ESSEX-II: Equipping Sparse Solvers for Exascale

Gerhard Wellein University of Erlangen Department of Computer Science	**Achim Basermann** German Aerospace Center	**Holger Fehske** University of Greifswald Institute for Physics	**Georg Hager** Erlangen Regional Computing Center	**Bruno Lang** University of Wuppertal Applied Computer Science
Resilience, Performance Engineering, Parallelization, Optimization	Scalable Preconditioners, Eigenvalue & Linear Solvers	Quantum Physics Algorithms & Applications	Performance Engineering, Tools, Parallelization, Optimization	Efficient Direct and Iterative Eigensolvers

Tetsuya Sakurai University of Tsukuba Department of Computer Science	**Kengo Nakajima** University of Tokyo Information Technology Center
Advanced Eigensolvers	Scalable Preconditioners, AMG

Collaborations

H. Anzt	**A. R. Bishop**	**E. Romero Alcade**	**Olaf Schenk**
Blocked sparse kernels	Physics applications	Mixed precision solvers	Multi-coloring

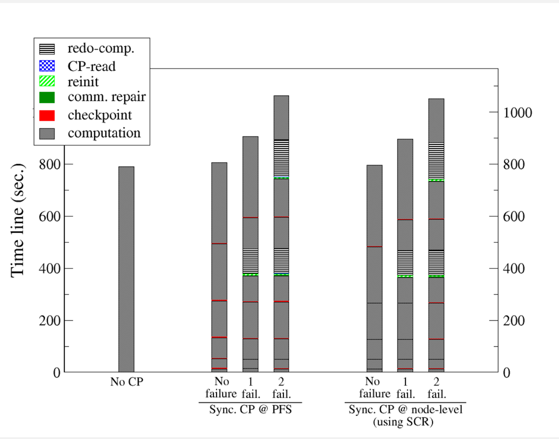
blogs.fau.de/essex

CRAFT: Efficient Checkpoint/ Restart & Automatic FT

- C++ abstraction for easy C/R and Automatic Fault Tolerance (AFT)
- Basic data types provided; user-extensible with custom data types
- AFT based on MPI-ULFM
- Shrinking & non-shrinking recovery
- C/R based on MPI-I/O or the SCR library
- Multiple checkpoints
- Nested checkpoints

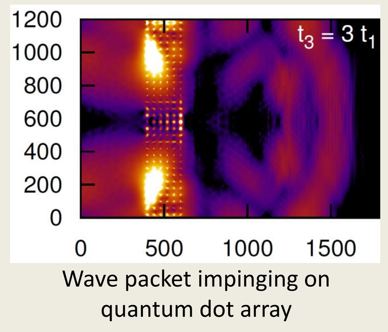
→ Basic structure of code with C/R and AFT facilities via CRAFT

```
#include <mpi.h>
#include <craft.h>
int main(int argc, char* argv[]){
  ...
  int myrank, iteration = 0, cpFreq = 10;
  MPI_Comm FT_Comm;
  MPI_Comm_dup(MPI_COMM_WORLD, &FT_Comm);

  AFT_BEGIN(FT_Comm, &myrank, argv);

  double data = 0;
  Checkpoint  myCP( "myCP", FT_Comm);
  myCP.add("data", &data);
  myCP.add("iteration", &iteration);
  myCP.commit();
  myCP.restartIfNeeded(&iteration);
  for(; iteration <= n; iteration++){
    /* Computation-communication */
    myCP.updateAndWrite(iteration, cpFreq);
  }
  ...

  AFT_END();

}
```



← A Lanczos benchmark showing the CR and AFT overheads on 128 IVB nodes. The average communication recovery time is 2.6 sec.

