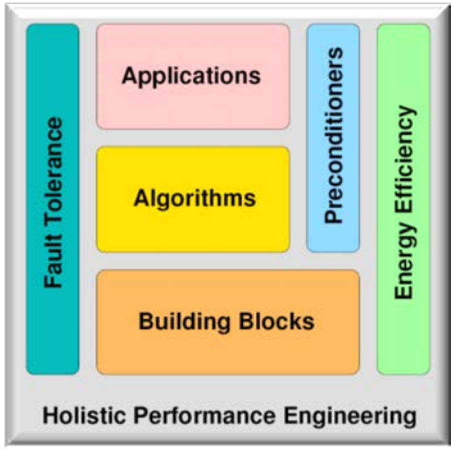
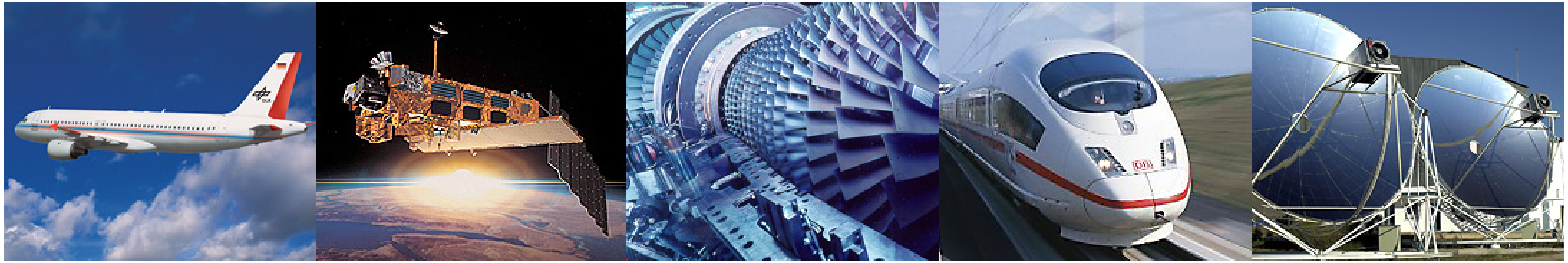
DFG Projekt ESSEX



Equipping Sparse Solvers for Exascale – A Survey of the DFG Project ESSEX

Achim Basermann
German Aerospace Center (DLR)
Simulation and Software Technology
Linder Höhe, Cologne, Germany

DLR
German Aerospace Center



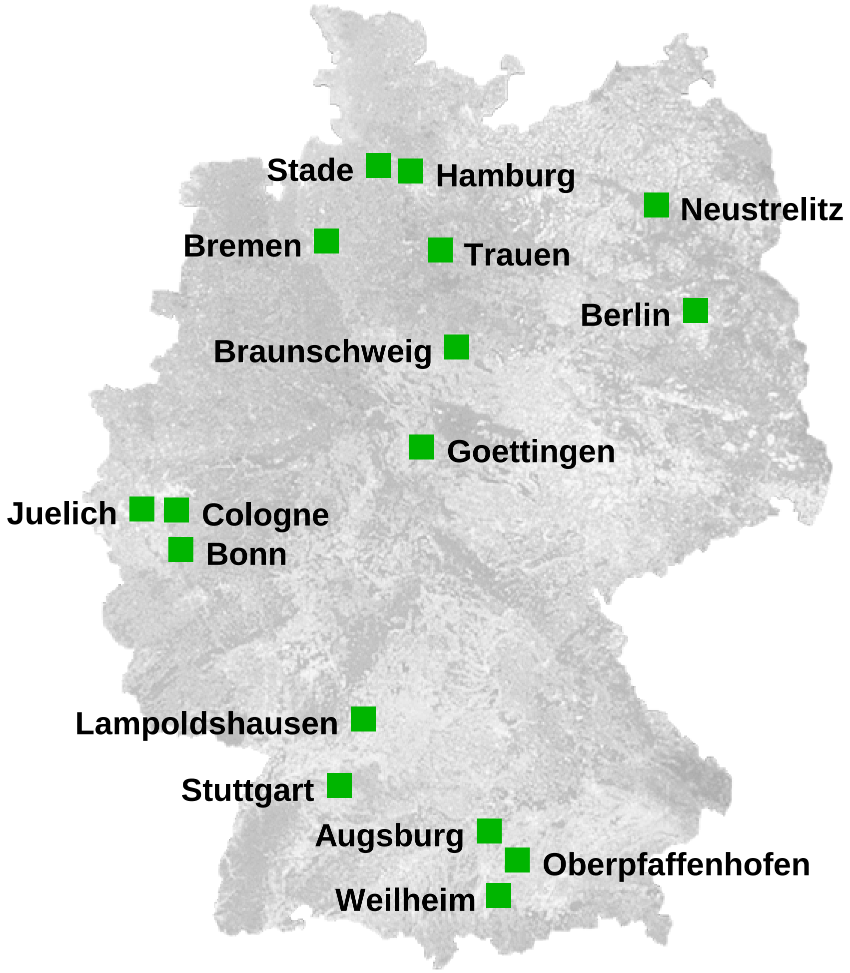
- Research Institution
- Space Agency
- Project Management Agency

DLR Locations and Employees

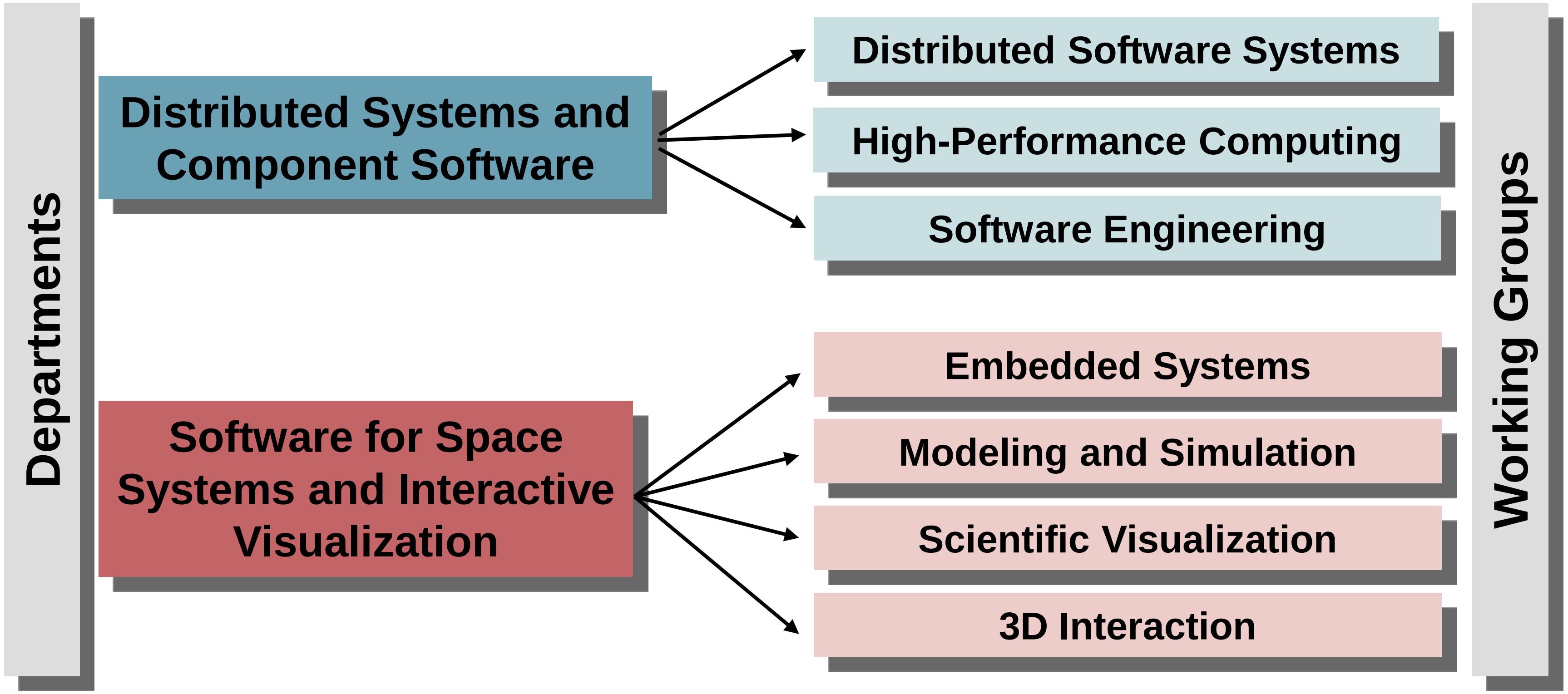
Approx. 8000 employees across 33 institutes and facilities at

■ 16 sites.

Offices in Brussels, Paris, Tokyo and Washington.



