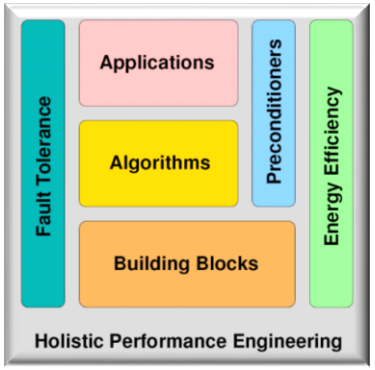


ESSEX – Equipping Sparse Solvers for Exascale

A. Alvermann[1], A. Basermann[2], H. Fehske[1], M. Galgon[3], G. Hager[4], M. Kreutzer[4], L. Krämer[3], B. Lang[3], A. Pieper[1], M. Röhrig-Zöllner[2], F. Shahzad[4], J. Thies[2], and Gerhard Wellein[4]

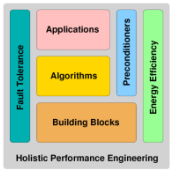
[1]Institute for Physics, Univ. Greifswald

[2]Simulation & SW Technology, German Aerospace Center

[3]Applied Computer Science Univ. Wuppertal

[4]Dept. Computer Science & RRZE Univ. Erlangen-Nürnberg

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)

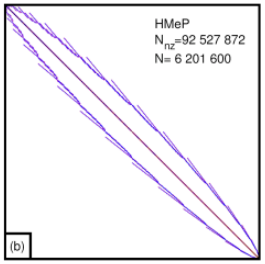
FT concepts, programming for extreme parallelism

