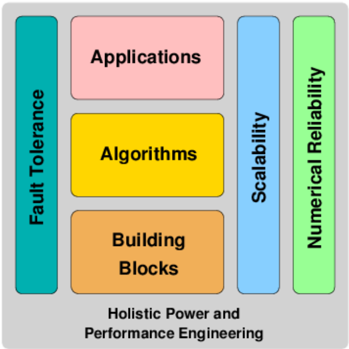
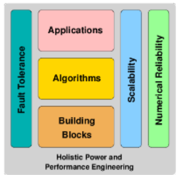
Equipping Sparse Solvers for Exascale (ESSEX / ESSEX II)



Gerhard Wellein	Computer Science, University Erlangen
Bruno Lang	Applied Computer Science, University Wuppertal
Achim Basermann	**Simulation & SW Technology, German Aerospace Center**
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

ESSEX: 2013 – 2015
ESSEX II: 2016 – 2018



- Motivation

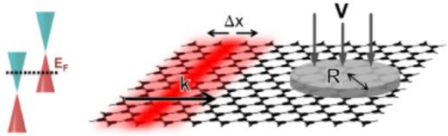
- Software:
  - Interoperability, portability & performance

- Multicoloring and ILU Preconditioning

- Scaling Results: Eigenvalue Computations

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....

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

"Few" (1,...,100s) of eigenpairs

"**Bulk**" (100s,...,1000s) eigenpairs

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

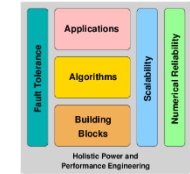
Good approximation to full spectrum (e.g. Density of States)

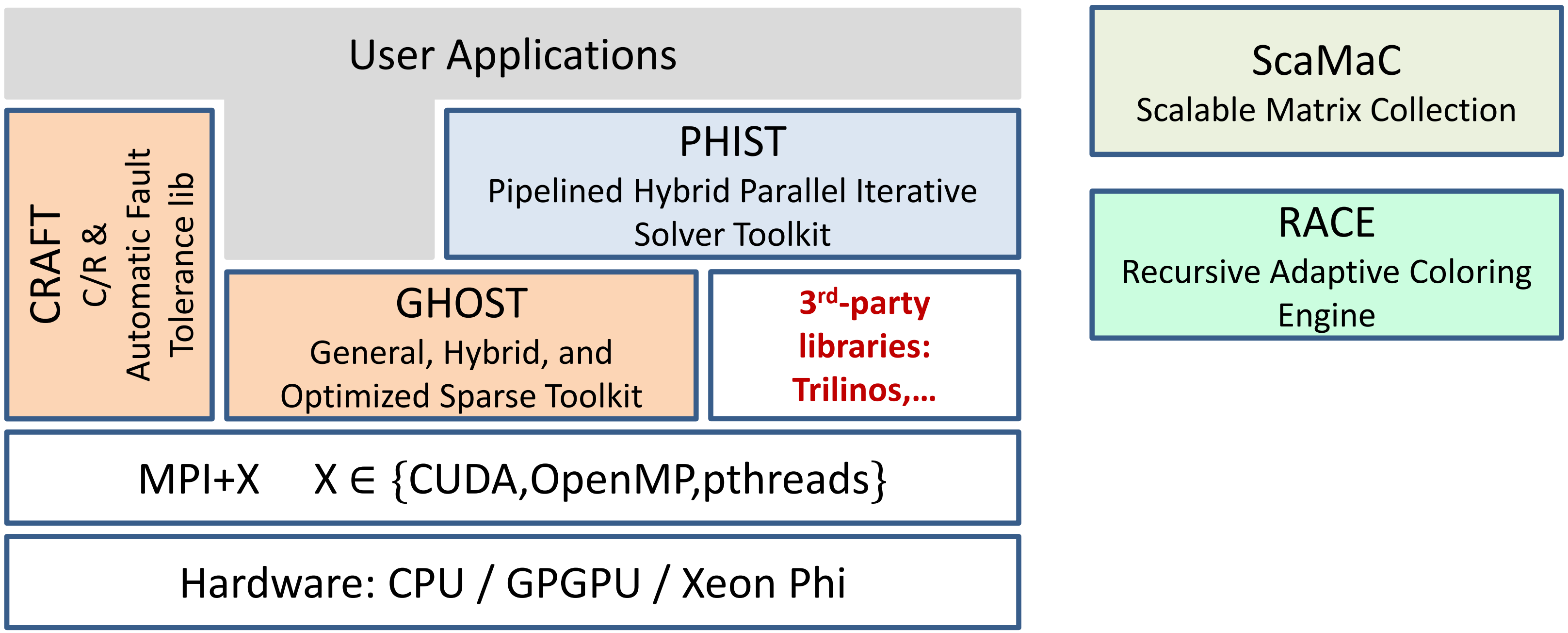
→ Sparse eigenvalue solvers of broad applicability