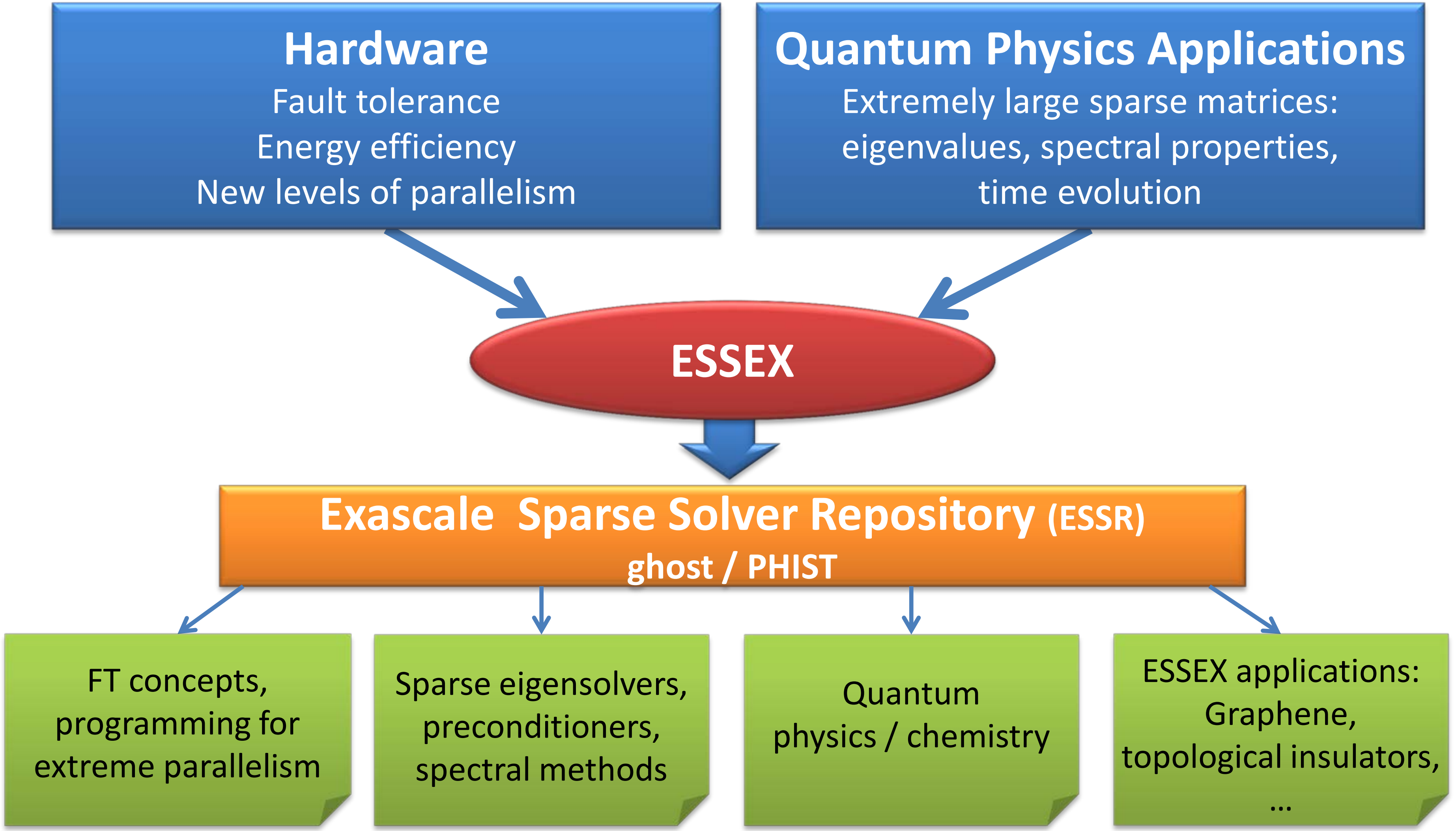
DLR Institute Simulation and Software Technology
Scientific Themes and Working Groups



Survey

- ESSEX motivation
- The ESSEX software infrastructure
- Holistic view: application, algorithm and performance
- Algorithmic developments: JADA, FEAST, CARP-CG
- Application results
- Conclusions
- The Future: ESSEX II

ESSEX Motivation: Requirements for Exascale

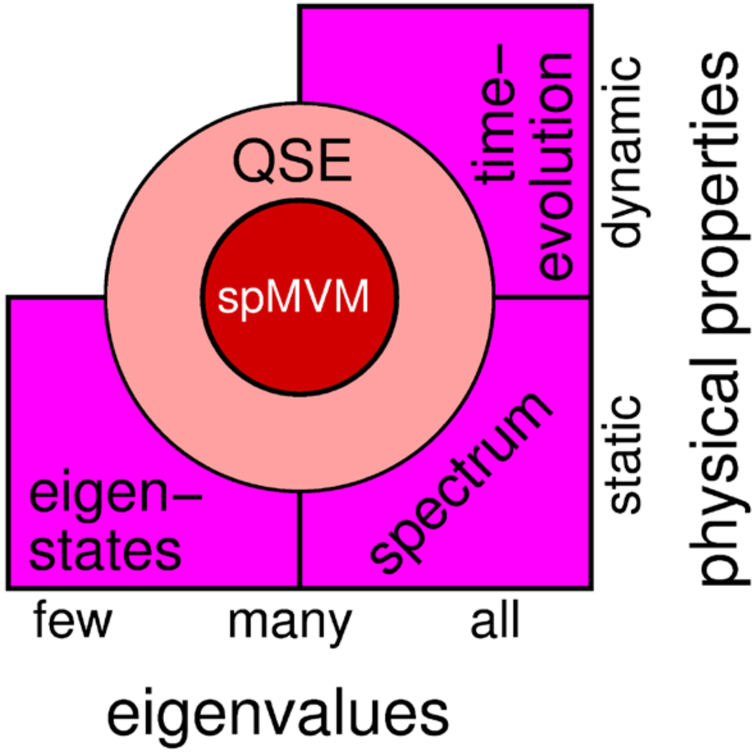


ESSEX: Physical Motivation and Sparse Eigenvalue problem

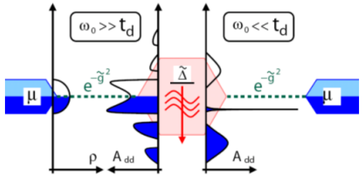
Solve large sparse eigenvalue problem

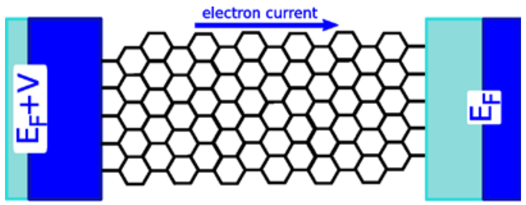
$$\boldsymbol{H}\,\boldsymbol{x} = \boldsymbol{\lambda}\,\boldsymbol{x}$$

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

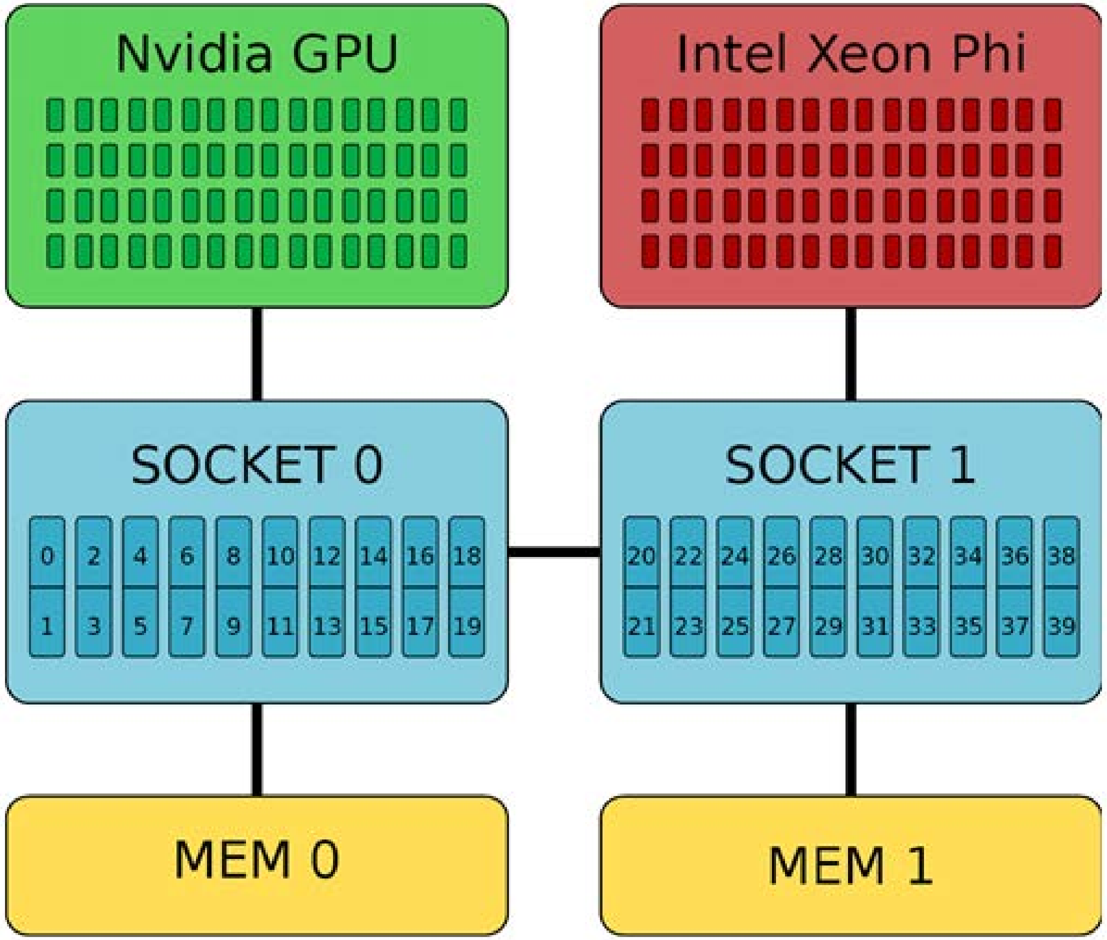


(λ_i, x_i)