Sparse eigensolvers, preconditioners, spectral methods

Quantum physics / chemistry

ESSEX applications: Graphene, topological insulators, …

Physical properties and sparse eigenvalue problem

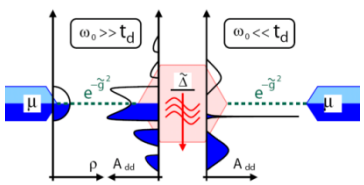


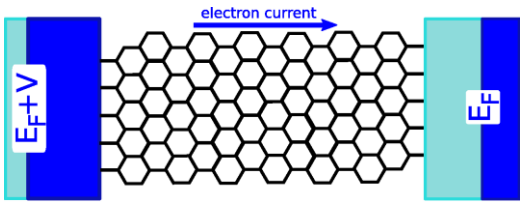
Solve eigenvalue problem

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

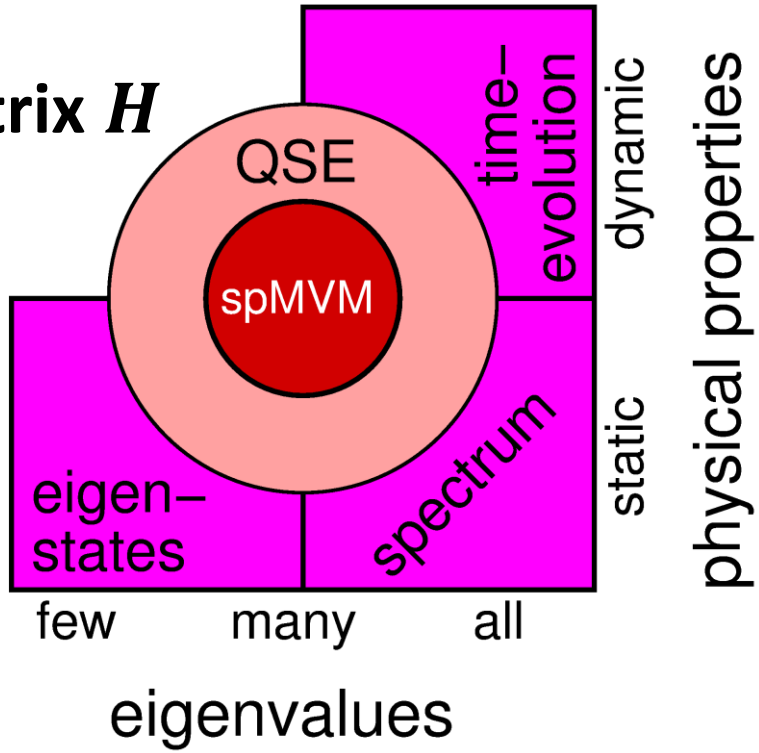
Large & sparse matrix H

Matrix H defined by quantum physics/information applications



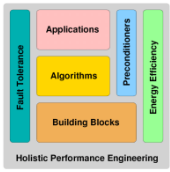


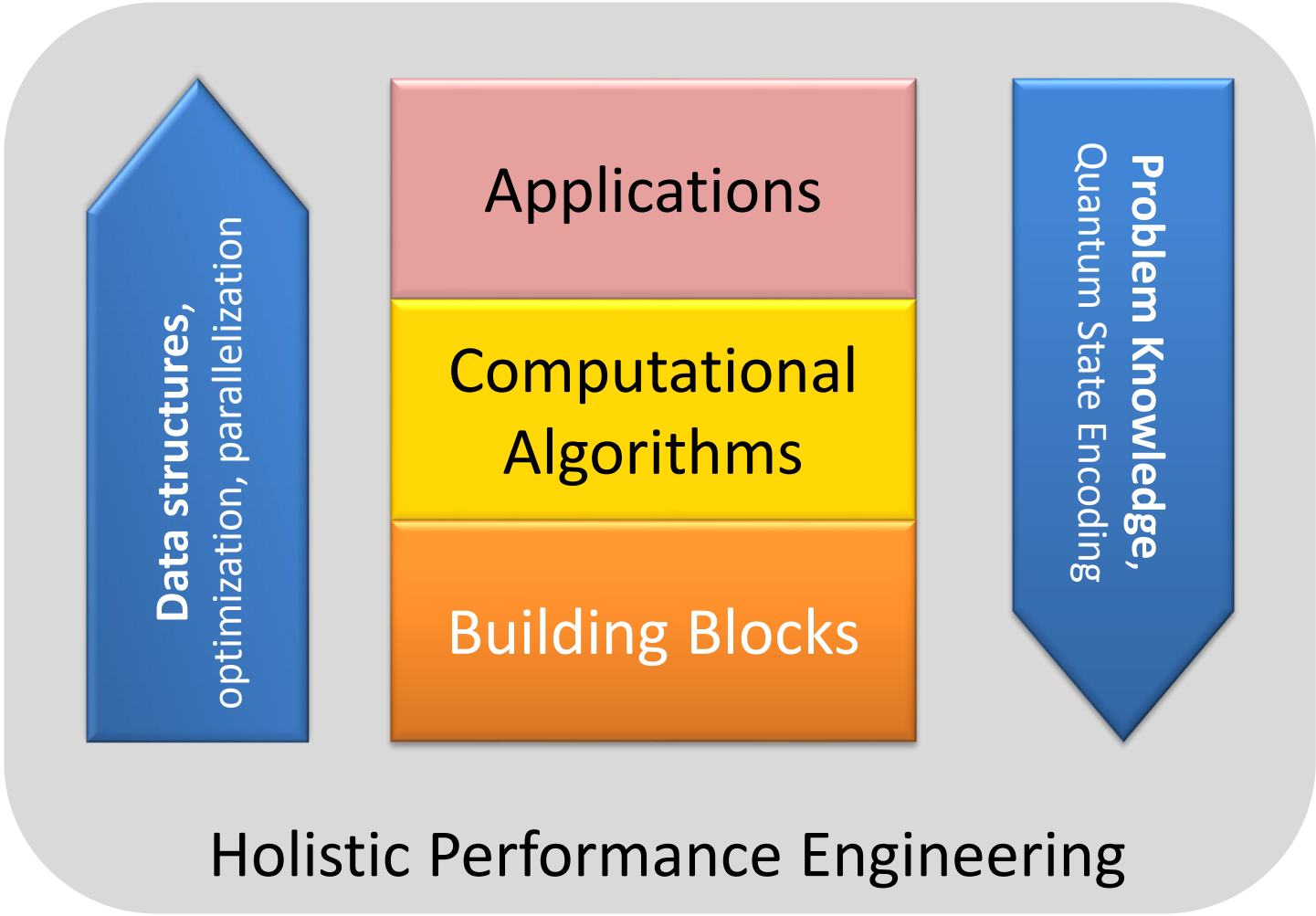
$$i\hbar\frac{\partial}{\partial t}\psi(r,t) = H\psi(r,t)$$