Software: Interoperability portability & performance

Kernel library (GHOST) and solver framework (PHIST)

ESSEX-II: Software Packages

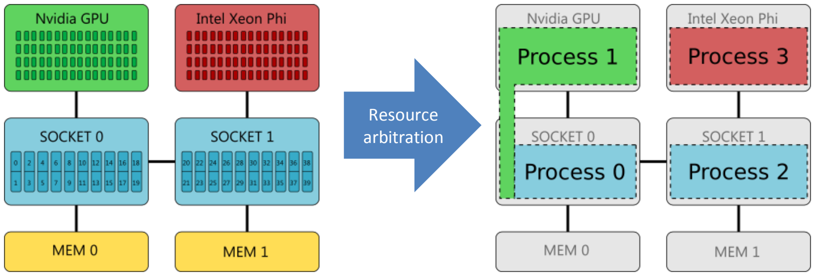




Links to open source repositories at **`https://blogs.fau.de/essex/code`**

GHOST library

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



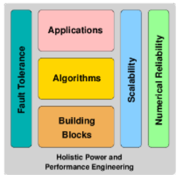
- **Algorithm specific kernels**: SIMD Intrinsics (KNL) and CUDA (NVIDIA)
  → 2x – 5x speed-up vs. Optimized general building block libraries

- **Tall & skinny matrix-matrix kernels** (block orthogonalization)
  → 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