ESSEX Motivation: Programming Heterogeneous HPC Systems



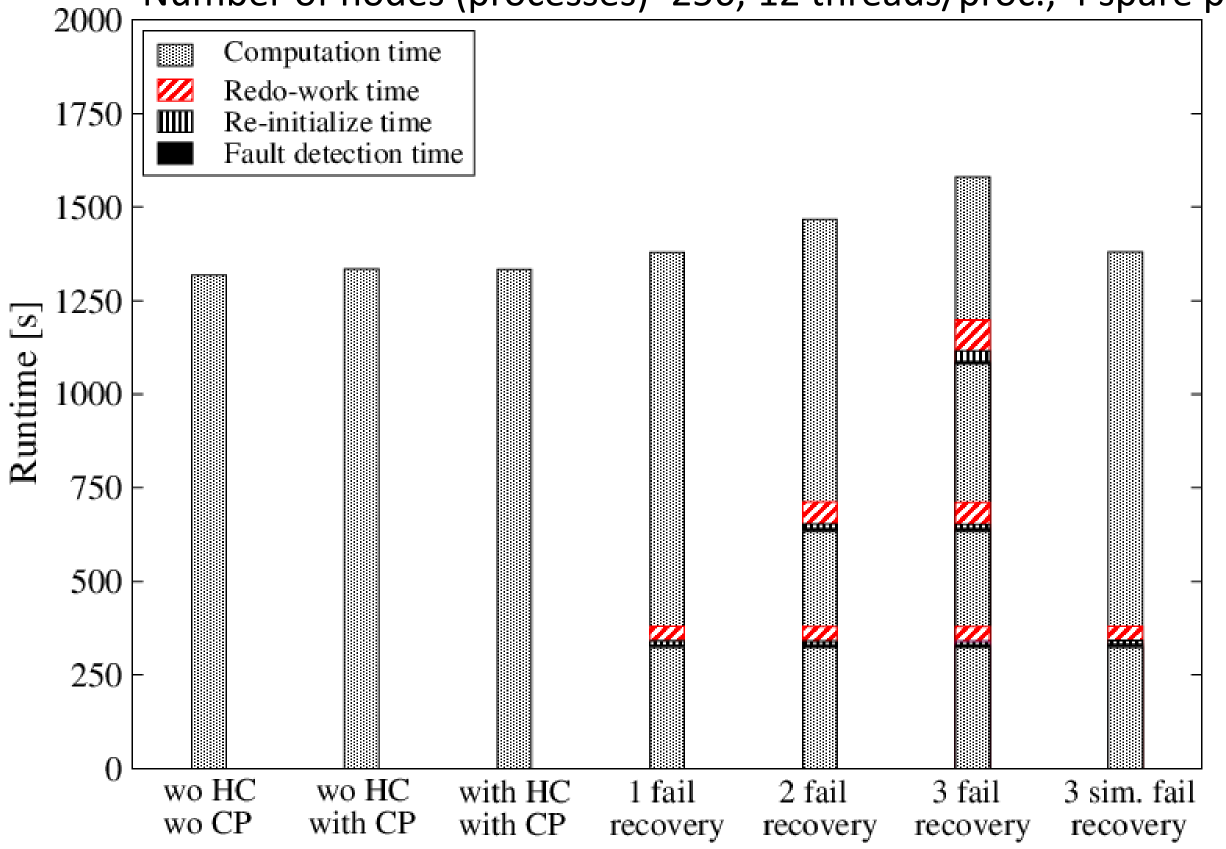
- Flat MPI + off-loading
- Runtime (e.g. MAGMA, OmpSs)
  - Dynamic scheduling of small tasks → good load balancing
- Kokkos (Trilinos)
  - High level of abstraction (C++11)
- **MPI+X strategy in ESSEX**
  - **X: OpenMP, CUDA, SIMD Intrinsics, e.g. AVX**
  - **Tasking for bigger asynchronous functions → functional parallelism**
  - **Experts implement the kernels required.**

ESSEX Motivation: Application Driven Fault Tolerance (FT)

- Application **asynchronously** writes checkpoints **(CP)**
  - to a local disk
  - to memory of a neighbor node
- Dedicated process performs health checks **(HC)** of all nodes, GASPI/GPI used rather than MPI
- If a node fails:
  - Pool of substitute processes
  - Rollback to last checkpoint

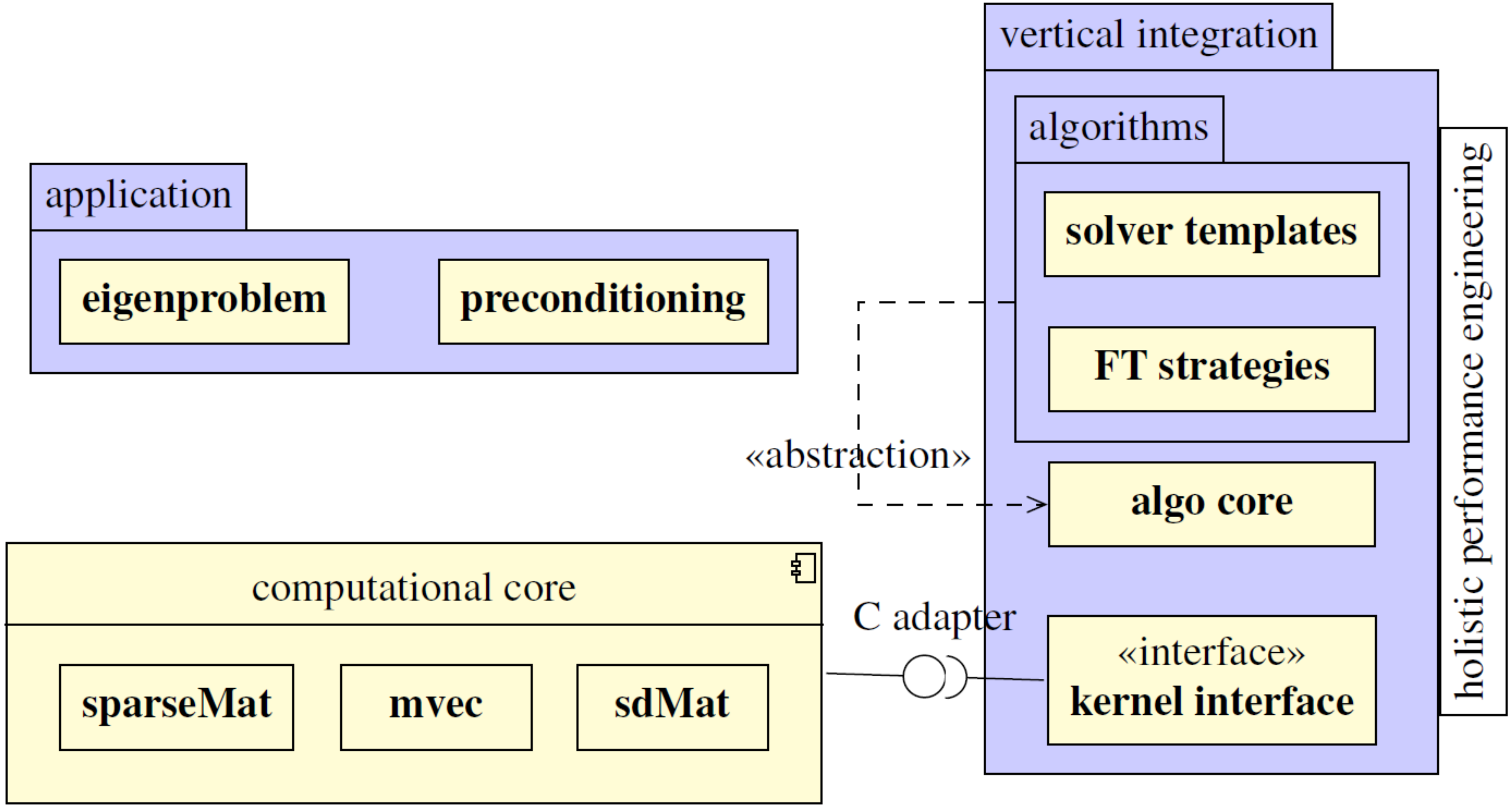
Lanczos application benchmark

Number of nodes (processes)=256, 12 threads/proc., 4 spare processes

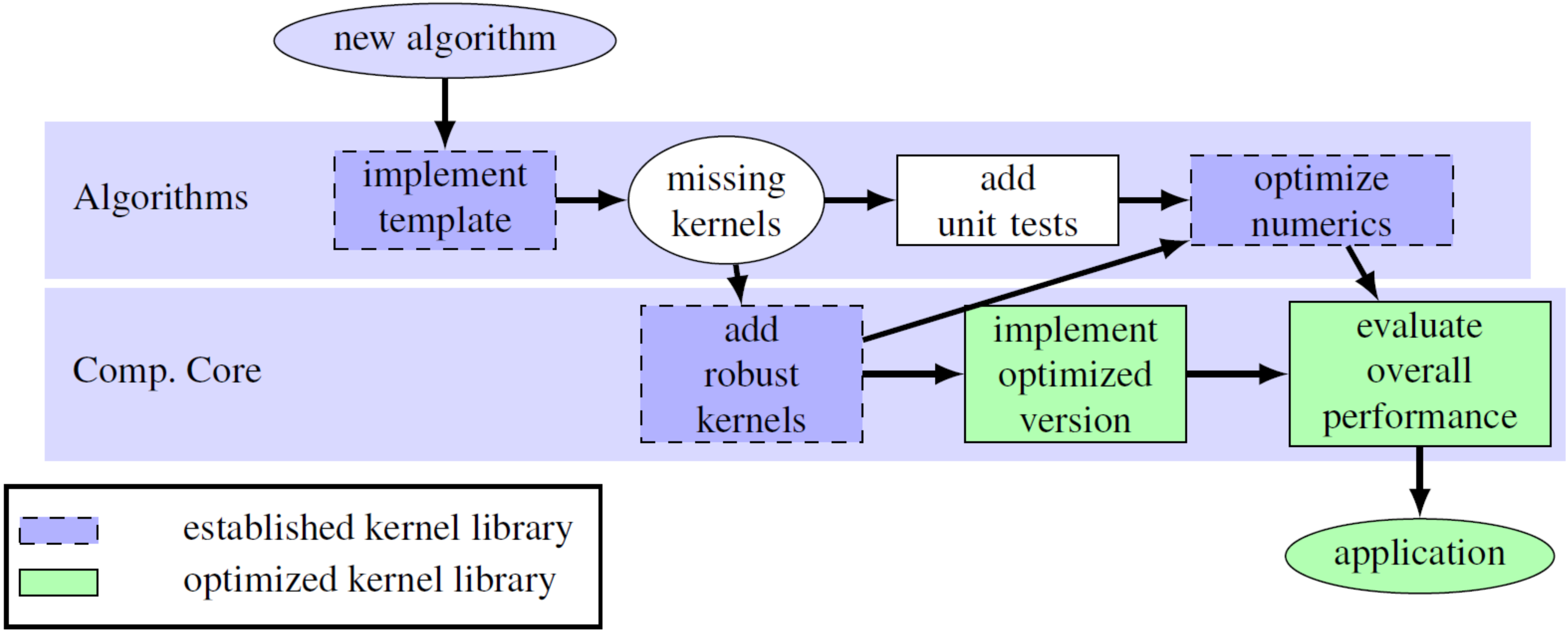


Overhead for recovery ca 18 s
+ computations to be repeated

The ESSEX Software Infrastructure



The ESSEX Software Infrastructure: Test-Driven Algorithm Development



Optimized ESSEX Kernel Library

General, Hybrid, and Optimized Sparse Toolkit

- MPI + OpenMP + SIMD + CUDA
- Sparse matrix-(block-)vector multiplication
- Dense block-vector operations
- Task-queue for functional parallelism
- Asynchronous checkpoint-restart