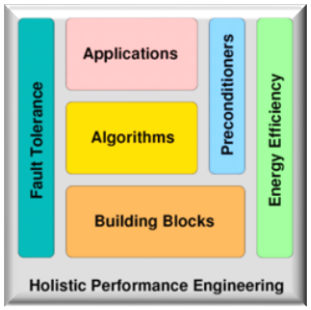


(λ_i, x_i)

Enabling Exascale through software codesign



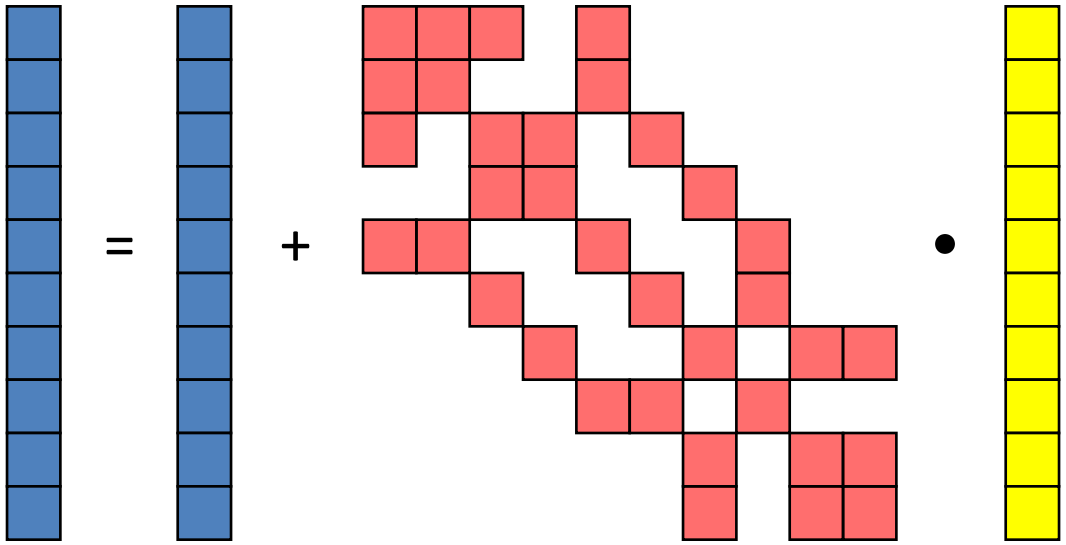




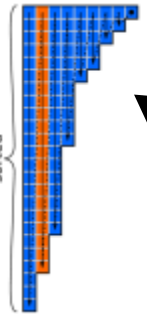
BASIC BUILDING BLOCKS: SPARSE MATRIX VECTOR MULTIPLICATION

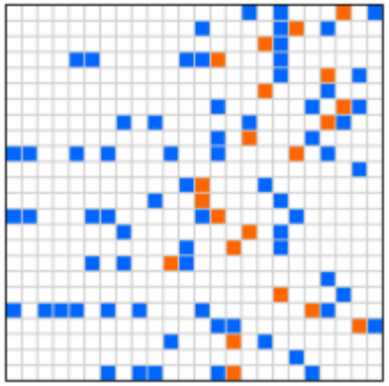
1) A UNIFIED STORAGE FORMAT

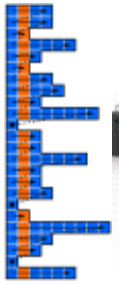
2) BLOCK OPERATIONS

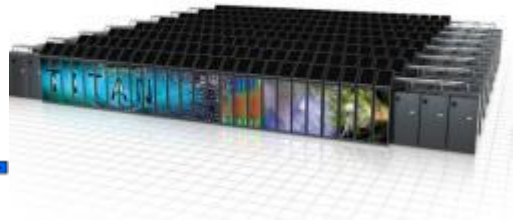


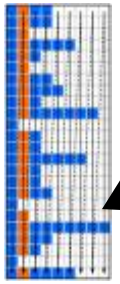
Sparse Matrix Storage Formats – A jungle





