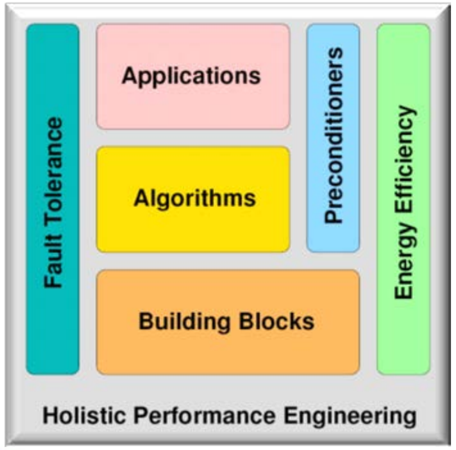
SPPEXA

DFG Projekt ESSEX

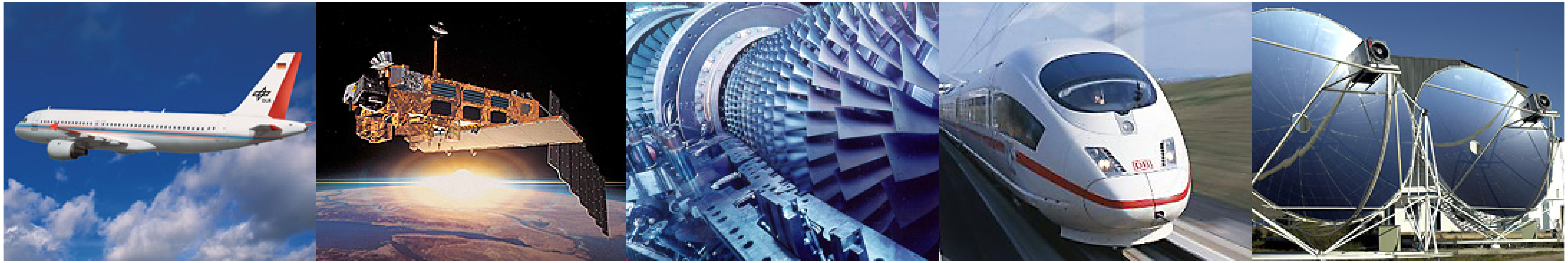


A Software Infrastructure for Solving Quantum Physics Problems on Extremely Parallel Systems

Achim Basermann, Jonas Thies, Melven Röhrig-Zöllner
German Aerospace Center (DLR)
Simulation and Software Technology
Department High-Performance Computing
Linder Höhe, Cologne, Germany

DLR

DLR
German Aerospace Center



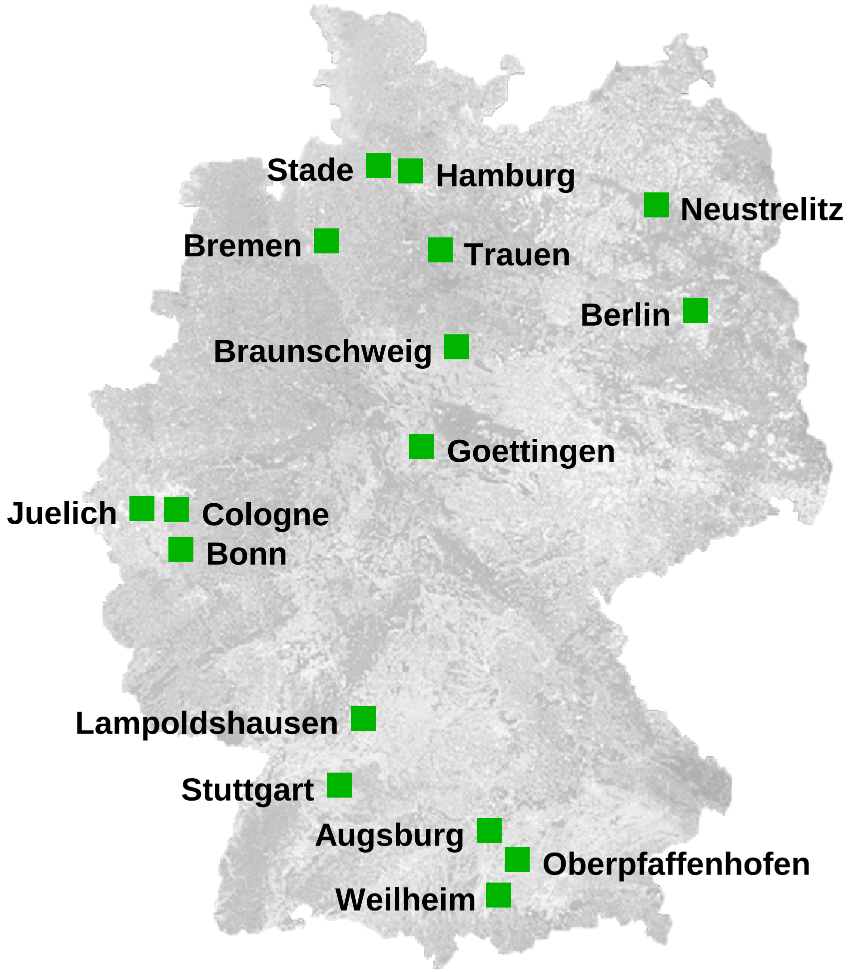
- Research Institution
- Space Agency
- Project Management Agency

DLR Locations and Employees

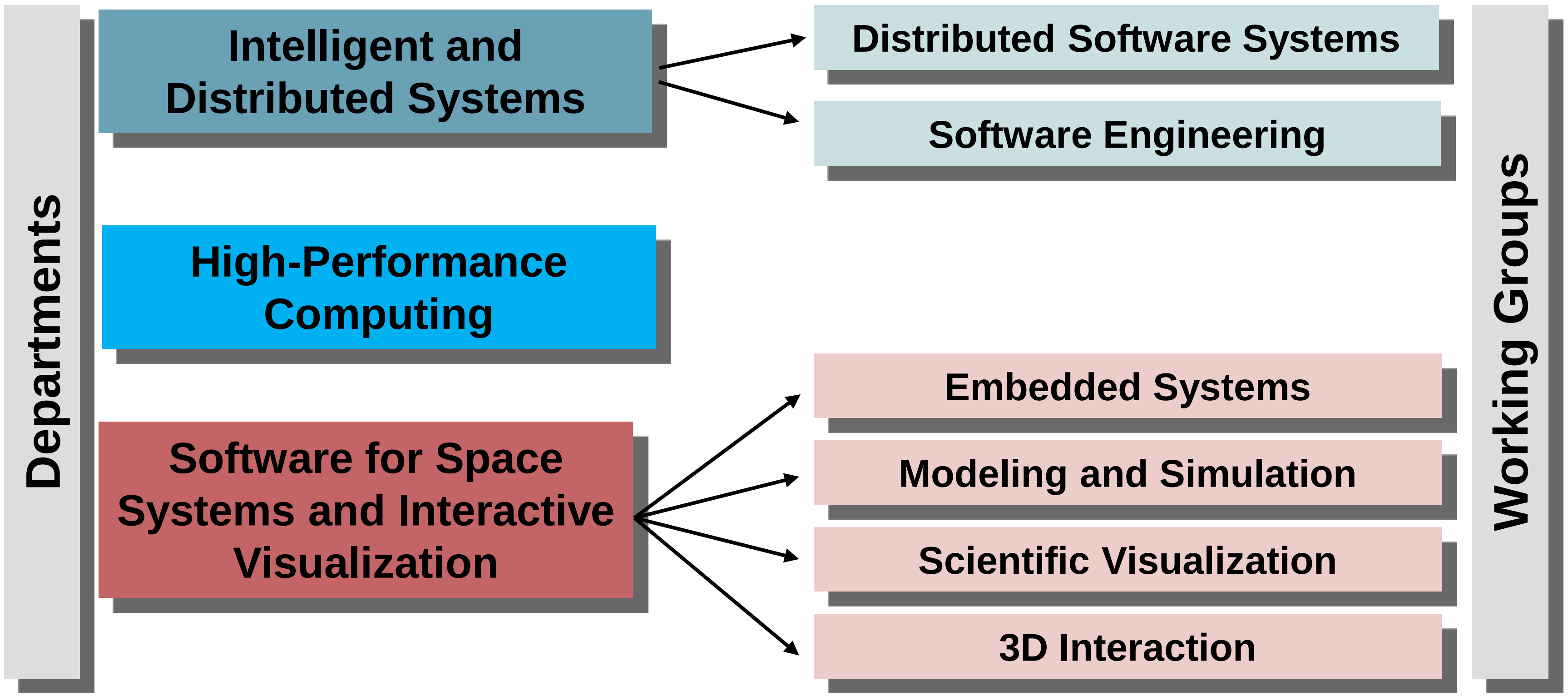
Approx. 8000 employees across 33 institutes and facilities at