Status: beta version, suitable for experienced HPC C programmers

http://bitbucket.org/essex/ghost

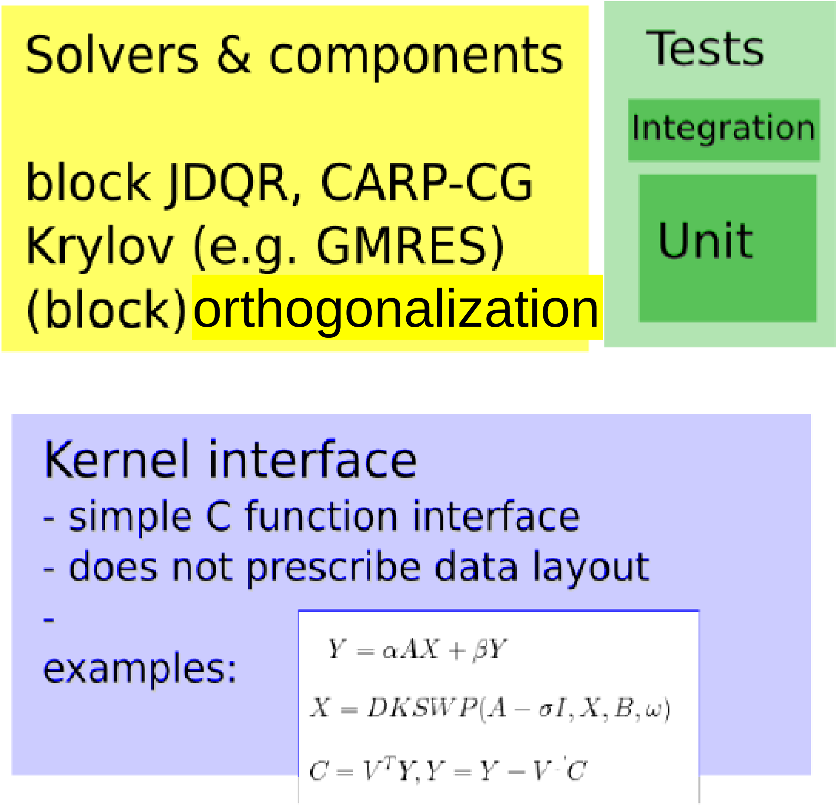
BSD License

The Iterative Solver Library PHIST

PHIST **Pipelined Hybrid parallel Iterative Solver Toolkit**

- Iterative solvers for sparse matrices
  - Eigenproblems: Jacobi-Davidson, FEAST
  - Systems of linear equations: GMRES, MINRES, CARP-CG
- Provides some abstraction from data layout, process management, tasking etc.
- Adapts algorithms to use block operations
- Implements asynchronous and fault-tolerant solvers
- Simple functional interface (C, Fortran, Python)
- Systematically tests kernel libraries for correctness and performance
- Various possibilities for integration into applications

Status: beta version with extensive test framework
http://bitbucket.org/essex/phist
BSD License



Integration of PHIST into Applications

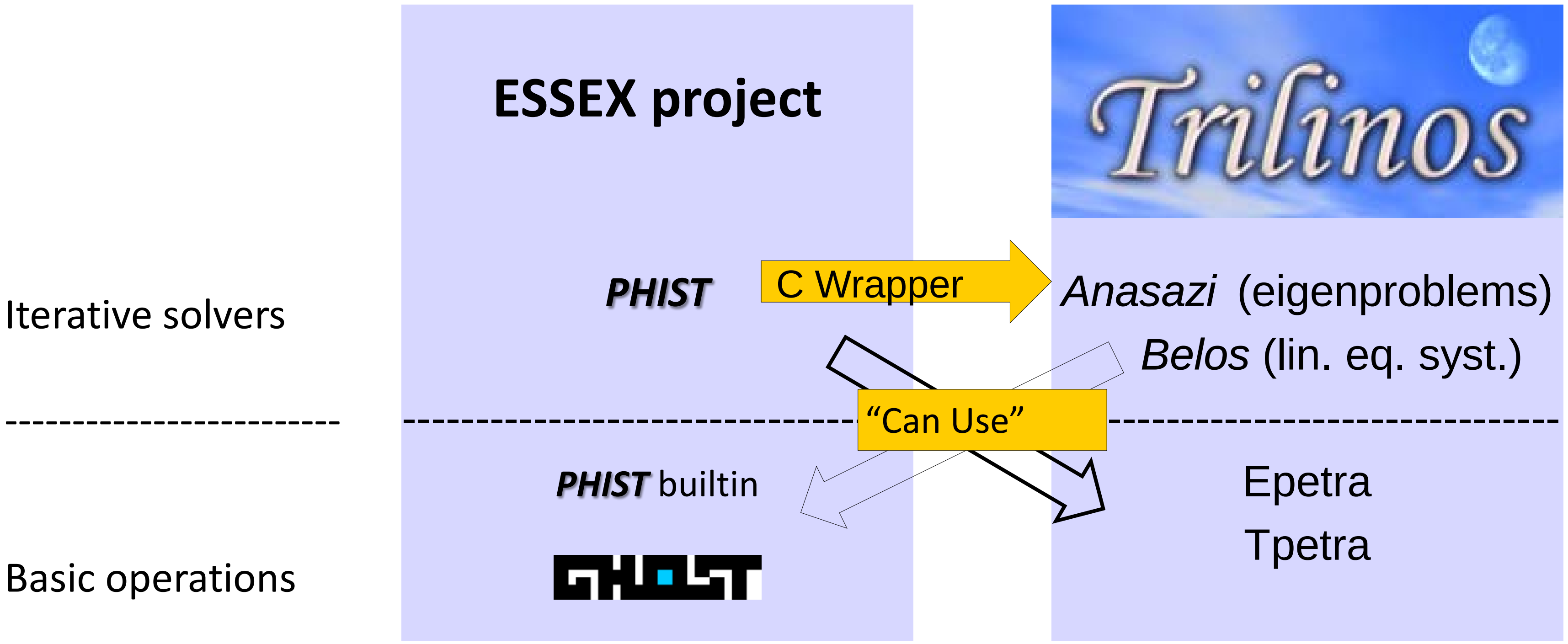
Selection of kernel library

Required flexibility: low → high

PHIST (GHOST)	*PHIST* builtin	Trilinos	Own data structures
No easy access to matrix elements	Only CPU F‘03+OpenMP CRS format	Various arch. Large C++ code base	Adapter ca 1000 lines of code

Hardware awareness: high ← low

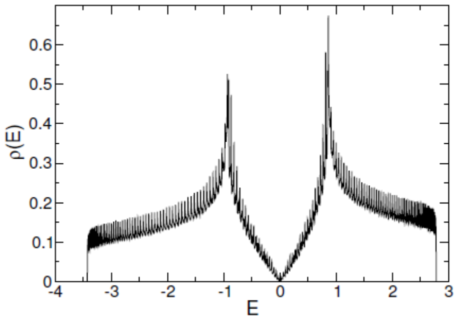
Interoperability of PHIST and Trilinos

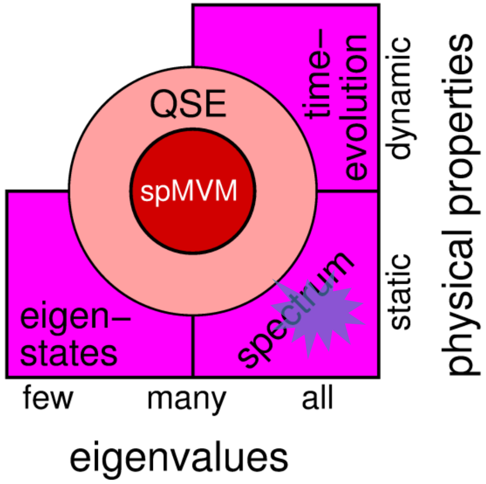


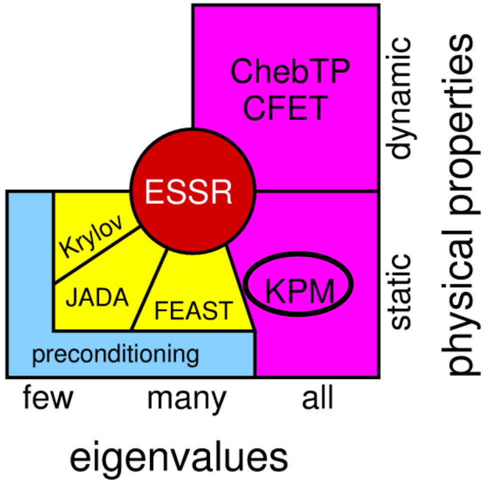
Application, Algorithm and Performance: Kernel Polynomial Method (KPM) – A Holistic View

- 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**
  $|v\rangle \leftarrow |\mathrm{rand}()\rangle$
  Initialization steps and computation of η_0, η_1
  for $m = 1$ to $M/2$ **do**
    $\mathrm{swap}(|w\rangle, |v\rangle)$
    $|u\rangle \leftarrow H|v\rangle$ ▷ 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**
    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**
    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!

