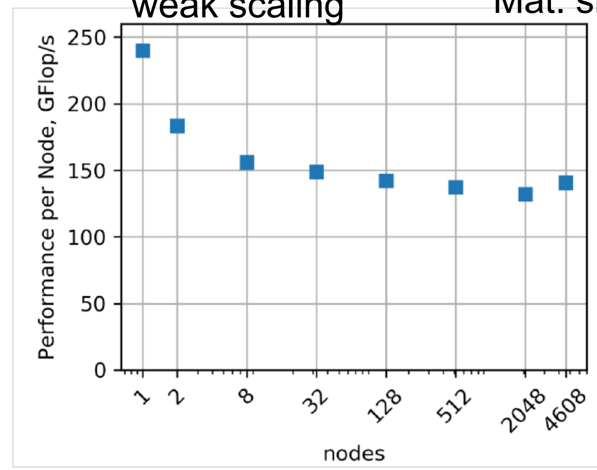

Non-symmetric PDE



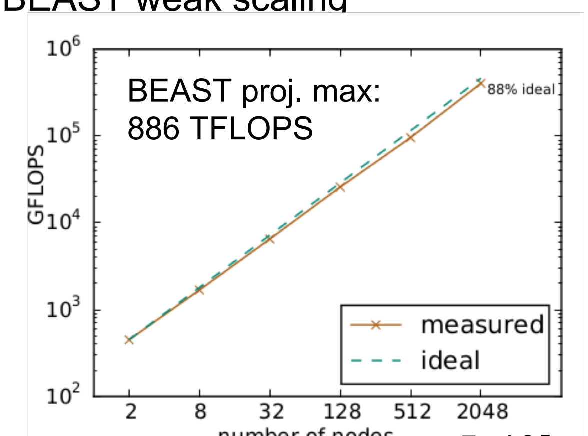

- MINRES/GMRES preconditioning
- 20 eigenpairs near 0
- Largest problem solved: 69B DoF on 0.5 M cores

GHOST-ChebFD weak scaling

Mat. size $10^{10}$



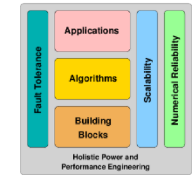

BEAST weak scaling




$5 \times 10^5$ rows/node

# ScaMaC: Scalable Matrix Collection

(major update v0.8 → v0.9 ≈ March 2019)




- Benchmark & „real world“ matrices from classical & quantum physics
- Approach: generate matrices row by row in parallel
  compatible with PETSc, Trilinos, GHOST, PHIST

```
// step 1: obtain a generator - per process
ScamacGenerator * my_gen;
err = scamac_parse_argstr("Hubbard,n_sites=20", &my_gen, &errstr);
err = scamac_generator_finalize(my_gen);
    ................
// step 2: allocate workspace - per thread
ScamacWorkspace * my_ws;
err = scamac_workspace_alloc(my_gen, &my_ws);
    ................
// step 3: generate the matrix row by row
ScamacIdx nrow = scamac_generator_query_nrow(my_gen);
for (idx=0; idx<nrow; idx++) { // parallelize loop with OpenMP, MPI, ...
    // obtain the column indices and values of one row
    err = scamac_generate_row(my_gen, my_ws, idx, SCAMAC_DEFAULT, &nz, cind, val);
    // store or process the row
        ................
}
// step 4: clean up
err = scamac_workspace_free(my_ws);      // in each thread
err = scamac_generator_destroy(my_gen); // in each process
// step 5: use matrix
    ................
```

- Example: Hubbard-N24 (generation: < 1 min @ OakforestPACS/1024 nodes)
  DIM = 9'658'958'400, NNZ = 158'682'888'000

# Awards

- Hans Meuer Award Nominee (ISC 2018):
  *Chebyshev Filter Diagonalization on Modern Manycore Processors and GPGPUs* by M. Kreutzer, D. Ernst, A. R. Bishop, H. Fehske, G. Hager, K. Nakajima, and G. Wellein