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

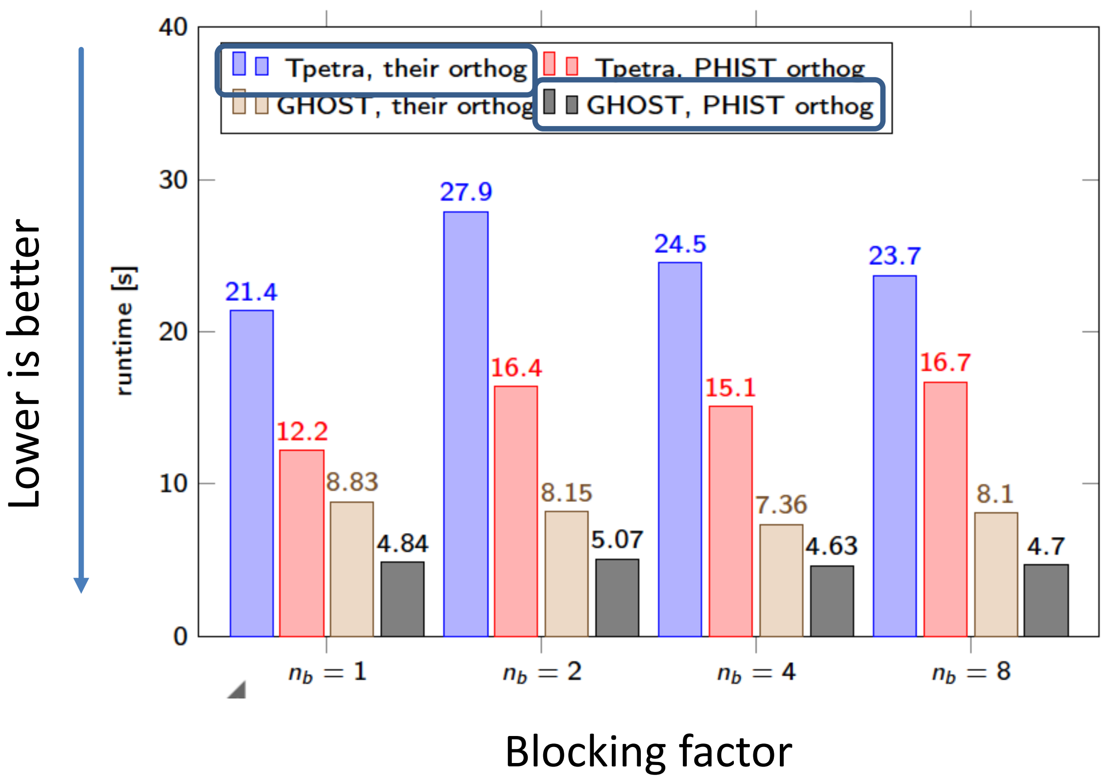
Getting PHIST and GHOST

- https://bitbucket.org/essex/[ghost,phist]
- Cmake build system
- Availale via Spack
- https://github.com/spack/spack/
- PHIST will join **Extreme-Scale Development Kit**, 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)



- **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/

Multicoloring and ILU Preconditoning

RACE and ILU preconditioning

Recursive algebraic coloring engine (RACE)

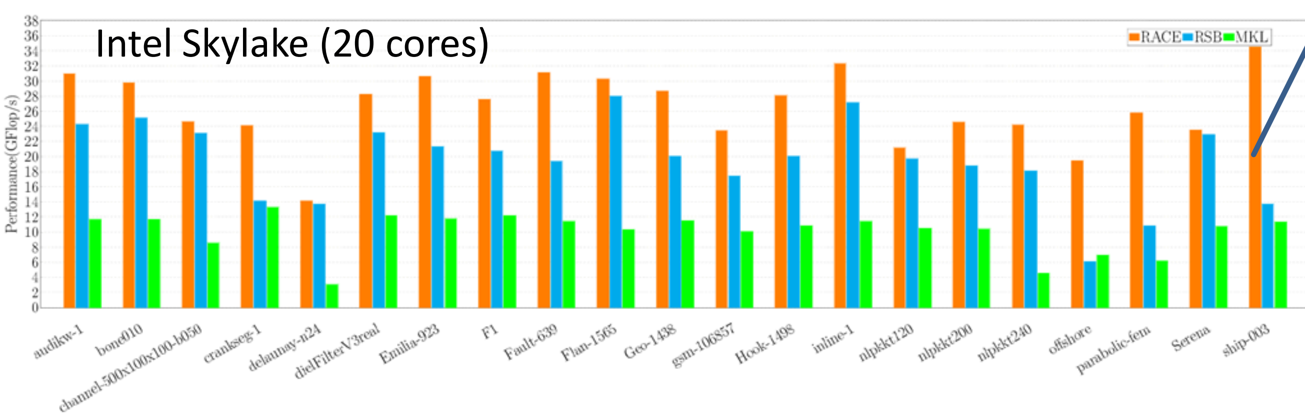
Graph coloring: RACE uses recursive BFS level based method for “distance-k coloring” of symmetric matrices

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)

RACE



Compare with
- Intel MKL
- RSB (data format)
- Multicoloring

Recursive algebraic coloring engine (RACE)