Starting point: Quantum physics & information applications



Wave packet impinging on quantum dot array

$$i\eta \frac{\partial}{\partial t}\Psi(r,t) = H\Psi(r,t)$$

$$H\,x = \lambda\,x$$

Large, ↗ Sparse

"Few" (1,…,100s) of eigenpairs

"Bulk" (100s,…,1000s) eigenpairs

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

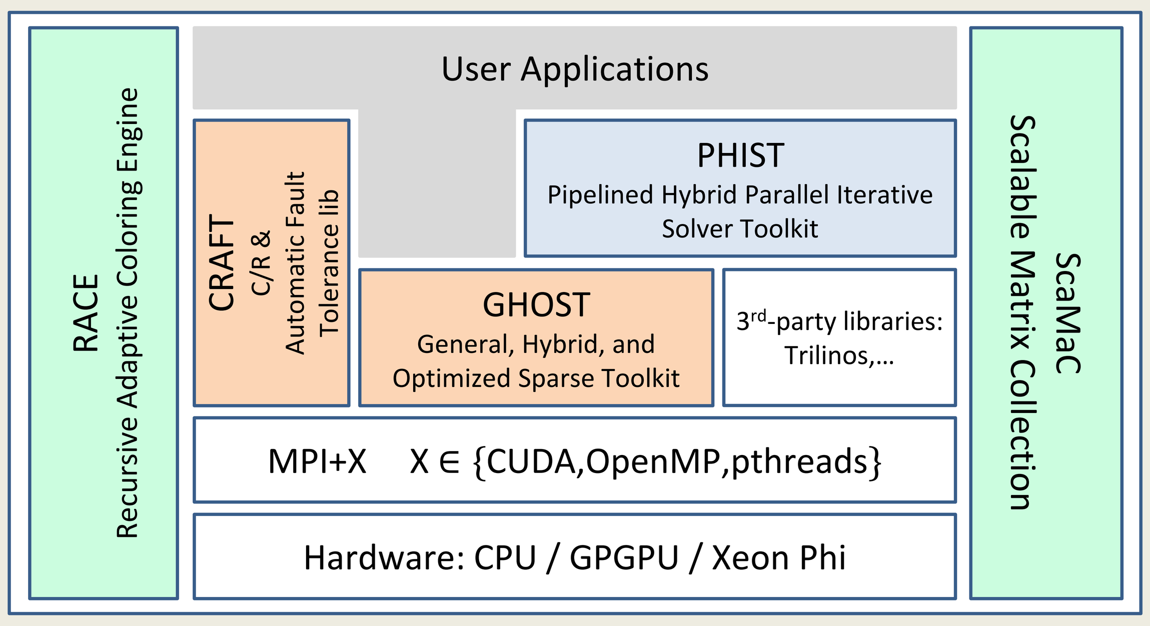
Good approximation to full spectrum (e.g. Density of States)

Need sparse eigenvalue solvers of broad applicability!

Project goals

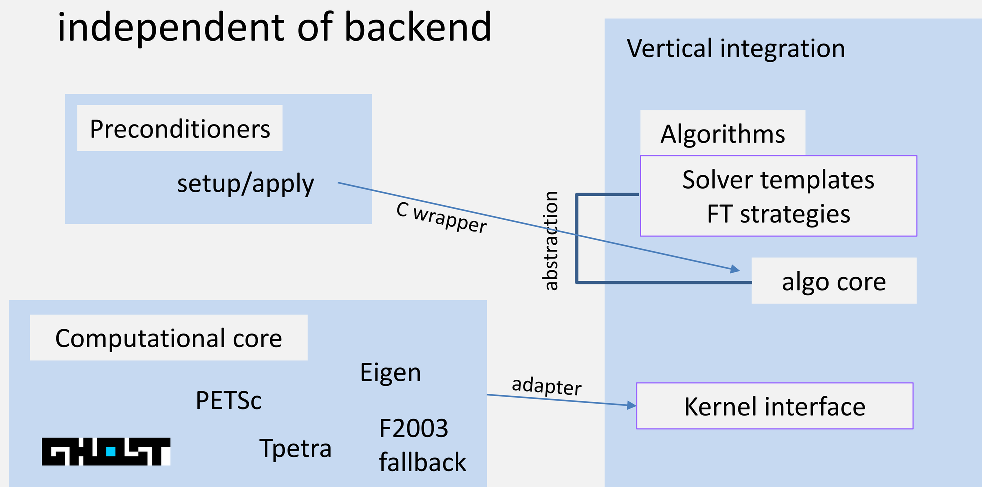
ESSEX-I (2013-2015)	ESSEX-II (2016-2018)
Conservative Quantum Systems	Dissipative Quantum Systems
$A^+ = A$	$A^+ \neq A$
$Ax = \lambda x$	$Ax = \lambda Bx$ $A(\lambda)x = 0$
Code Blueprints	Interoperable Libraries