- 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
- Conclusions
- Future work

ESSEX Motivation: Requirements for Exascale

Hardware
Fault tolerance
Energy efficiency
New levels of parallelism

Quantum Physics Applications
Extremely large sparse matrices:
eigenvalues, spectral properties,
time evolution

ESSEX

Exascale Sparse Solver Repository (ESSR)
ghost / PHIST

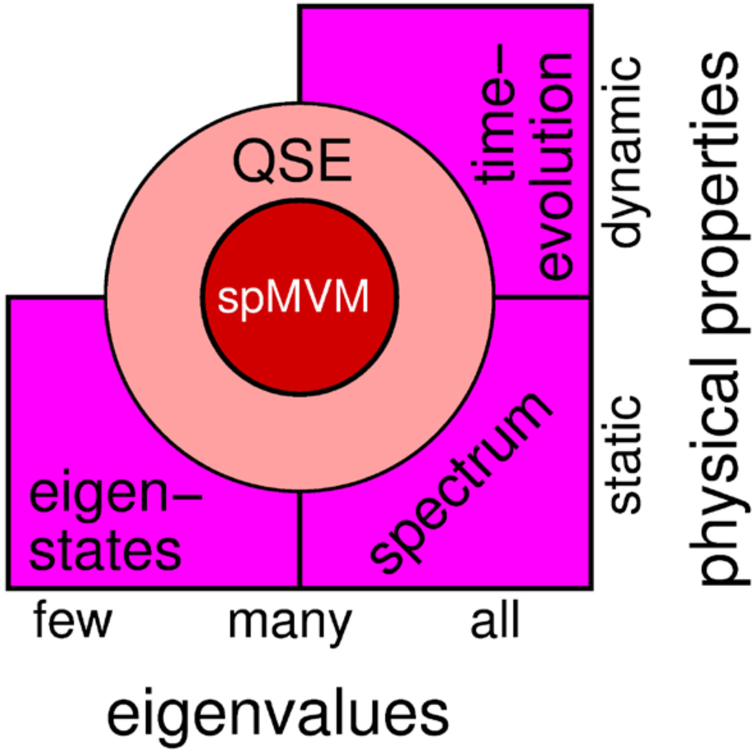
- FT concepts, programming for extreme parallelism
- Sparse eigensolvers, preconditioners, spectral methods
- Quantum physics / chemistry
- ESSEX applications: Graphene, topological insulators, …

ESSEX: Physical Motivation and Sparse Eigenvalue problem

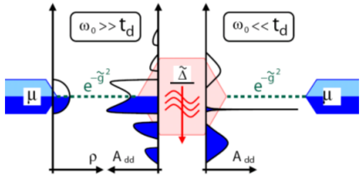
Solve large sparse eigenvalue problem

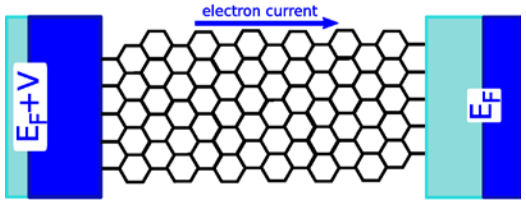
$$\boldsymbol{H}\,\boldsymbol{x} = \boldsymbol{\lambda}\,\boldsymbol{x}$$

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



$(\boldsymbol{\lambda_i}, \mathbf{x_i})$





ESSEX Software Development: Basics

- **Git** for distributed software developement
- **Merge-request workflo**w for code review; changes only in branches
- Own MPI extension for **Google Test**
- Realization of **continuous-integratio**n with Jenkins server

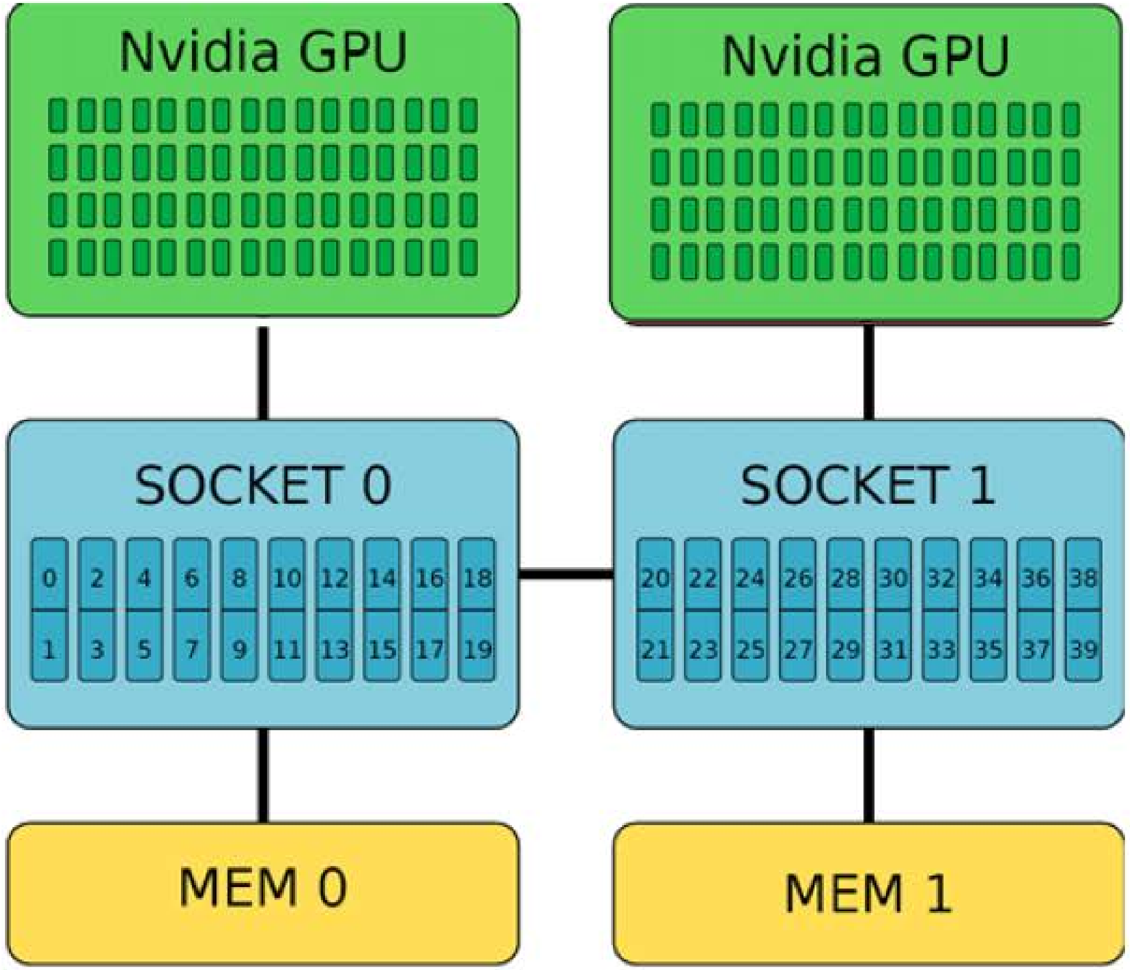
The ESSEX Software Infrastructure: Kernel Library GHOST