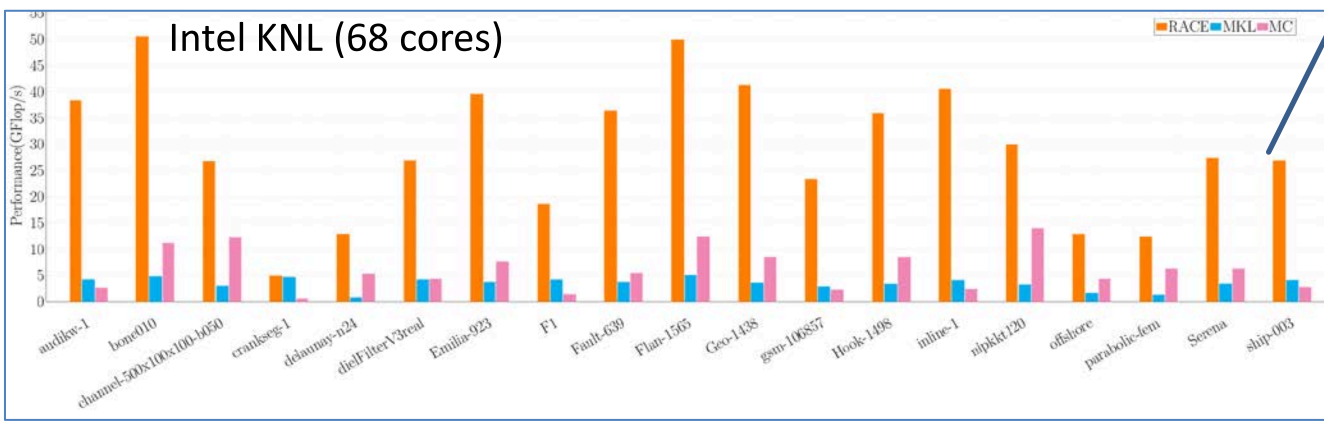
Graph coloring: RACE uses recursive BFS level based method for “distance-k coloring” of symmetric matrices

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)

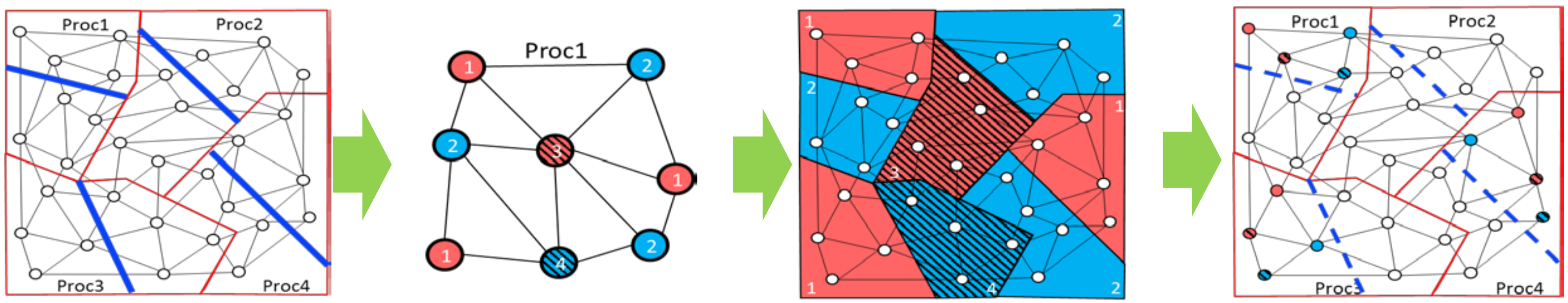
RACE



Compare with
- Intel MKL
- RSB (data format)
- Multicoloring

Robustness & Scalability of ILU preconditioning

- Hierarchical parallelization of multi-colorings for ILU precond.



- High precision Block ILU preconditioning: Achieved almost constant iterations and good scalability with a graphene model (500 million DoF)

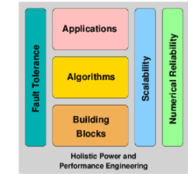
Tokyo Univ.: Masatoshi Kawai (now Riken) , Kengo Nakajima et al.

- Apply algebraic block multi-coloring to ILU preconditioning: 2.5x – 3.5x speed-up vs multicoloring

Hokkaido Univ.: Takeshi Iwashita et al.

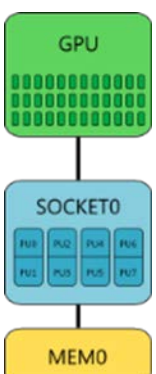
Scaling Results: Eigenvalue Computations

Scalability on Oakforest-PACS
since 6 / 2018 number 12 of TOP 500 The List.