Usable software



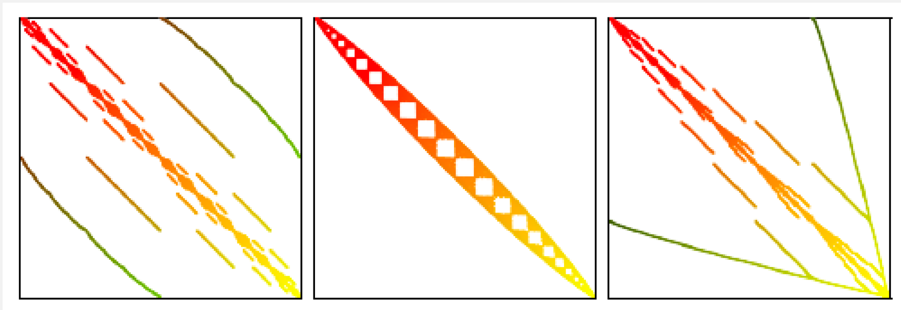
PHIST: Sparse Solver Framework

- General-purpose block Jacobi-Davidson Eigensolver, Krylov methods, preconditioning interface
- C, C++, Fortran 2003, and Python bindings
- Backends: GHOST, Tpetra, PETSc, Eigen, Fortran
- Can use Trilinos solvers Belos and Anasazi, independent of backend



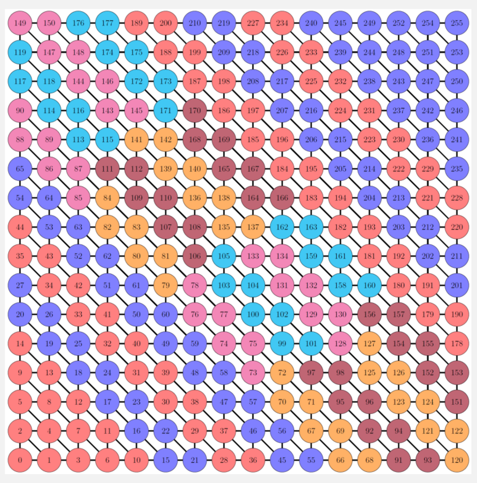
ScaMaC: Scalable Matrix Collection

- Easy generation of large sparse matrices for quantum problems, library & stand-alone
- Fully parallel & scalable generation