(General Hybrid and Optimized Sparse Toolkit) provides

- intelligent resource management for heterogenous systems
  - automatic pinning of threads to cores
  - asynchronous execution of (larger) tasks
- some fully optimized kernels for sparse matrix methods
  - sparse matrix-(multi)vector multiplication (spM(M)VM)
  - 'tall and skinny' matrices in row or column major ordering
- target platforms right now: Intel CPUs, Xeon Phi and Nvidia GPUs
- programming model: 'MPI+X',
  with X=SIMD intrinsics, OpenMP and CUDA

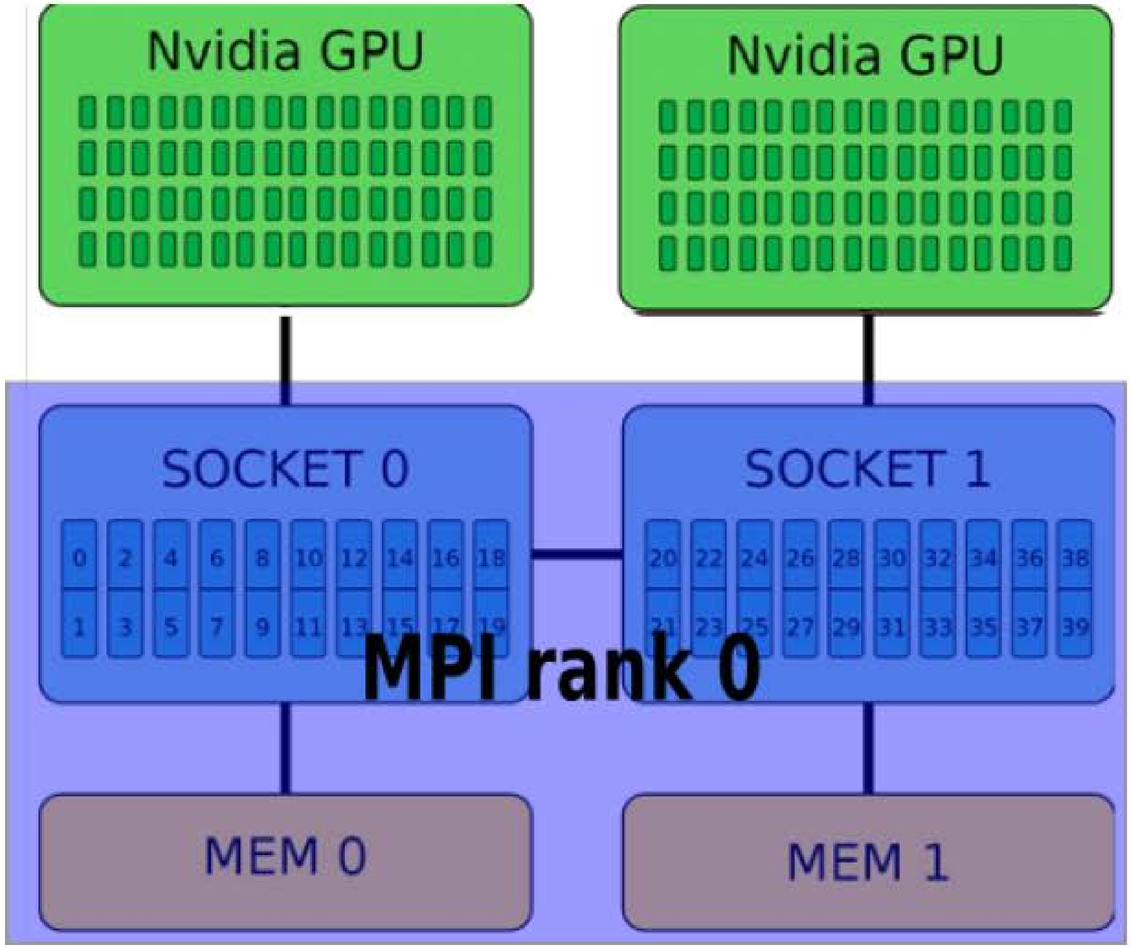
The ESSEX Software Infrastructure: MPI + X with GHOST

- System with multiple CPUs (NUMA domains) and GPUs



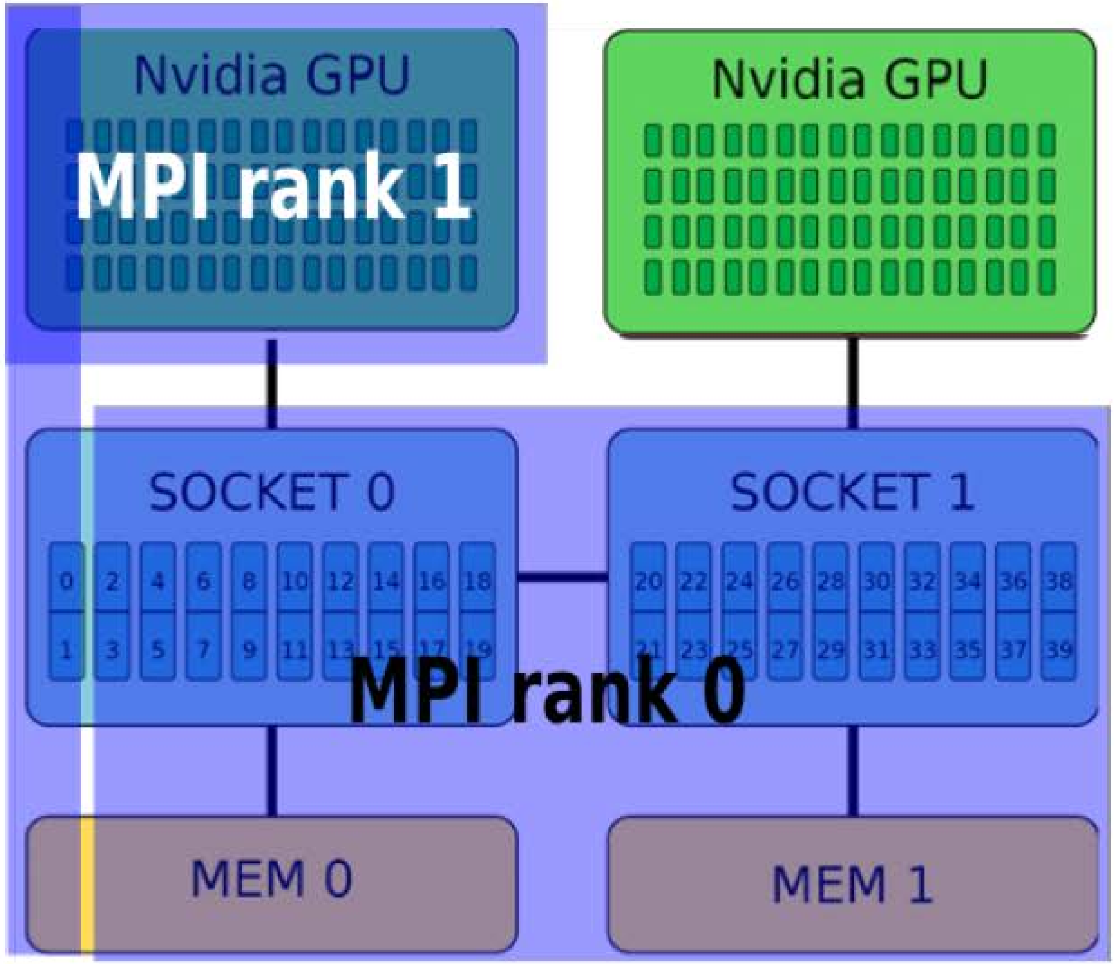
The ESSEX Software Infrastructure: MPI + X with GHOST