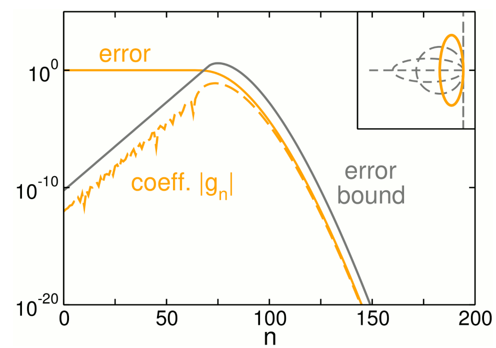
- C.L. Alappat (FAU) wins
  ACM Student Research Competition at SC18
  *Recursive Algebraic Coloring Engine (RACE)*


- S. Huber (BUW) and C.L. Alappat receive poster awards at CoSaS 2018 workshop at FAU Erlangen-Nürnberg

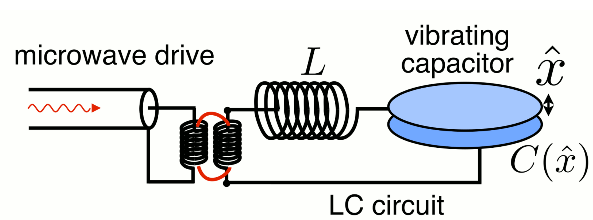
# ChebTP: Time-propagation for non-Hermitian matrices with Faber-Chebyshev polynomials

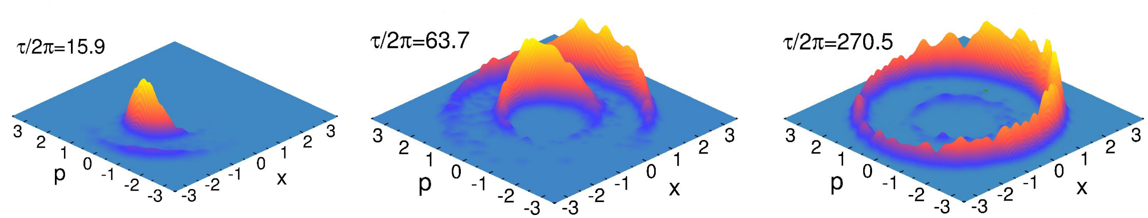
- HPC alternative to the more common Krylov approach
  → no scalar products, less synchronization, optimized kernels
- re-use the „Chebyshev core“ (GHOST → KPM, ChebFD, ...)
  with predetermined coefficients $g_n$ according to the ChebTP analysis

```
for k = 1 to N_S do                                  ▷ First two recurrence steps
    u_k = (αH + β𝟙) x_k                              ▷ spmv()
    w_k = 2(αH + β𝟙) u_k − x_k                       ▷ spmv()
    x_k = g_0 c_0 x_k + g_1 c_1 u_k + g_2 c_2 w_k    ▷ axpy & scal
end for
for n = 3 to N_p do                                  ▷ Remaining recurrence steps
    for k = 1 to N_S do
        swap(w_k, u_k)                               ▷ swap pointers
        w_k = 2(αH + β𝟙) u_k − w_k                   ▷ spmv()
        x_k = x_k + g_n c_n w_k                      ▷ axpy
    end for
end for
```