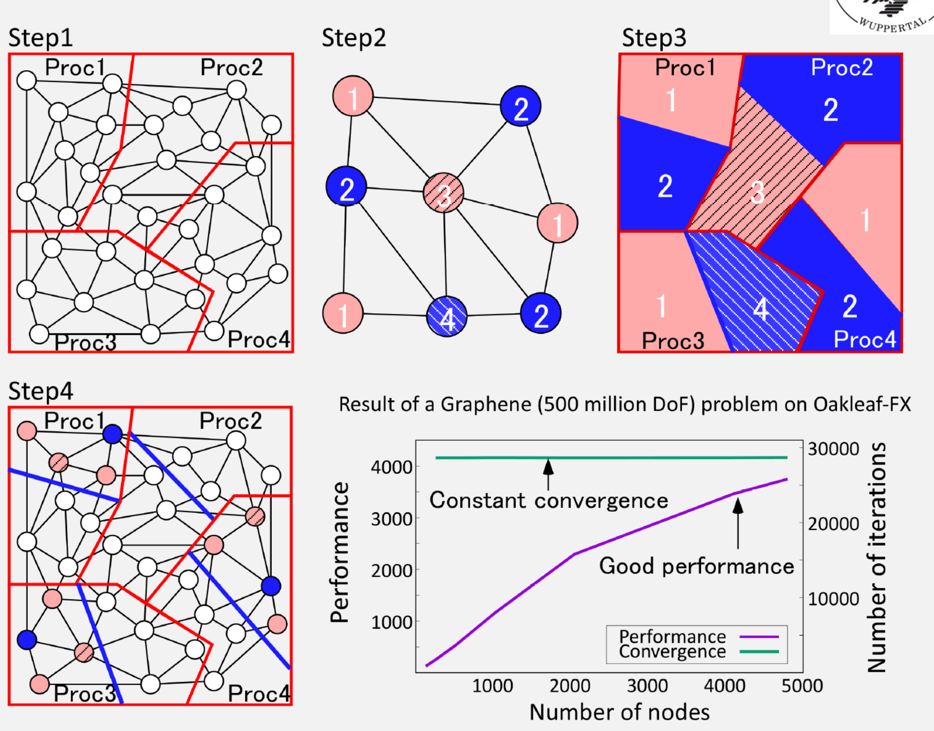
RACE: Recursive Adaptive Coloring Engine

- Block multicoloring for resolving data dependencies in sparse algorithms
- Automatic load balancing
- Cache-friendlier partitioning as compared to standard multicoloring



ppOpen-SOL: Robust ILU Preconditioner For Exascale

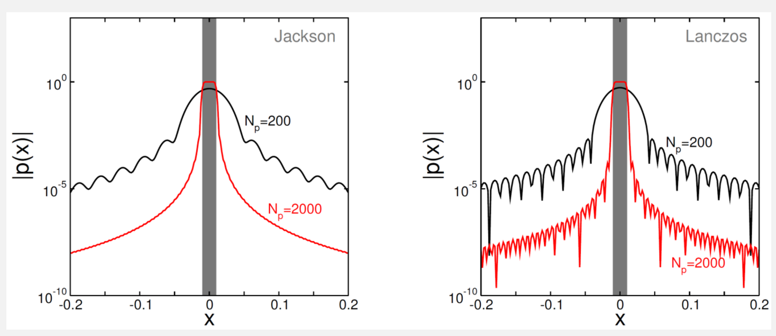
- Block ILU preconditioner with diagonal shifting
- Hierarchical parallelization of multicoloring



Inner Eigenvalues

Chebyshev Filter Diagonalization

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

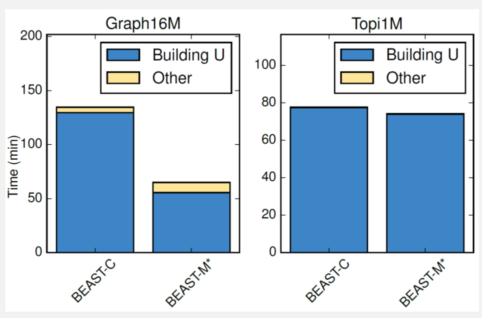


High-degree polynomials required for good filter functions

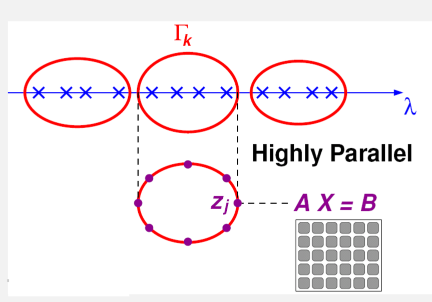
BEAST: Framework for interior eigenproblems

Solver alternatives:

- BEAST-M: initial outer iterations → perform.
- BEAST-C: later outer iterations → accuracy
- Adaptive accuracy support (FP32 → FP64)