- System with multiple CPUs (NUMA domains) and GPUs
- -np 1: use entire CPU



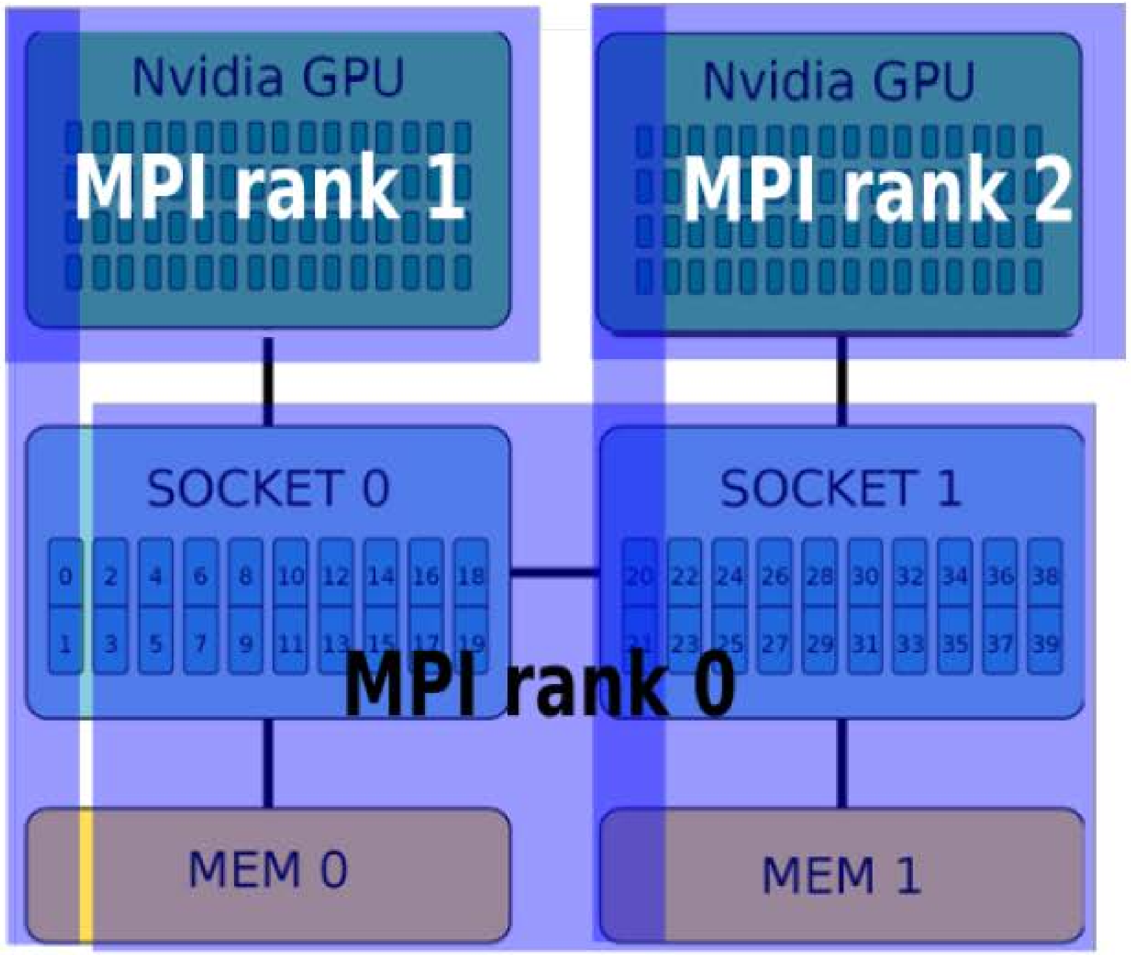
The ESSEX Software Infrastructure: MPI + X with GHOST

- System with multiple CPUs (NUMA domains) and GPUs
- -np 1: use entire CPU
- -np 2: use CPU and first GPU



The ESSEX Software Infrastructure: MPI + X with GHOST

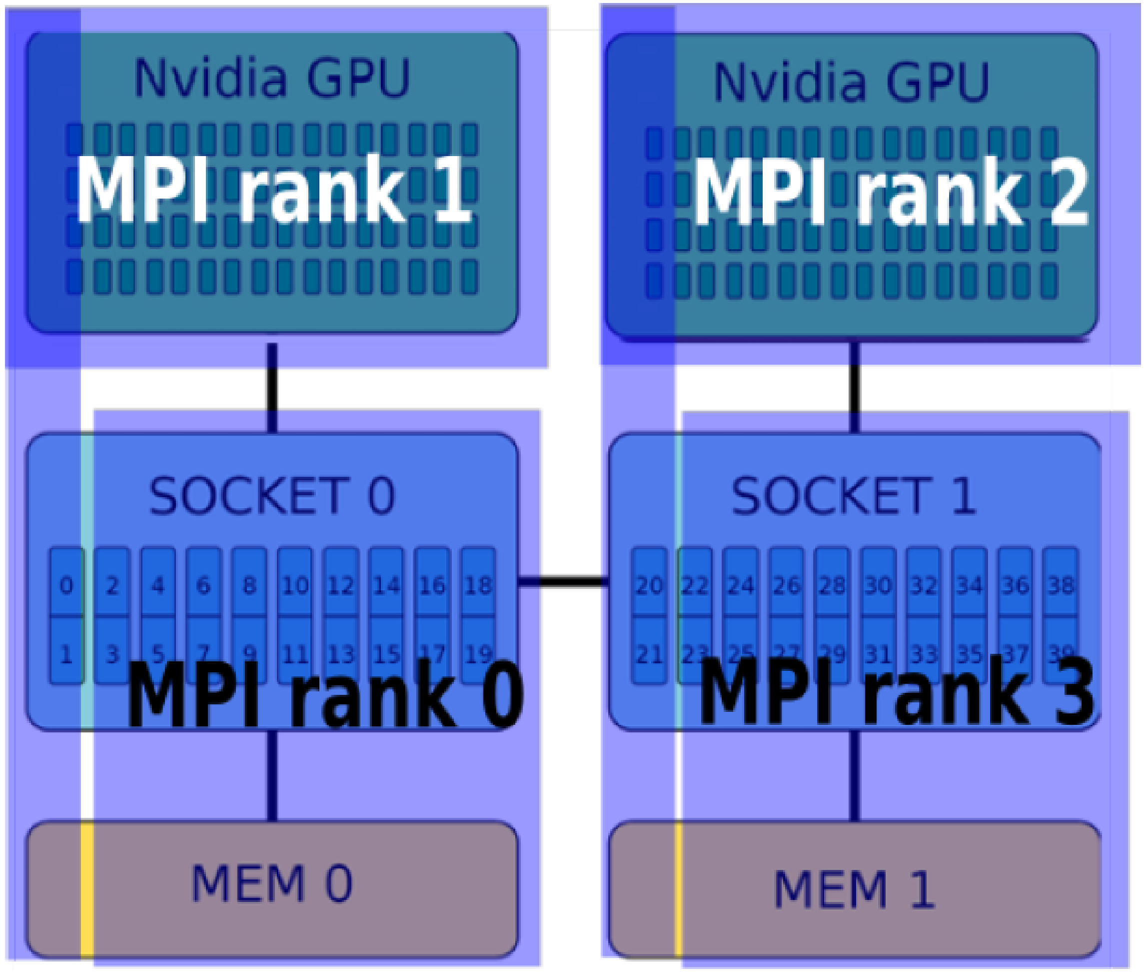
- System with multiple CPUs (NUMA domains) and GPUs
- -np 1: use entire CPU
- -np 2: use CPU and first GPU
- -np 3: use CPU and both GPUs