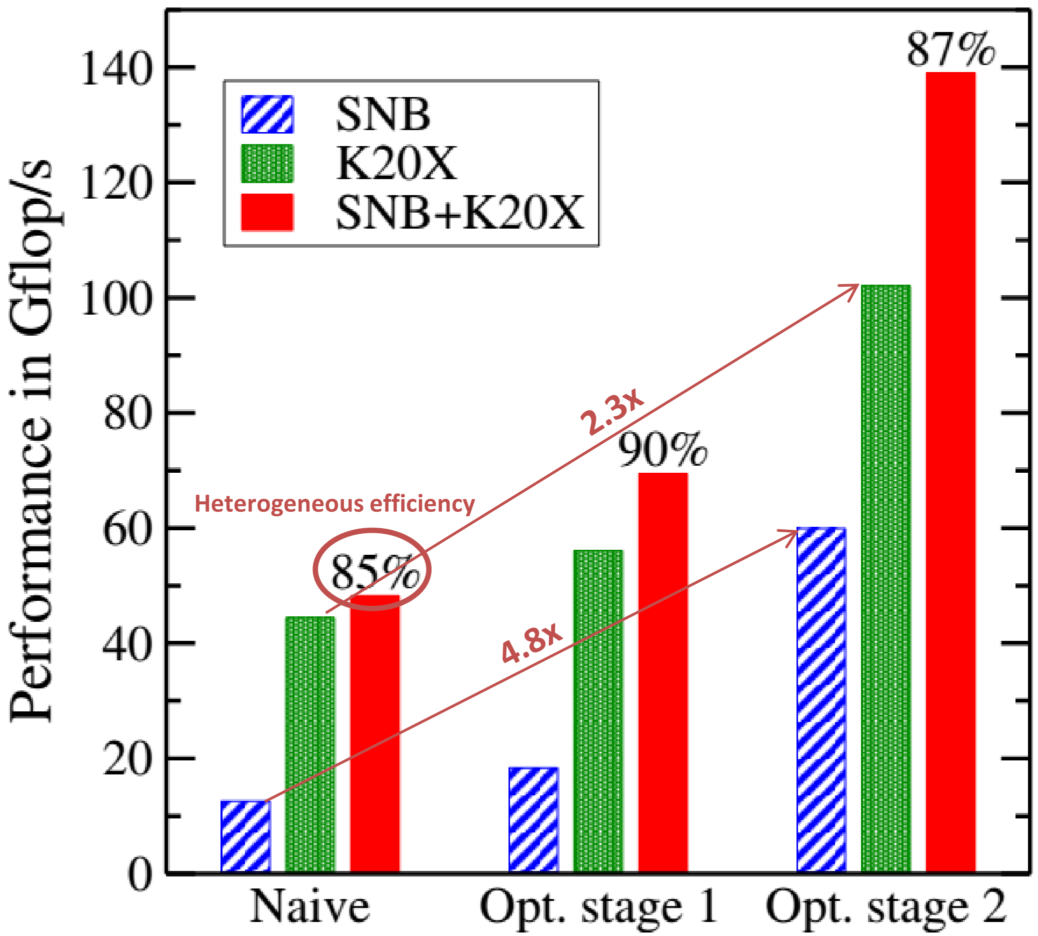
Basic algorithm – Compute Cheyshev polynomials/moments:

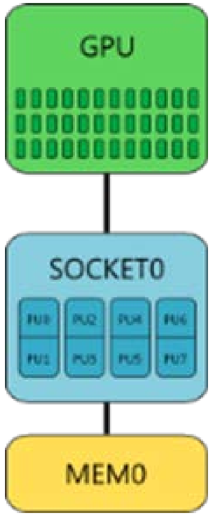
for $r = 0$ to $R-1$ **do**
  $|v\rangle \leftarrow |\text{rand}()\rangle$
  Initialization steps and computation of η_0, η_1
  for $m = 1$ to $M/2$ **do**
    $\text{swap}(|w\rangle, |v\rangle)$
    $|w\rangle = 2a(H - b\mathbb{1})|v\rangle - |w\rangle$ &
      $\eta_{2m} = \langle v|v\rangle$ &
  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



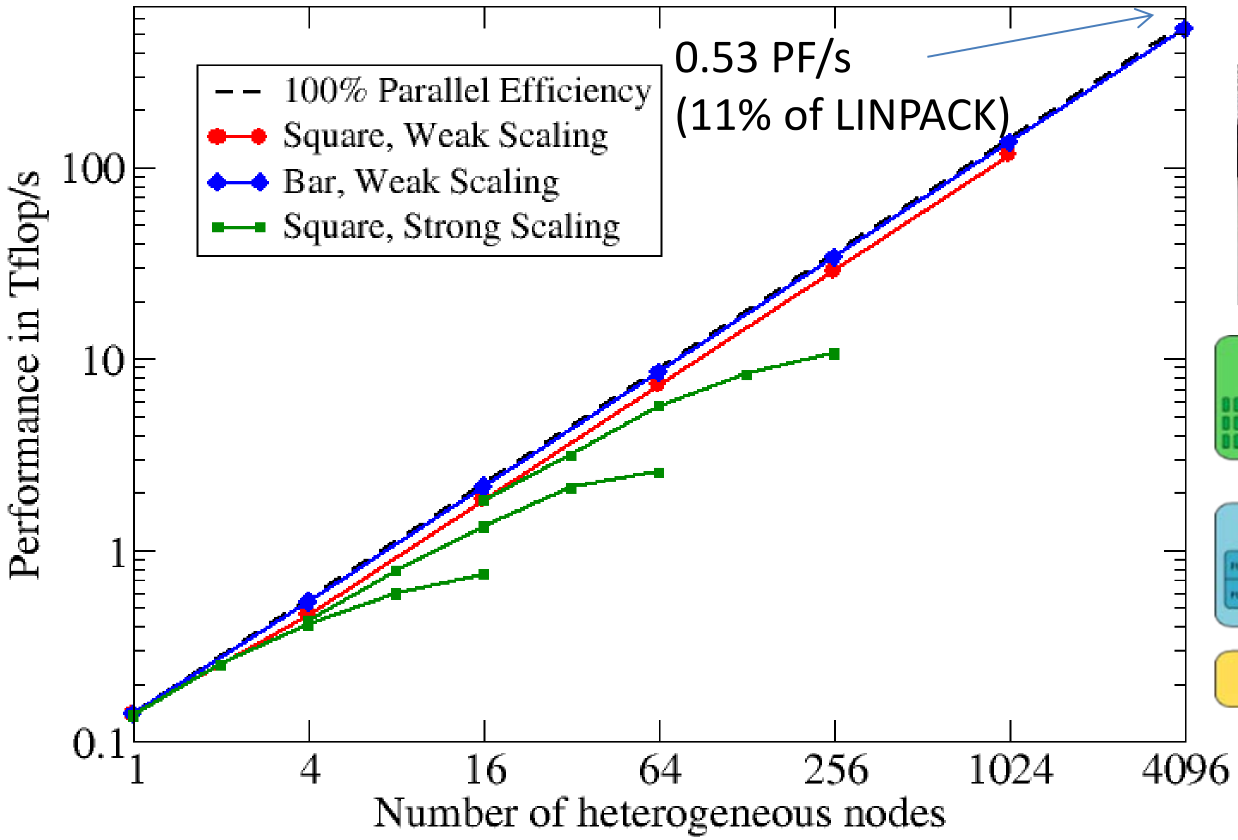


NVIDIDA K20X

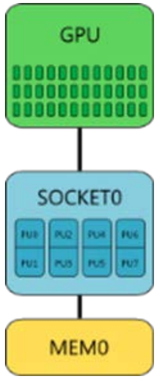
Intel
Xeon E5-2670 (SNB)

- 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

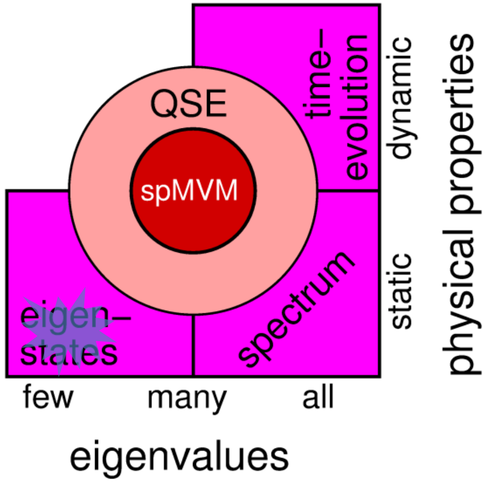
Performance Engineering of the Kernel Polynomial Method on Large-Scale CPU-GPU Systems
M. Kreutzer, A. Pieper, G. Hager, A. Alvermann, G. Wellein and H. Fehske, IEEE IPDPS 2015