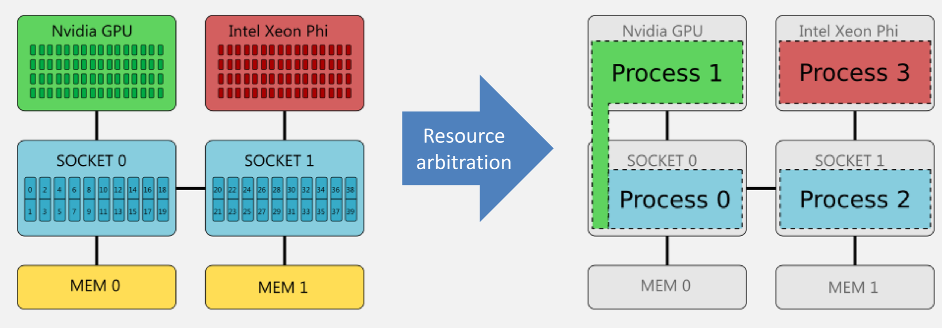


Nonlinear EV Problems $A(\lambda)x = 0$

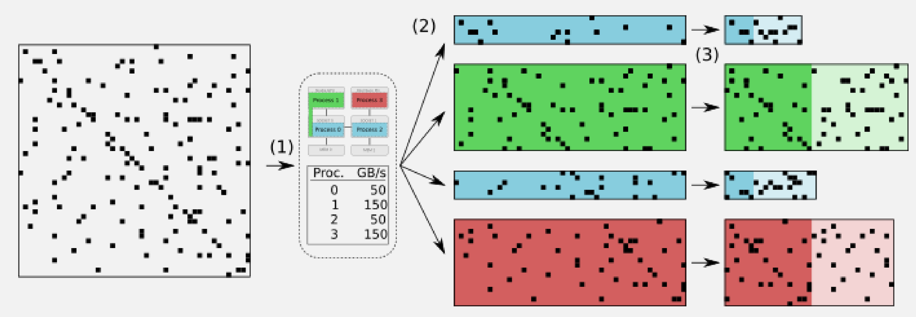


GHOST: General, Hybrid, Optimized Sparse Toolkit

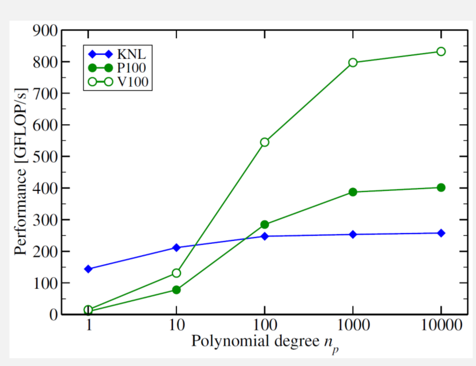
- Sparse building blocks (spM[M]VM, simple algorithms, blueprints) and tasking library
- MPI+X, X ∈ {CUDA,OpenMP,pthreads}
- Fully heterogeneous parallelism (CPU, GPGPU, Phi)



Heterogeneous compute node — Mapping of MPI processes



Data-parallel and hardware-aware heterogeneous work distribution

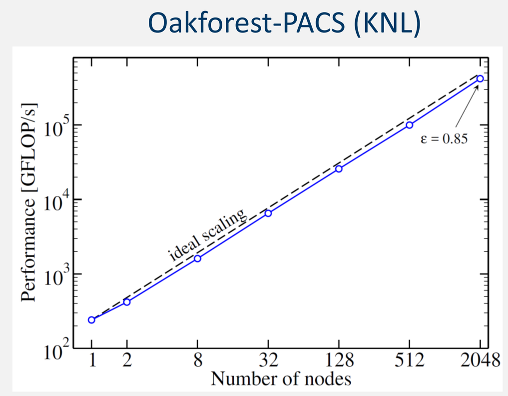


← Polynomial filter performance vs. polynomial degree on KNL, P100, and V100 for a topological insulator matrix

Filter diagonalization of a topological insulator matrix of size up to 4×10^9, computing 100 inner eigenvalues (weak scaling)

↓

Oakforest-PACS (KNL)



Piz Daint (P100)