The ESSEX Software Infrastructure: MPI + X with GHOST

- System with multiple CPUs (NUMA domains) and GPUs
- -np 1: use entire CPU
- -np 2: use CPU and first GPU
- -np 3: use CPU and both GPUs
- -np 4: use one process per socket and one for each GPU

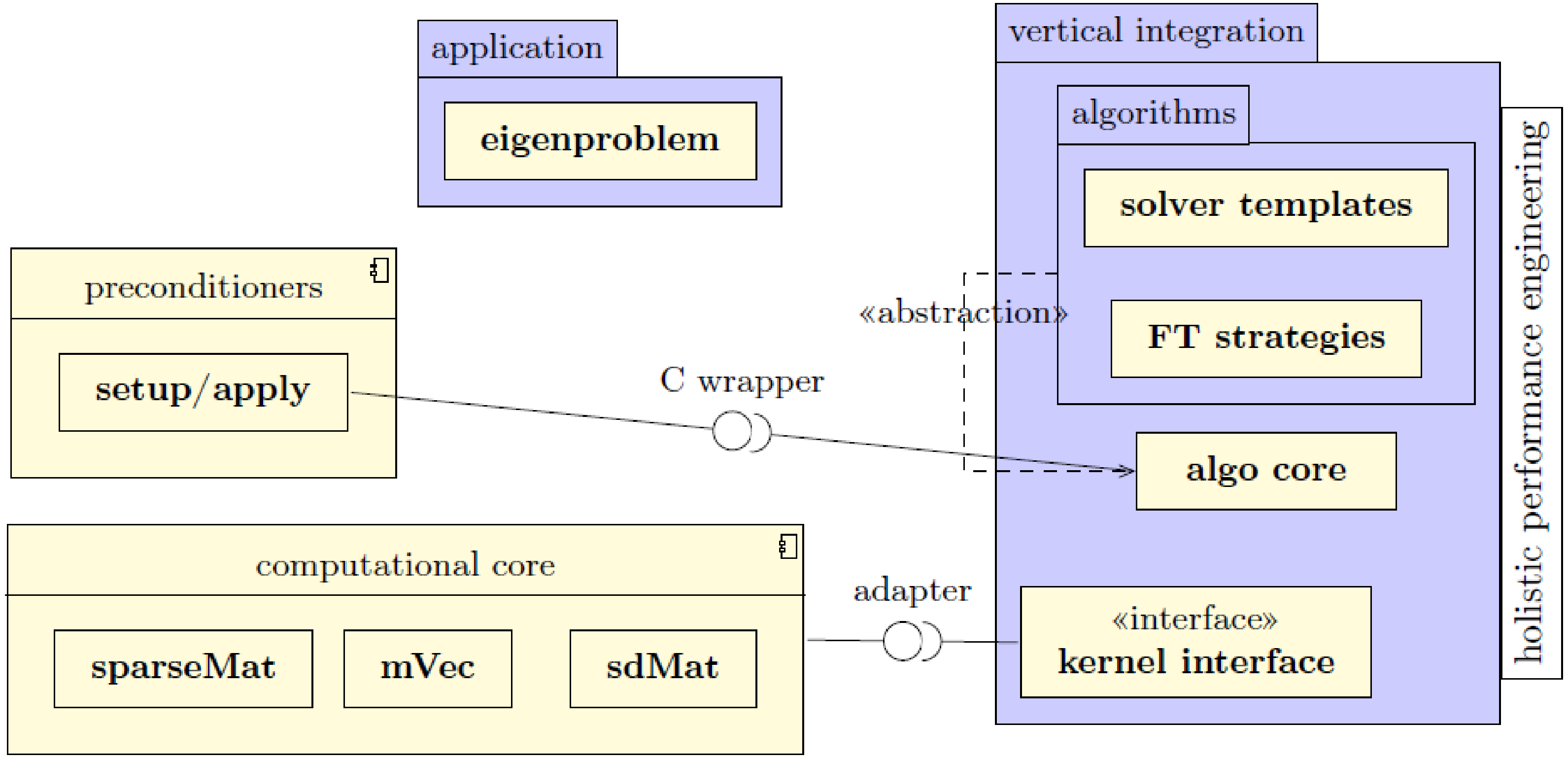
Option: distribute problem according to memory bandwidth measured