Cores: Memory: Processor: Interconnect:	556,104 919,296 GB Intel Xeon Phi 7250 68C 1.4GHz (KNL) Intel Omni-Path
Linpack Performance (Rmax) Theoretical Peak (Rpeak)	13.554 PFlop/s 24.913 PFlop/s
Nmax HPCG [TFlop/s]	9,938,880 385.479

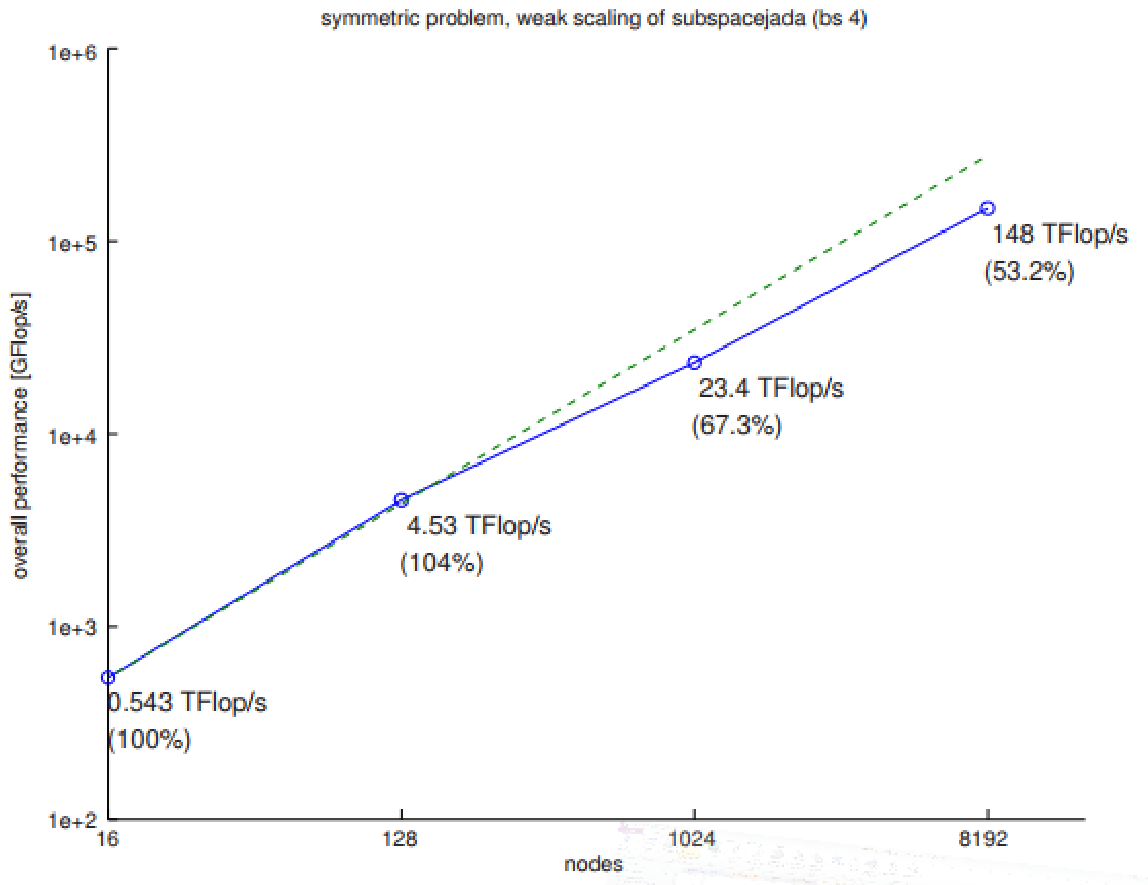
CRAY XC30 – PizDaint



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

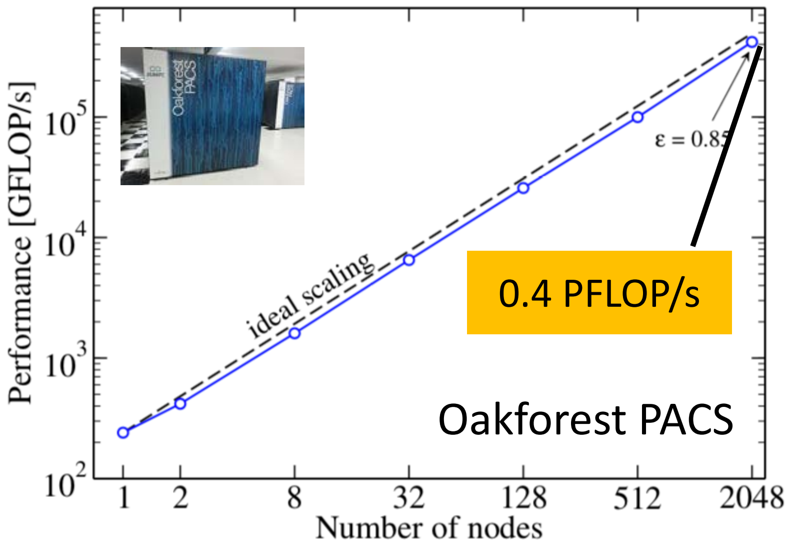
Weak scaling: Jacobi-Davidson Method

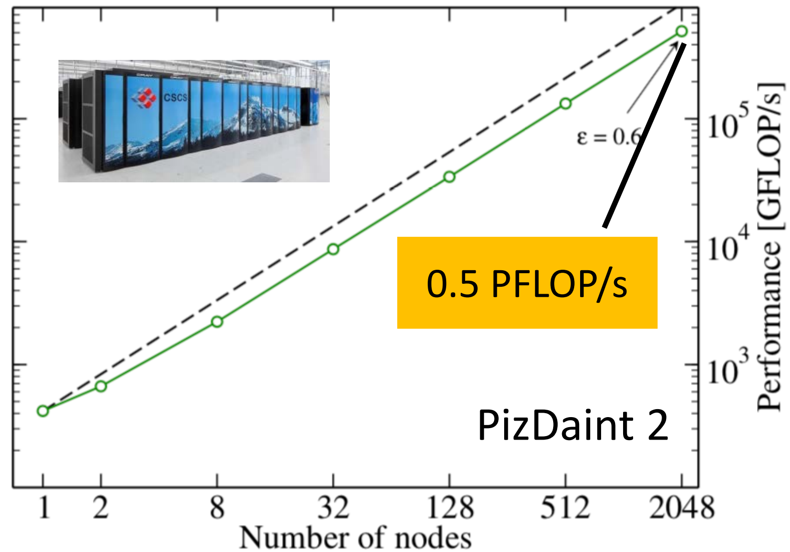
- Up to 0.5M cores
- Percentage indicates the parallel efficiency compared to the first measurement (smallest node count).
- Symmetric PDE problem with the largest matrix size N = 40 963,
- target eigenpairs near 0 ,
- The best performance was obtained with a block size of 4.



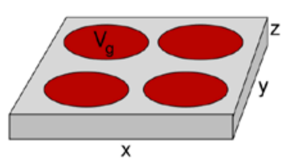
Large scale performance – weak scaling

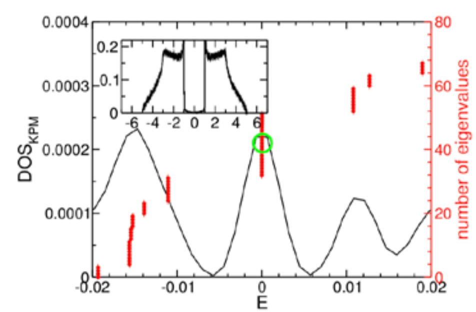
Computing 100 inner eigenvalues on matrices up to $n = 4 \times 10^9$

