Performance Engineering for Sparse Eigensolvers on Heterogenous Clusters

Jonas Thies Melven Röhrig-Zöllner Moritz Kreutzer
Achim Basermann Georg Hager Gerhard Wellein

German Aerospace Center
Simulation and Software Technology

and University of Erlangen

SPPEXA

project ESSEX

DLR

Motivation

Mathematical problem

- Find 20 – 50 eigenpairs

$$Ax_i = \lambda_i x_i$$

of a large, sparse matrix A

- interior or extreme λ_i
- symmetric or general A

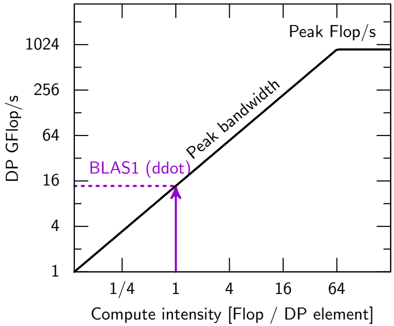
Memory gap

- small memory bandwidth vs. high peak flop rate

→ increase the compute intensity

Roofline performance model

(2x 12 core Haswell EP)



Block JDQR Method

Block Jacobi-Davidson correction equation

- n_b current approximations: $A\tilde{v}_i - \tilde{\lambda}_i\tilde{v}_i = r_i$, $i = 1, \ldots, n_b$
- previously converged Schur vectors $(q_1, \ldots, q_k) = Q$
- solve approximately (with $\tilde{Q} = \begin{pmatrix} Q & \tilde{v}_1 & \ldots & \tilde{v}_{n_b} \end{pmatrix}$):

$$(I - \tilde{Q}\tilde{Q}^T)(A - \tilde{\lambda}_i I)(I - \tilde{Q}\tilde{Q}^T)x_i = -r_i \qquad i = 1, \ldots, n_b$$

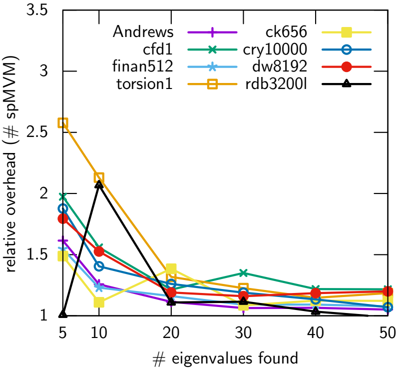
- use some steps of a block(ed) iterative solver
- orthogonalize new directions $x_1, \ldots, x_{n_b}$ (outer subspace iteration)

Properties (compared to single-vector method)

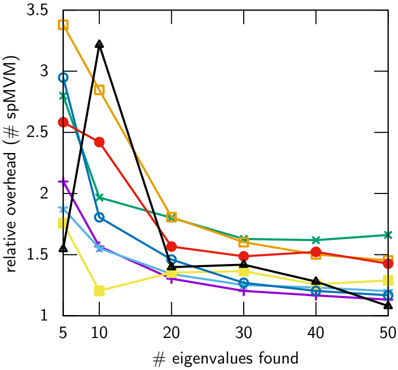
- usually needs more operations → shunned in practice
- more cache-friendly, fewer global operations