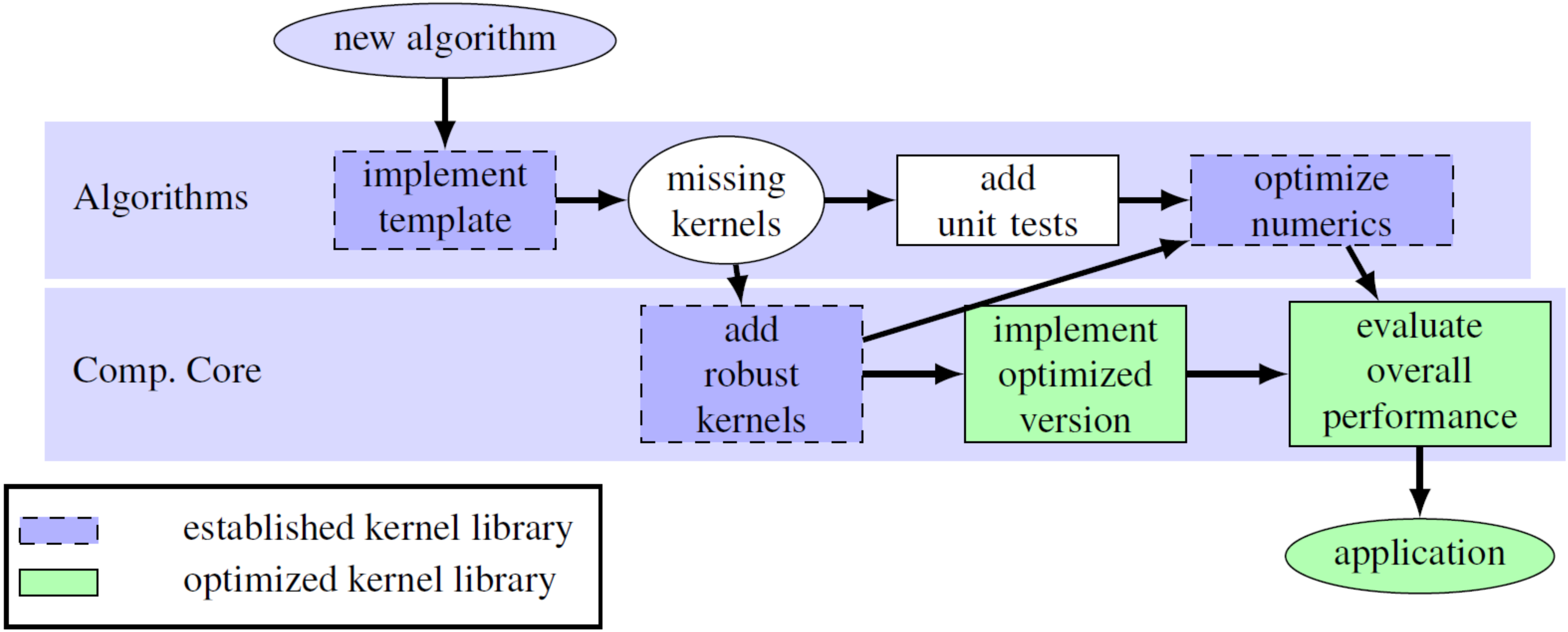
The ESSEX Software Infrastructure: PHIST for Implementing Iterative Solvers

a Pipelined Hybrid-parallel Iterative Solver Toolkit

- facilitate algorithm development using GHOST
- holistic performance engineering
- portability and interoperability



The ESSEX Software Infrastructure: Test-Driven Algorithm Development

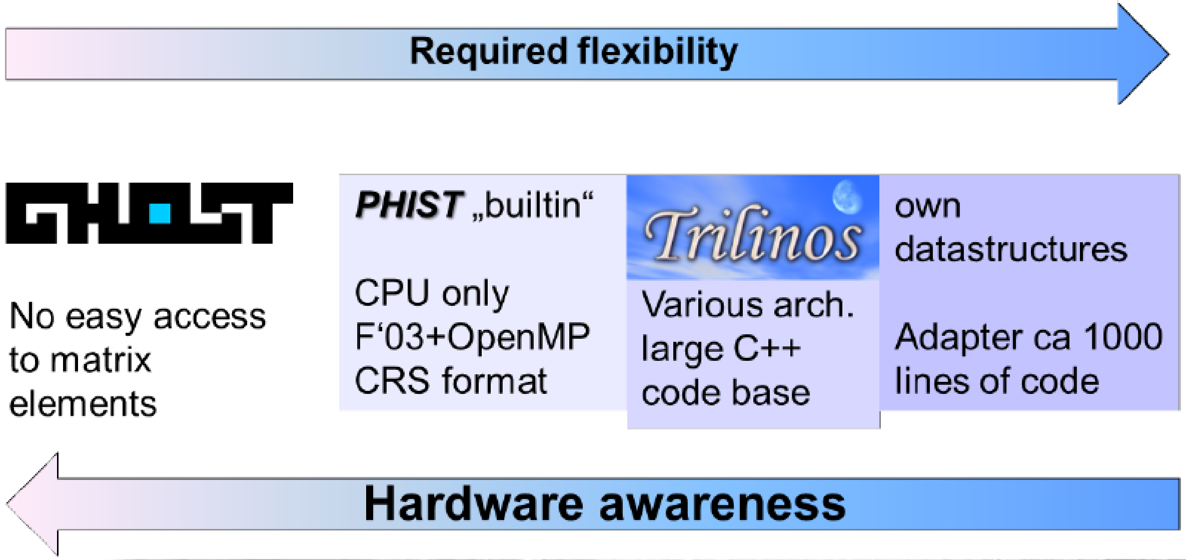


The ESSEX Software Infrastructure: PHIST for Implementing Iterative Solvers

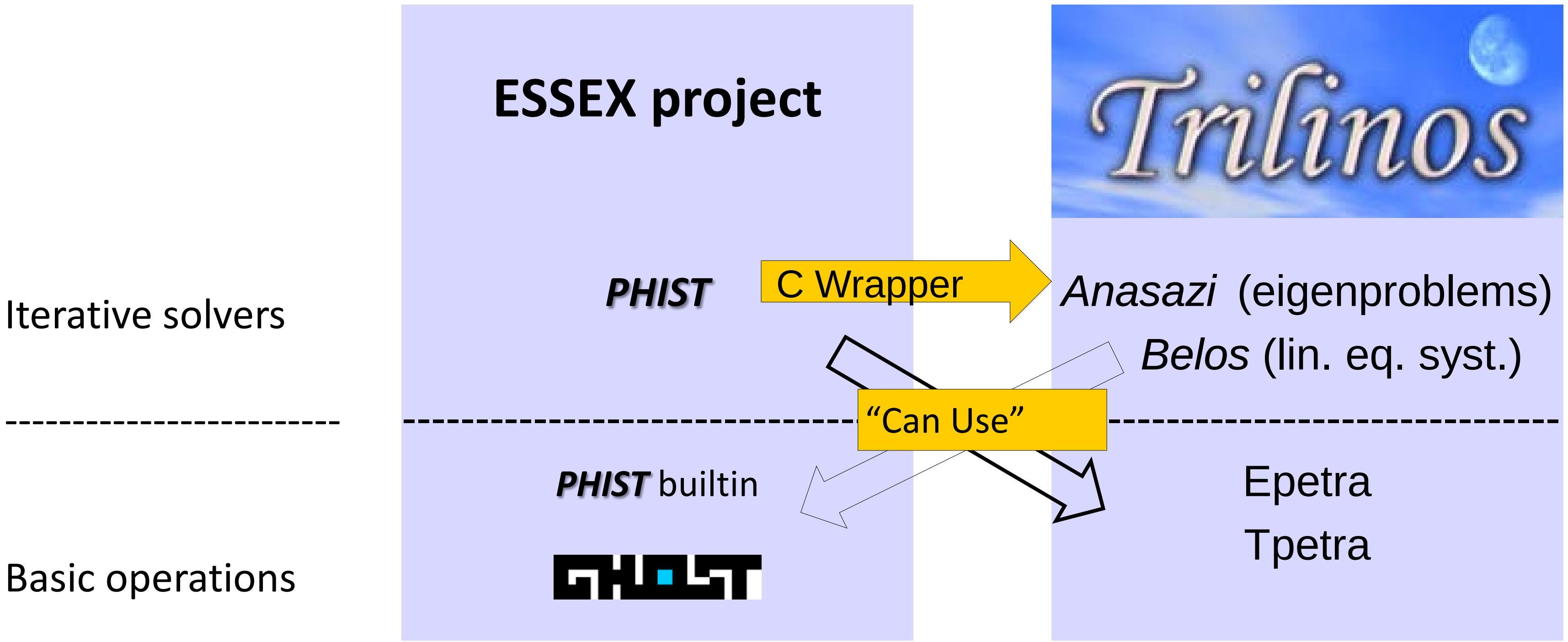
Useful abstraction: kernel interface

Choose from several 'backends' at compile time, to

- easily use **PHIST** in existing applications
- perform the same run with different kernel libraries
- compare numerical accuracy and performance
- exploit unique features of a kernel library (e.g. preconditioners)



Interoperability of PHIST and Trilinos



The ESSEX Software Infrastructure: PHIST for Implementing Iterative Solvers

Cool features of PHIST

Task macros

out-of-order execution of code blocks

- overlap comm. and comp.
- asynchronous checkpointing
- ...