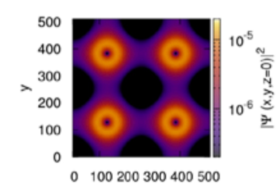




Typical Application[1]:
Topological Insulator

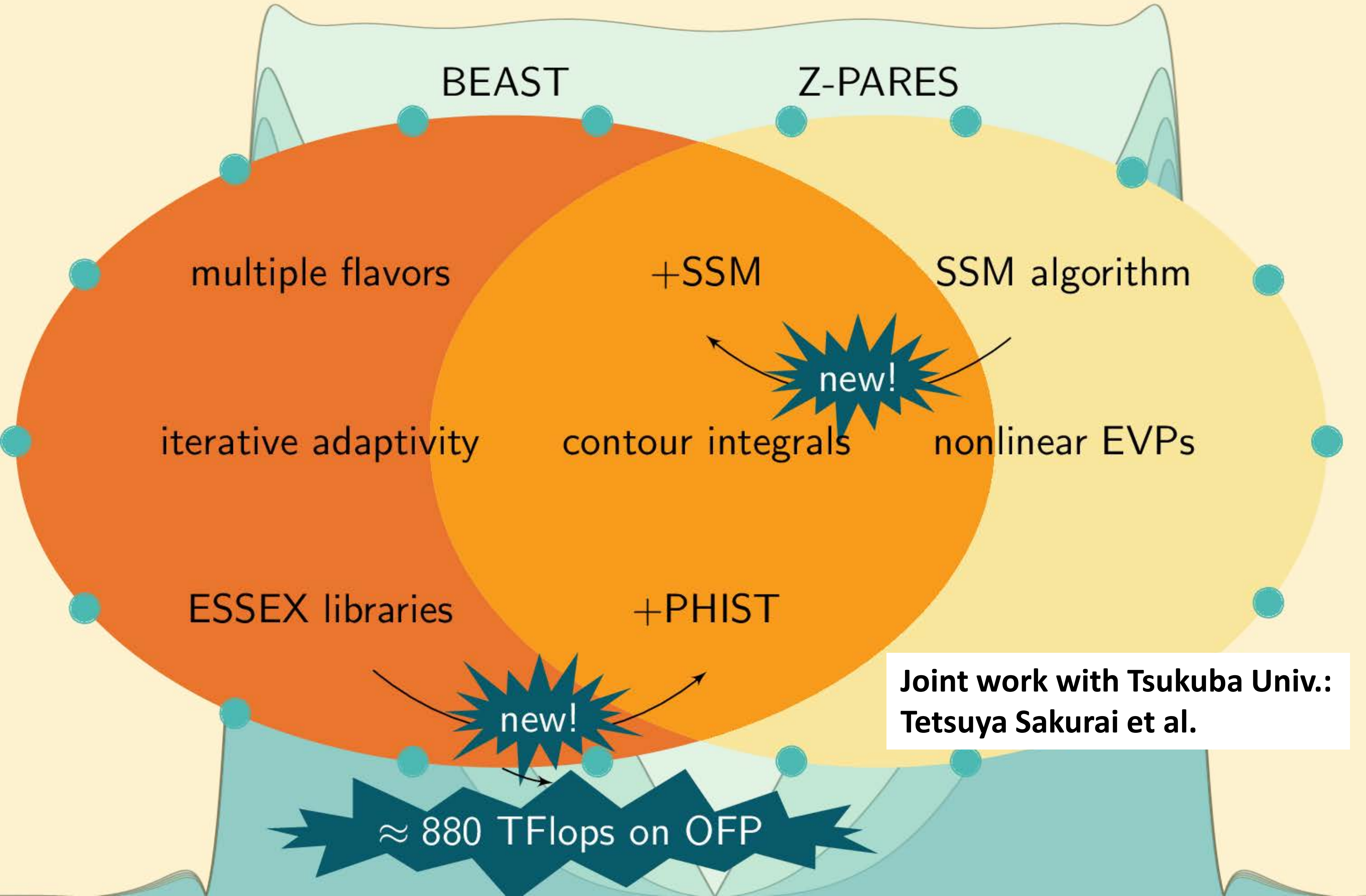






[1] Pieper, A., et al. Journal of Computational Physics 325, 226–243 (2016)

BEAST and Z-PARES: shared tools for large EVPs



Joint work with Tsukuba Univ.: Tetsuya Sakurai et al.

Visit our homepage: https://blogs.fau.de/essex/

THANK YOU!