Numerical Behavior

Block size 2



Block size 4



from: Röhrig-Zöllner *et al.* SISC 2015

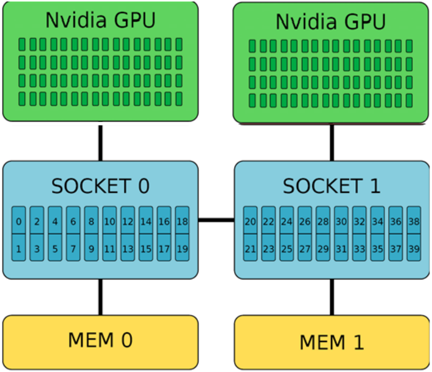
Software I: GHOST

(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

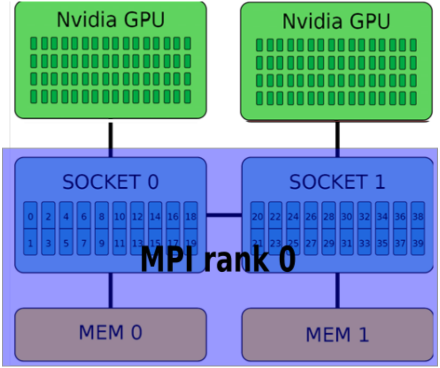
MPI+X with GHOST

- System with multiple CPUs (NUMA domains) and GPUs



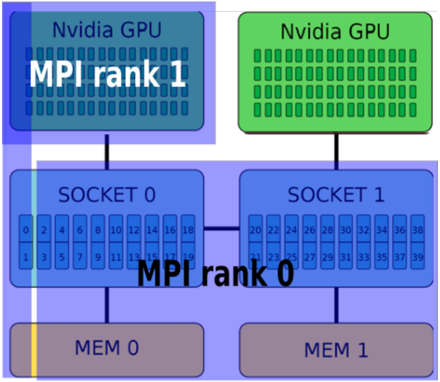
MPI+X with GHOST

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



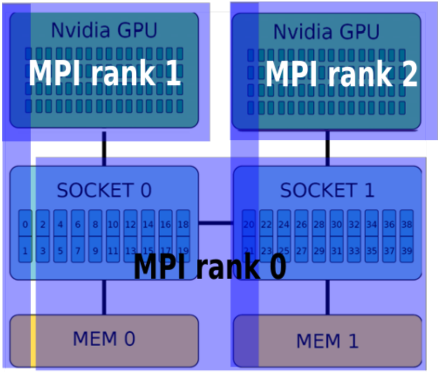
MPI+X with GHOST

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



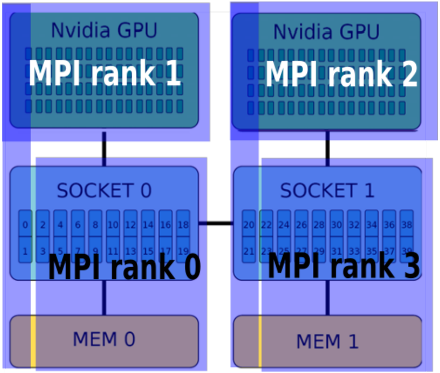
MPI+X with GHOST

- 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



MPI+X with GHOST

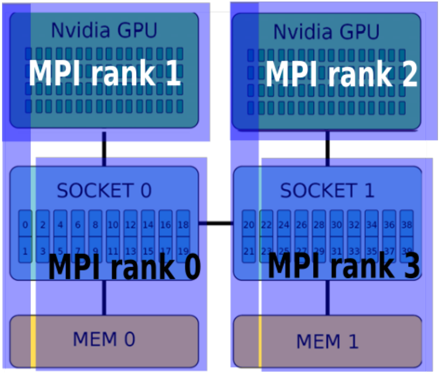
- System with multiple CPUs (NUMA domains) and GPUs
- -np 1: use entire CPU
- -np 2: use CPU and first GPU
- -np 3: use CPU and both GPUs
- -np 4: use one process per socket and one for each GPU



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



What GHOST is NOT

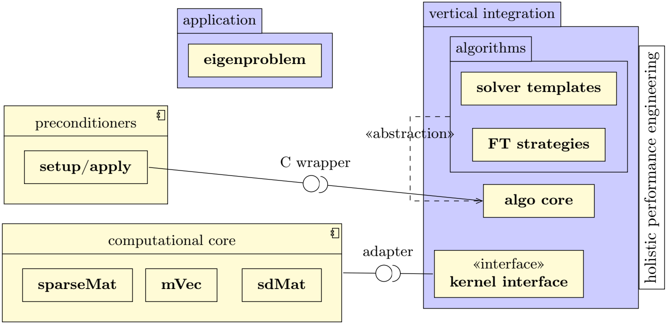
- a DSL for programming heterogenous hardware
- easily portable to platforms other than Intel and Nvidia
- easy to integrate in existing code
- a mature library

$\Longrightarrow$ For implementing iterative solvers we use an interface layer (up next)

Software II: PHIST

a Pipelined Hybrid-parallel Iterative Solver Toolkit

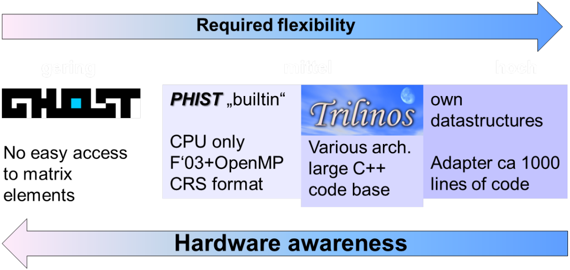
- facilitate algorithm development using GHOST
- holistic performance engineering
- portability and interoperability



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)



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:

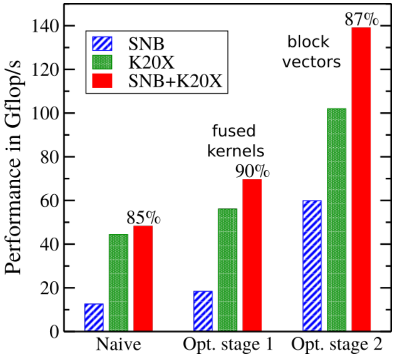
print achieved roofline performance of kernels after complete run to reveal

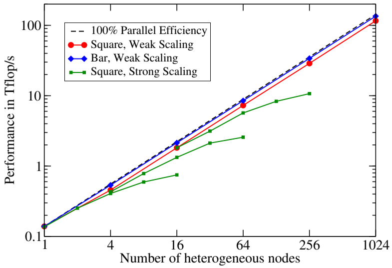
- deficiencies of kernel lib
- implemntation issues of algorithm (strided data access etc.)

Special-purpose operations

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

Sparse matrix-vector multiplication (in a Chebyshev solver)

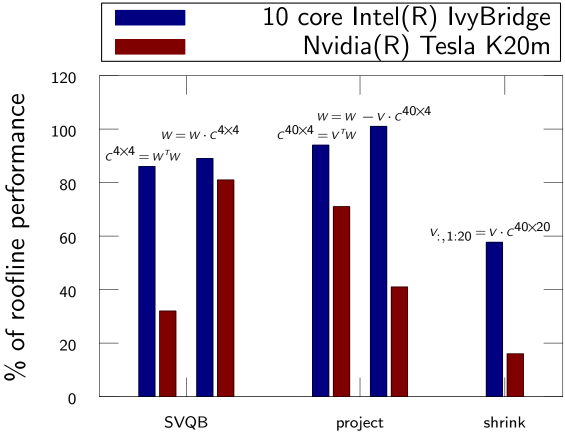




Weak and strong scaling
(on Piz Daint @ CSCS Lugano)

from: Kreutzer *et al.* IPDPS'15
SELL-C-σ sparse matrix storage format for heterogenous systems

'Tall & skinny' kernel performance ($V \in \mathbb{R}^{10M \times 40}$, $W \in \mathbb{R}^{10M \times 4}$)



⇒ some fallback kernels needed on GPU, further experiments postponed

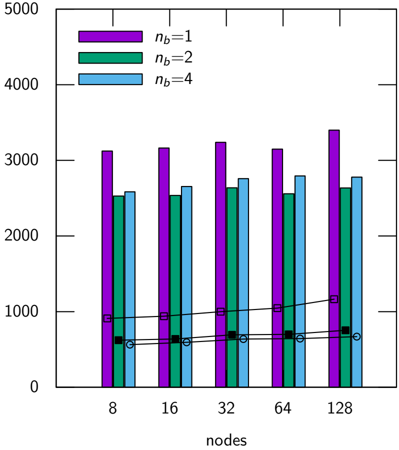
Strong scaling performance

Setup

- non-symmetric matrix from 7-point 3D PDE discretization ($n \approx 1.3 \cdot 10^8$, $n_{nz} \approx 9.4 \cdot 10^8$)
- find 20 eigenvalues
- Ivy Bridge Cluster

Results

- $n_b = 2$: significantly faster
- $n_b = 4$: no further improvement



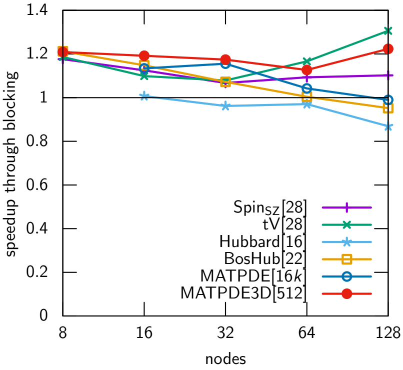
Block method faster for various matrices

Setup

- different large matrices from
  - Quantum physics
  - PDE discretization
- find 20 outmost eigenvalues using (block) Jacobi-Davidson
- block size $n_b = 2$ (similar for 4)

Results

- typically faster by a factor 1.2
- less synchronization but larger messages during spMMVM



Further information

GHOST and **PHIST** are developed within the DFG (SPPEXA) funded project ESSEX (Equipping Sparse Solvers for the EXa-scale).

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

We are happy to collaborate on
building blocks, algorithms and applications
and support 'friendly users'!

Contact: Jonas.Thies@DLR.de