Consistent random vectors

make **PHIST** runs comparable

- across platforms (CPU, GPU...)
- across kernel libraries
- independent of #procs, #threads

PerfCheck:

print achieved roofline performance of kernels after complete run to reveal

- deficiencies of kernel lib
- implementation issues of algorithm (strided data access etc.)

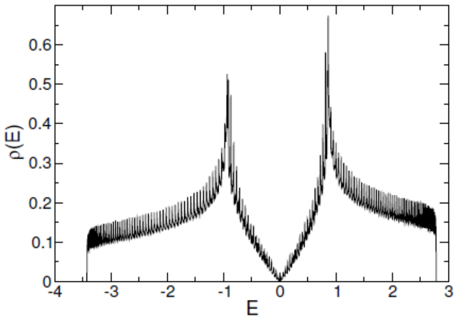
Special-purpose operations

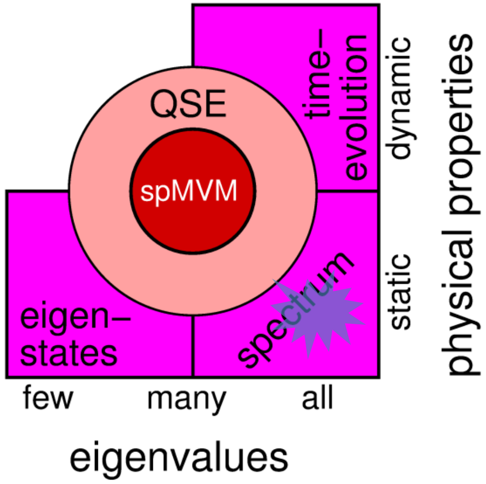
- fused kernels, e.g. compute $Y = \alpha AX + \beta Y$ *and* $Y^T X$
- highly accurate core functions, e.g. block orthogonalization in simulated quad precision

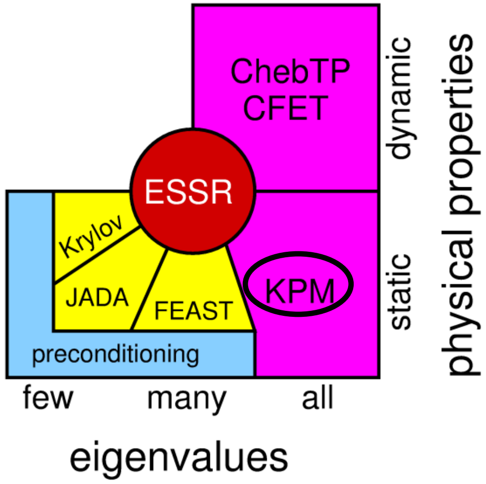
Application, Algorithm and Performance: Kernel Polynomial Method (KPM) – A Holistic View

- 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**
$\quad |v\rangle \leftarrow |\text{rand}()\rangle$
$\quad$ Initialization steps and computation of η_0, η_1
$\quad$ **for** $m = 1$ to $M/2$ **do**
$\quad\quad \text{swap}(|w\rangle, |v\rangle)$
$\quad\quad |u\rangle \leftarrow H|v\rangle$ ▷ spmv()
$\quad\quad |u\rangle \leftarrow |u\rangle - b|v\rangle$ ▷ axpy()
$\quad\quad |w\rangle \leftarrow -|w\rangle$ ▷ scal()
$\quad\quad |w\rangle \leftarrow |w\rangle + 2a|u\rangle$ ▷ axpy()
$\quad\quad \eta_{2m} \leftarrow \langle v|v\rangle$ ▷ nrm2()
$\quad\quad \eta_{2m+1} \leftarrow \langle w|v\rangle$ ▷ dot()
$\quad$ **end for**
end for

Application:
Loop over random initial states

Algorithm:
Loop over moments

Building blocks:
(Sparse) linear algebra library

- spmv() – Sparse matrix vector multiply
- axpy() – Scaled vector addition
- scal() – Vector scale
- axpy() – Scaled vector addition
- nrm2() – Vector norm
- dot() – Dot Product

DLR

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!

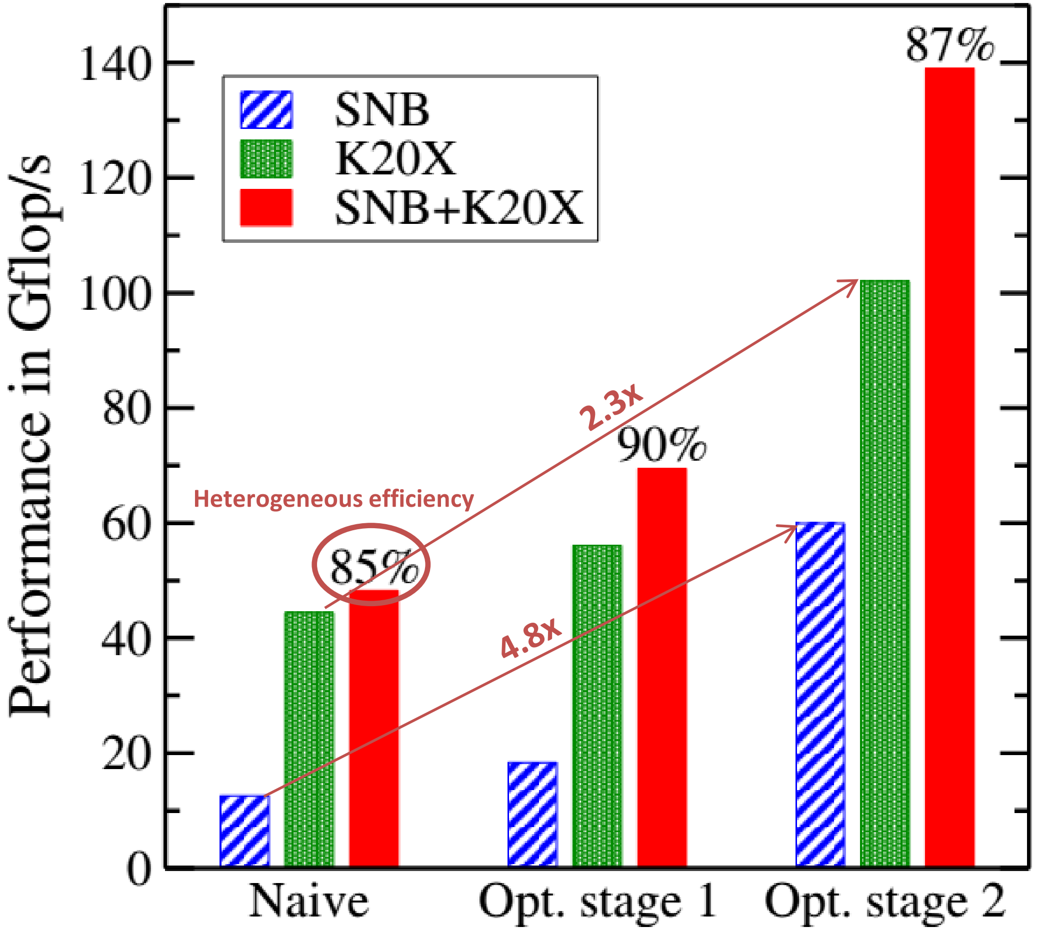
Basic algorithm – Compute Cheyshev polynomials/moments:

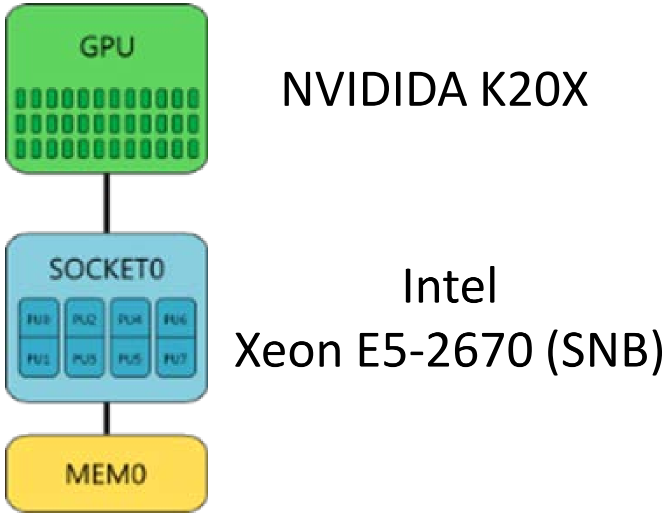
for $r = 0$ to $R-1$ **do**
  $|v\rangle \leftarrow |\text{rand}()\rangle$
  Initialization steps and computation of η_0, η_1
  for $m = 1$ to $M/2$ **do**
    $\text{swap}(|w\rangle, |v\rangle)$
    $|w\rangle = 2a(H - b\mathbb{1})|v\rangle - |w\rangle$ &
      $\eta_{2m} = \langle v|v\rangle$ &
  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

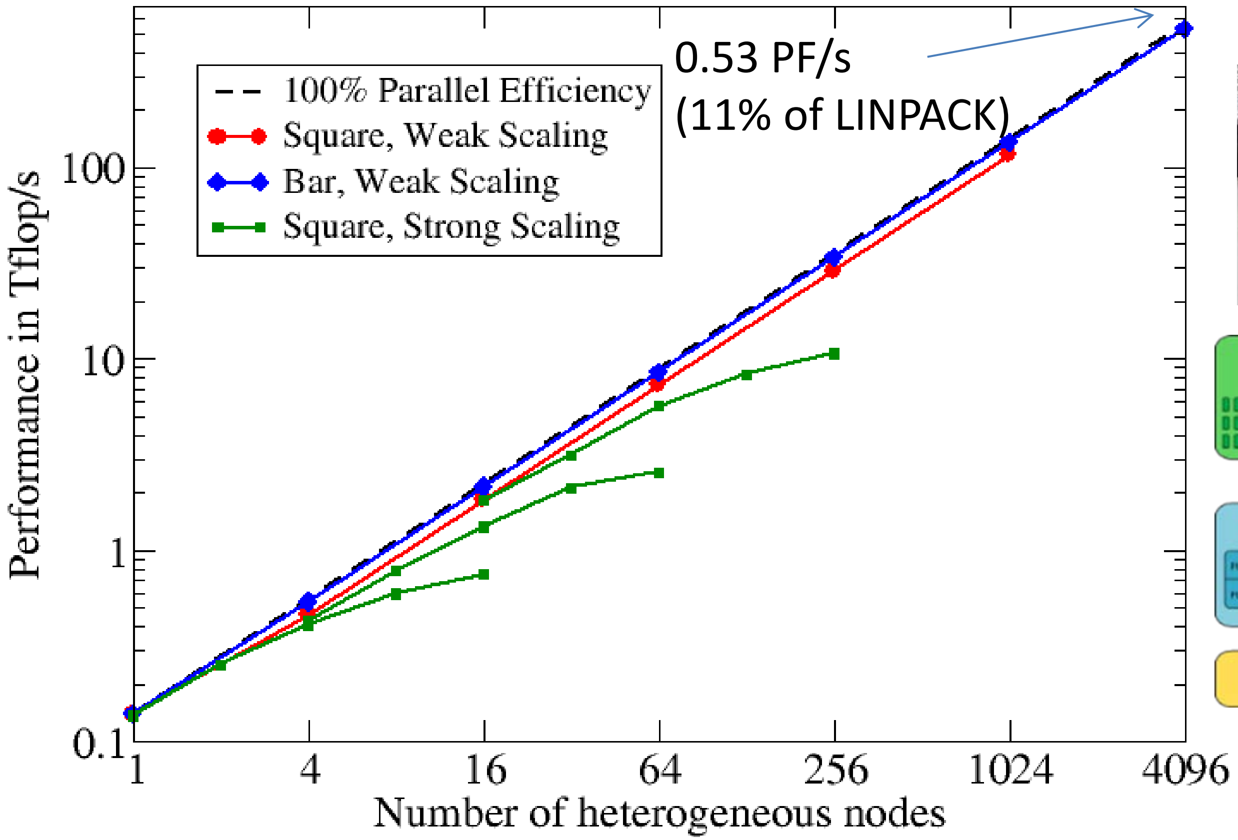
KPM: Heterogenous Node Performance



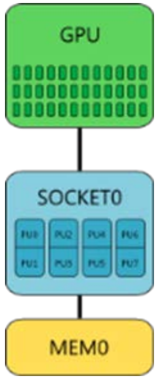


- Topological Insulator Application
- Double complex computations
- Data parallel static workload distribution

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

Conclusions

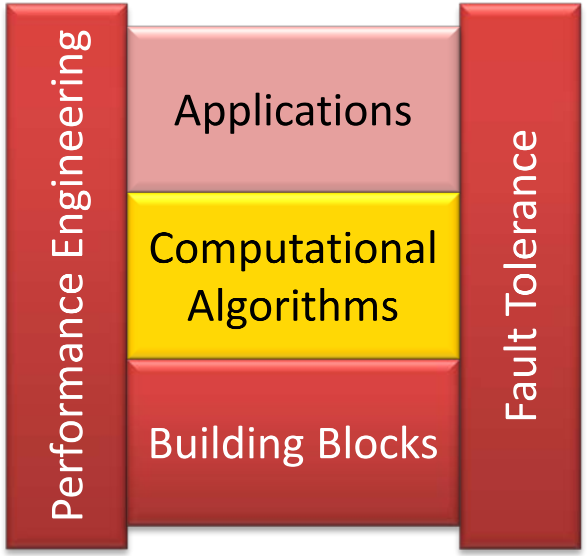
- Holistic performance engineering strategie successful for developing highly scalable solutions, cf. KPM.
- ***PHIST*** with GHOST provides a pragmatic, flexible and hardware-aware programming model for heterogeneous systems.
  - Includes highly scalable sparse iterative solvers for eigenproblems and systems of linear equations
  - Well suited for iterative solver development and solver integration into applications
- First convincing results with quantum physics applications

Future Work: 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

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

- project website incl. list of publications: http://blogs.fau.de/essex/
- source code: https://bitbucket.org/essex/[ghost|phist]

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 High-Performance Computing

Achim.Basermann@dlr.de

http://www.DLR.de/sc