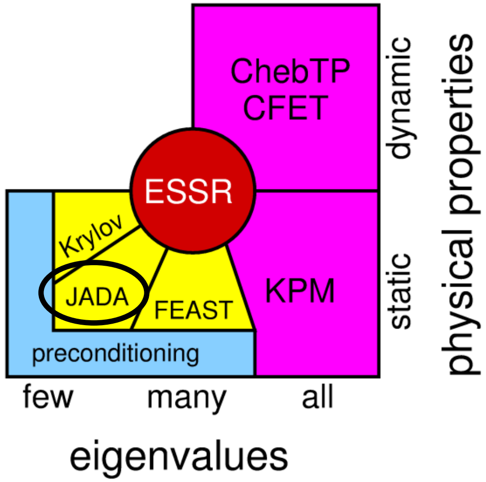
*Thanks to CSCS/T. Schulthess for granting access and compute time

Algorithmic Developments: Blocked Jacobi-Davidson (JADA) Method

Compute l **extreme eigenvalues/-vectors** $\{(\lambda_i, v_i), i = 1, \ldots, l\}$
of sparse matrix A:

$$A\, v_i = \lambda_i v_i$$





Algorithmic Developments: Blocked JADA – exploit benefit of block spMVM

Blocked JADA method: Solve n_b correction equations at the same time.
Basic BLOCKED JADA operator becomes (j=1,…,n_b):

Scalar

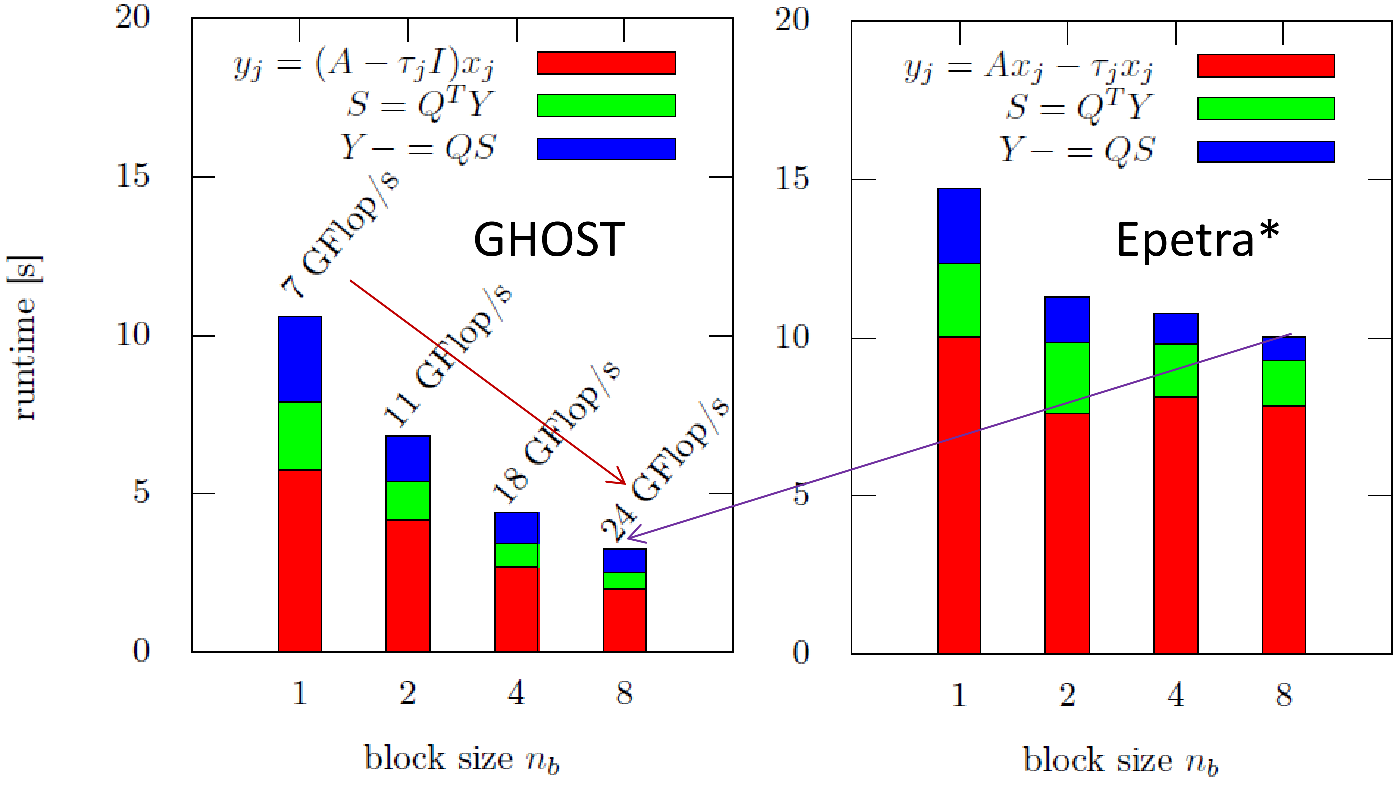
$$(y_j \leftarrow (I - QQ^T)(A - \tau_j I)x_j)$$

Dense matrix (Tall & skiny)

Sparse Matrix

BLOCKED JADA operation available in GHOST for CPU - GPGPU & Xeon Phi: work in progress.

Algorithmic Developments: Blocked JADA – performance of basic operation



Matrix:
D=10^7; n_{nzr}=14
Intel Xeon E5-2660 v2
120 JADA operations

3.3x over-compensates numerical overhead of blocking!

2.5x vs. Trilinos building blocks

Increasing the Performance of the Jacobi-Davidson Method by blocking
M. Röhrig-Zöllner, J. Thies, A. Basermann et al., SIAM SISC, in print.

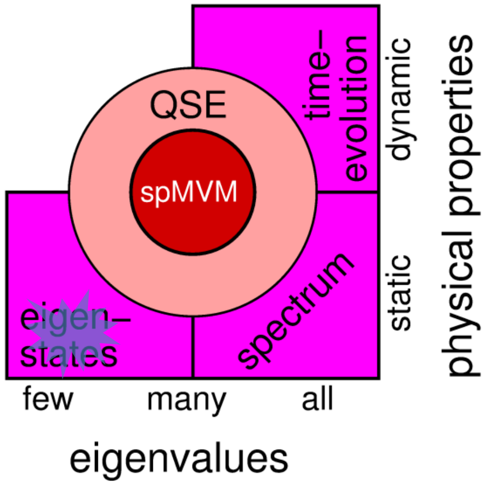
*http://trilinos.sandia.gov/packages/epetra/

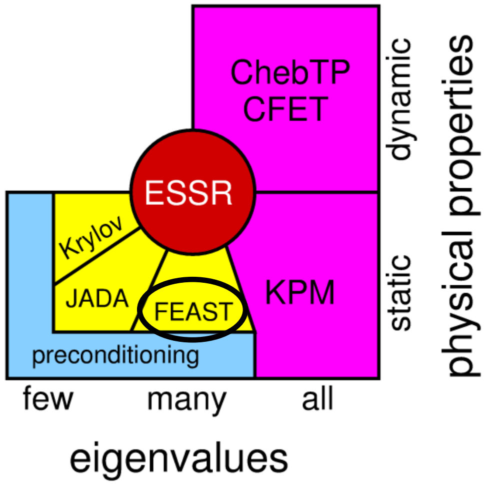
DLR

Algorithmic Developments: FEAST method and CARP-CG solver

Compute ***l* interior eigenvalues/-vectors** $\{(\lambda_i, v_i), i = 1, \ldots, l\}$
of sparse matrix A:

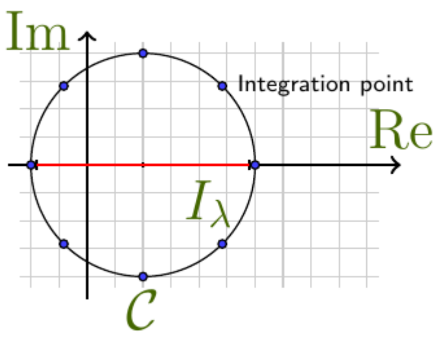
$$A\, v_i = \lambda_i v_i$$





Algorithmic Developments: FEAST – Progress towards Large Scale

- FEAST = Numerical integration + Rayleigh-Ritz
- Eigenvalues in given **interval**
- Numerical Integration:
  Solution of **many large** linear systems



Achievements:

- Estimation of eigenvalue count (also with **KPM**)
- Integration of linear solver **CARP-CG**
- Graphene eigenvalue problems
- Substitution of linear solver by **polynomials**

On the parallel iterative solution of linear systems arising in the FEAST algorithm for computing inner eigenvalues
J. Thies, A. Basermann, B. Lang et al.:
Parallel Computing 49 (2015) 153–163

→ Few inner eigenvalues of graphene problem of size **10^8**

Compare with state of the art FEAST: 10^5 using direct sparse solver

Algorithmic Developments: CARP-CG Preconditioner for Inner Eigenproblems

FEAST eigensolver yields challenging linear systems
(indefinite, random entries, small diagonal elements)

CARP-CG: a Conjugate Gradient accelerated Kaczmarz method

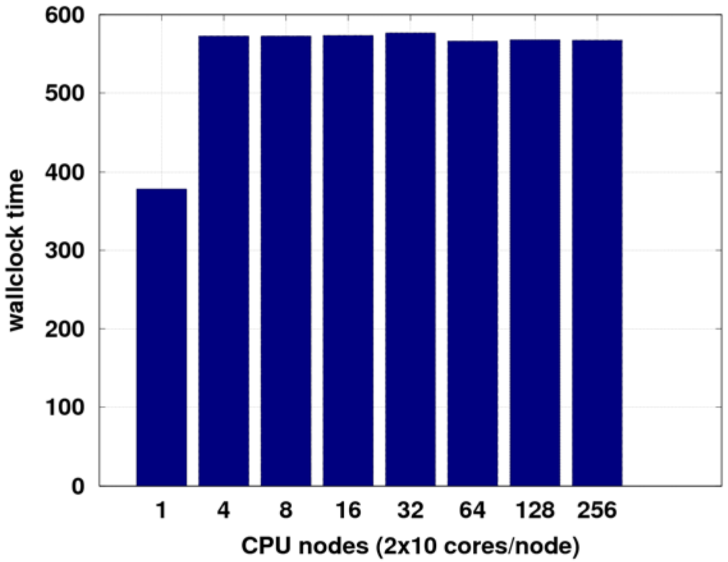
- Numerically very robust
- Sparse kernel: successive row projections ($a_{k,:}$ is the k'th row of A)

$$x_{k+1} \leftarrow x_k - (a_{k,:} x_k) a_{k,:}^T$$

- Data dependency resolved by node-local graph coloring
- Component averaging between nodes (recovers global Kaczmarz)