- Example: Quantum multistability in optomechanics

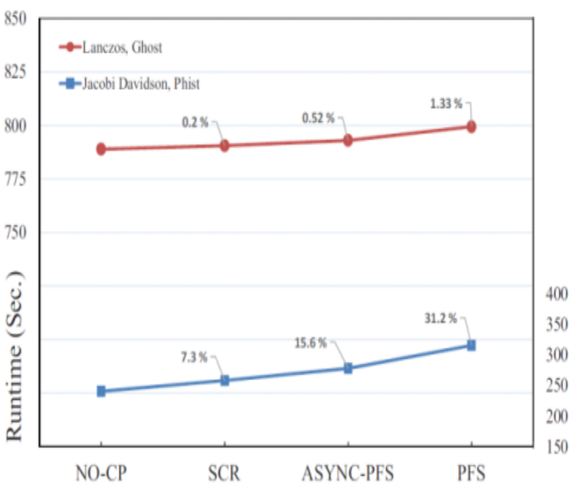


# CRAFT applications/benchmarks within ESSEX framework

## Jacobi-Davidson EV solver

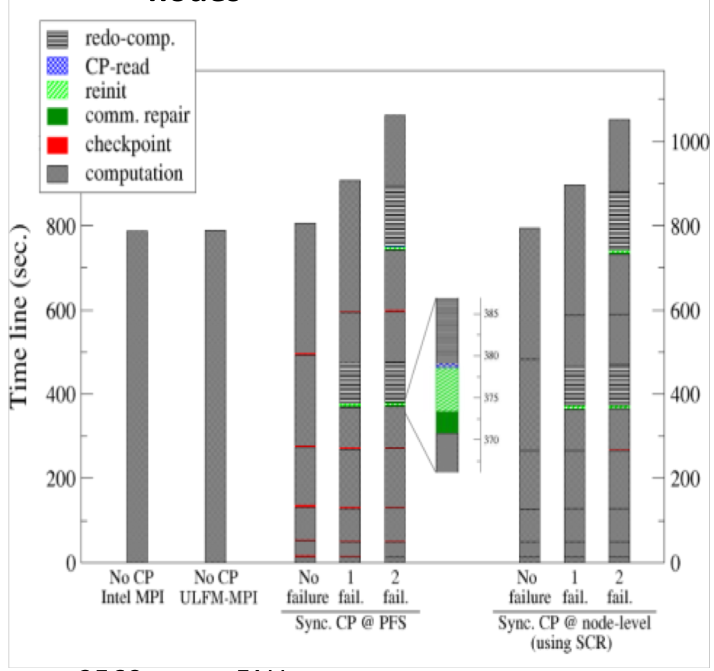
- ➢ PHIST implementation of JDQR
- ➢ CP data: PHIST data vectors
- ➢ FT scope:
  - Application-level C/R




- Overhead comparison
- 2560 cores, FAU

## Lanczos EV solver

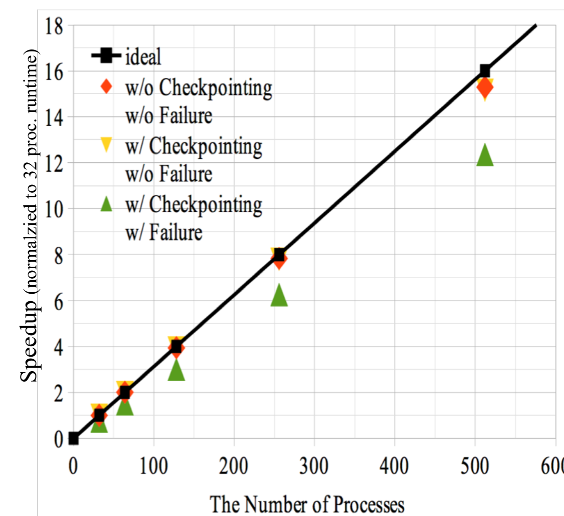
- ➢ GHOST library for sparse linear algebra
- ➢ CP data: Ghost data vectors
- ➢ FT scope:
  - Application-level C/R
  - Dynamic process recovery
  - **Non-shrinking recovery with idle nodes**




- 2560 cores, FAU
- Main overhead → redo of lost work.

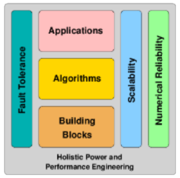
## FEM solver

- ➢ FEM solver for 3D Non-Steady Heat Conduction problem
- ➢ CP data: 1D- POD arrays
- ➢ FT scope:
  - Application-level C/R
  - Dynamic process recovery
  - **Shrinking recovery with domain load-balancing using ParMETIS**




- Application Speedup with various CRAFT-enabled FT cases.
- Strong scaling up to 512 cores of Reedbush-U (Univ. of Tokyo)

# PHIST Activities 2018




- Integration of several ESSEX efforts within PHIST
  - CRAFT, ScaMaC
  - Block ILU preconditioner (PI Nakajima)
  - Fortran 2003 bindings

- PHIST part of the "Extreme-scale Development Kit" (https://xsdk.info); major US centers

**xSDK numerical libraries:**

- AMReX
- deal.ii
- DTK
- hypre
- MAGMA
- MFEM
- Omega_h
- PETSc/TAO
- PHIST
- PLASMA
- PUMI
- SLEPc
- STRUMPACK
- SUNDIALS
- SuperLU
- Tasmanian
- Trilinos

- PHIST/GHOST performance
  - Skylake, KNL, Nvidia Volta
  - Petascale machine: Oakforest-PACS

- ACM-TOMS paper in the pipeline: [https://elib.dlr.de/123323/1/phist2018.pdf](https://elib.dlr.de/123323/1/phist2018.pdf)