- Not yet fully optimized in GHOST

Weak scaling

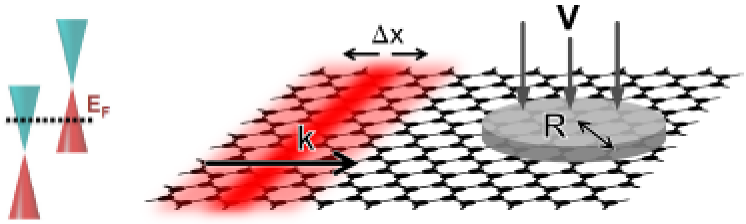


Application results

Graphene nanoribbon (GNR) with gate-defined quantum dots

$$H = \sum_i V(i) c_i^\dagger c_i - t \sum_{\langle i,j \rangle} c_i^\dagger c_j$$

$$V(i) = V \sum_n \Theta(R - |\vec{r}_i - \vec{r}_n|)$$

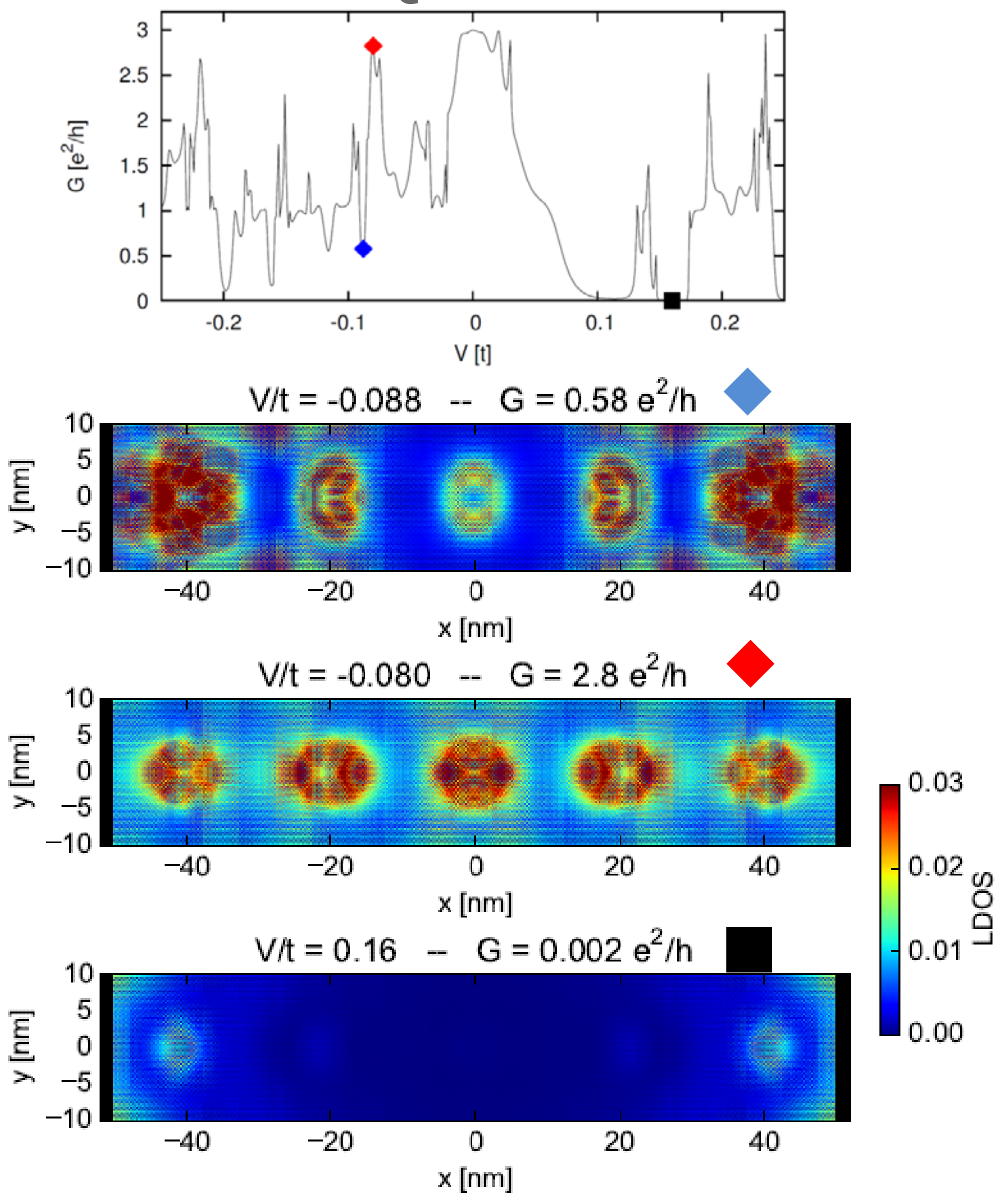


Application results: GNR with 5 Gate Defined Quantum Dots

- Conductivity **G** controlled by dot potential **V/t**

- Small change in V/t
  → large change in **G**
  → GNR may realize very sensitive switch

- Superlattice – opening of band gap
  → Vanishing conductance

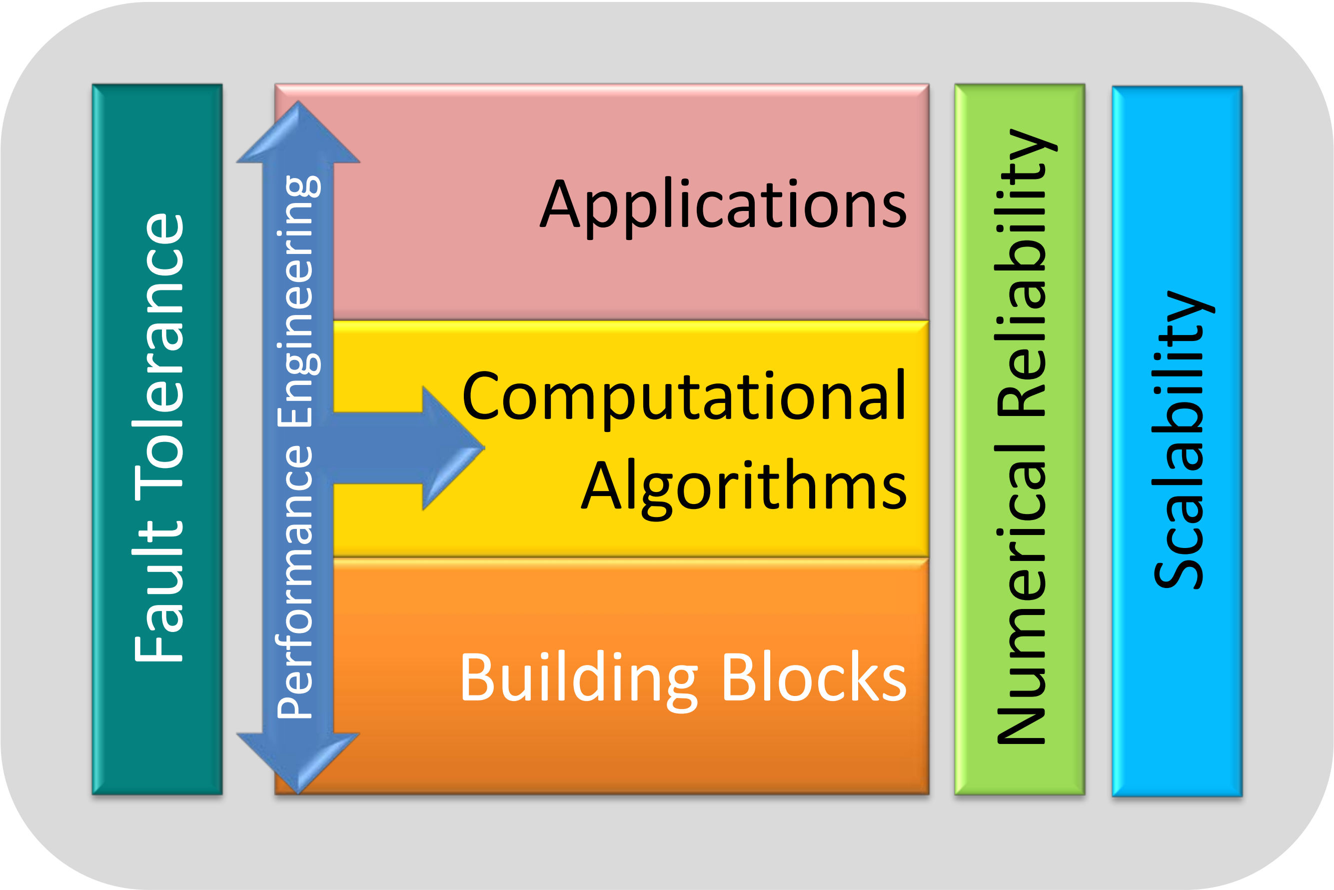


Conclusions

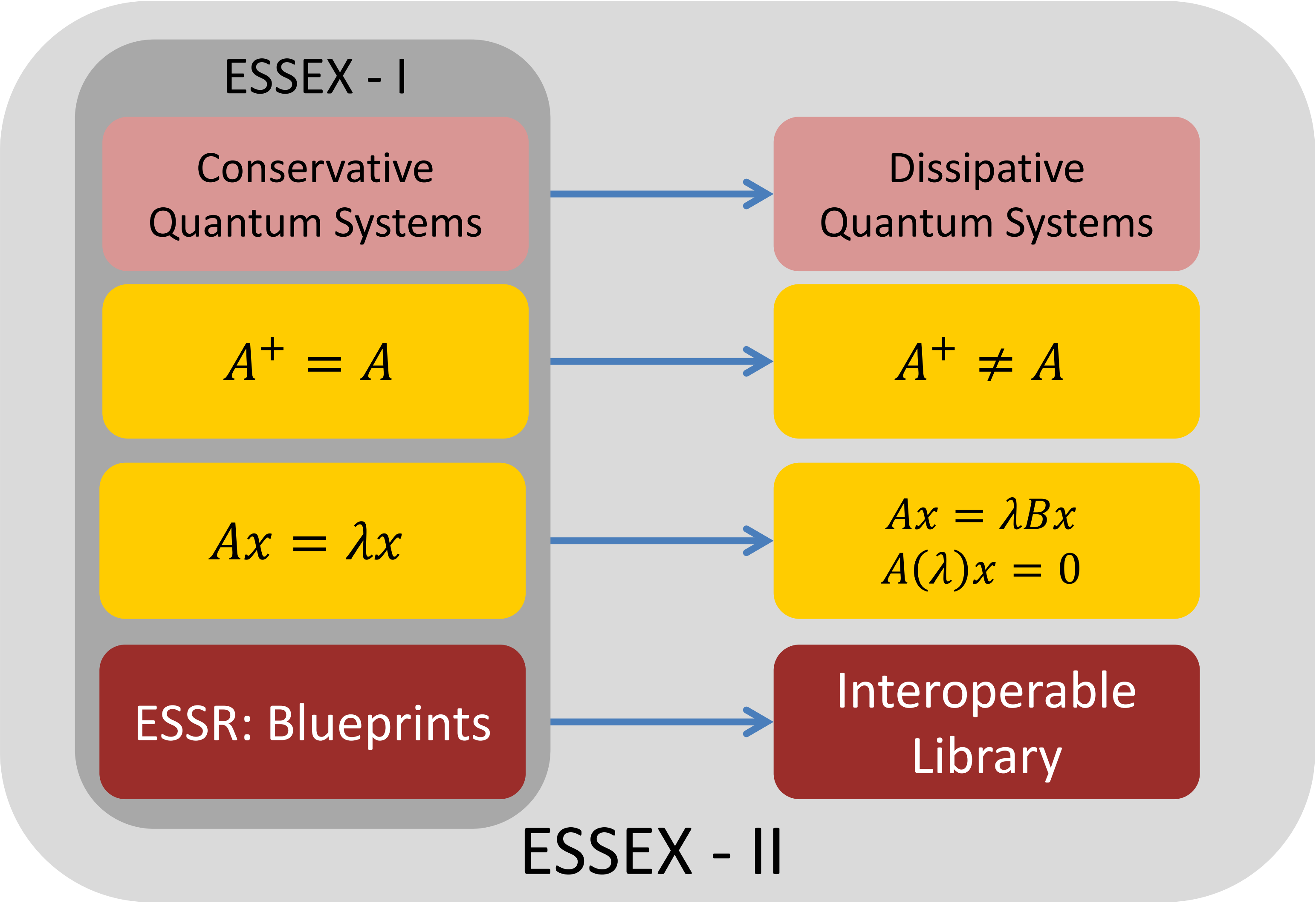
- Holistic performance engineering strategie successful for developing highly scalable solutions, cf. KPM.
- ***PHIST*** with GHOST provides a pragmatic, flexible and hardware-aware programming model for heterogeneous systems.
  - Includes highly scalable sparse iterative solvers for eigenproblems and systems of linear equations
  - Well suited for iterative solver development and solver integration into applications
- Block operations distinctly increase performance of building blocks for iterative eigensolvers like KPM or JADA.
- CARP-CG with node-level multi-coloring parallelization is suitable for robust iterative solution of the nearly singular equations.
  - Appropriate iterative solver for FEAST in order to find interior eigenpairs,
  - in particular for problems from graphene design
- First convincing results with quantum physics applications

The Future: ESSEX II

- DFG confirmed ESSEX extension to 2018.
- Additional partners from Japan
  - Kengo Nakajima, Computer Science, University of Tokyo
  - Tetsuya Sakurai, Applied Mathematics, University of Tsukuba
- Main objectives
  - Enabling Exascale through software co-design
  - Established exascale sparse solver repository



Project Evolution

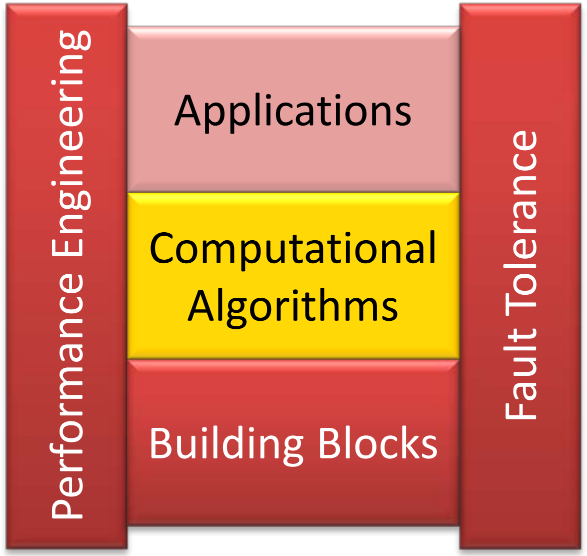


Programming

Extended

Building Blocks, Parallelization, and Performance Engineering

- Holistic performance and power engineering
- Advanced building blocks engineering



Extended

Fault Tolerance

- From prototype to application software
  - Asynchronous checkpointing & I/O
  - Automatically fault-tolerant applications

NEW

Numerical Reliability

- Performance aspects
  - Silent data corruption / skeptical programming
  - High-precision reduction operations

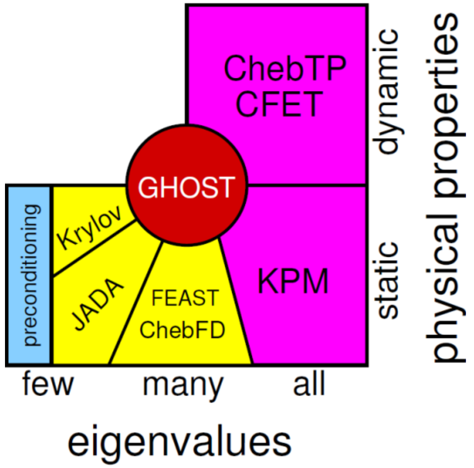
Computational Algorithms

NEW

- **Non-Hermitian**: ChebTP / CFET / JaDa
  - Extreme-scale simulations for dissipative quantum systems
  - Numerical range computation & matrix balancing

Extended

- **Cheby**shev**F**ilter**D**iagonalization:
  - $>10^3$ interior eigenvalues of $>10^9$ matrix dim.
  - Simple, HW-efficient & low synchronization cost



NEW

- **Preconditioning** & **Communication Hiding**
  - Asynchronous JaDa: "pipelining" & preconditioning
  - AMG preconditioning for blocked JaDa & FEAST

Kengo Nakajima, University of Tokyo

NEW

- Leveraging FEAST techniques + GHOST
  → Nonlinear Sakurai-Sugiura Method (NSSM)

Tetsuya Sakurai, University of Tsukuba

DLR

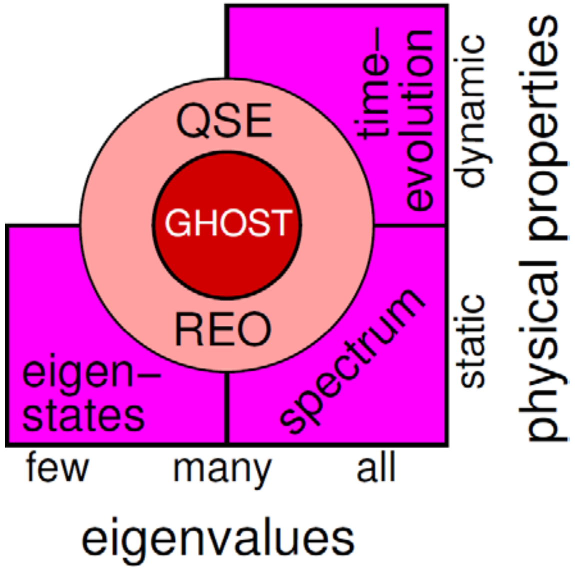
Applications

Extended

- **Quantum State Encoding (QSE)**
  - Complex (non-stencil) matrix structure encoding
  - Dissipative systems: Sparse ⊗ Dense

New

- **Matrix Reordering Strategies (REO)**
  - Application-specific
  - General techniques, e.g. **PMRSB**



Extended

- **Quantum Physics/Information Applications**
  - **Topological materials**
    Graphene & topological insulators
  - **Dissipative quantum systems**
    Light-harvesting molecules & optomechanics
  - Rich collection of quantum physics problems

$A^{+} = A$

$A^{+} \neq A$

Thanks

Thanks to all partners from the **ESSEX** project

and to DFG for the support through the Priority Programme 1648 “Software for Exascale Computing”.

Computer Science, Univ. Erlangen

Applied Computer Science, Univ. Wuppertal

Institute for Physics, Univ. Greifswald

Erlangen Regional Computing Center

International contacts

Sandia (Trilinos project)
Tenessee (Dongarra)
Japan: Tsukuba, Tokyo
The Netherlands: Groningen, Utrecht

Many thanks for your attention!

Questions?

Dr.-Ing. Achim Basermann

German Aerospace Center (DLR)

Simulation and Software Technology

Department Distributed Systems and Component Software

Team High Performance Computing

Achim.Basermann@dlr.de

http://www.DLR.de/sc