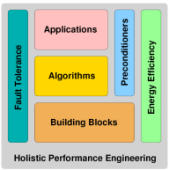




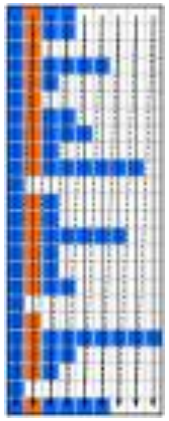


?

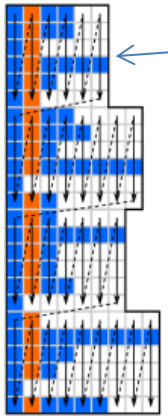
SELL-C- σ: a unified Sparse Matrix Storage Formats



ELLPACK[1]

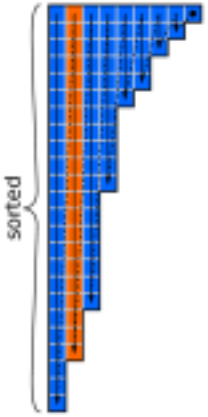


SELLPACK[2]

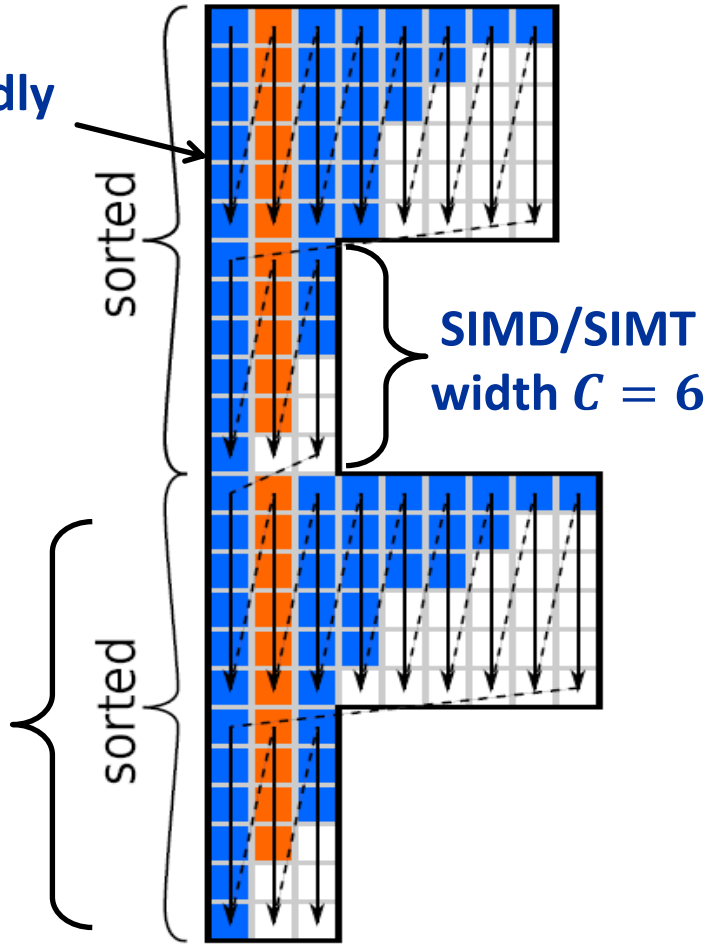


SIMD/SIMT friendly data chunks

JDS



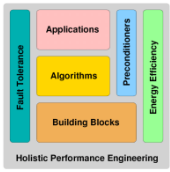
Local sorting scope
$\sigma = 12$ (σ =2 C)



[1]N. Bell and M. Garland. Implementing sparse matrix-vector multiplication on throughput-oriented processors. SC '09

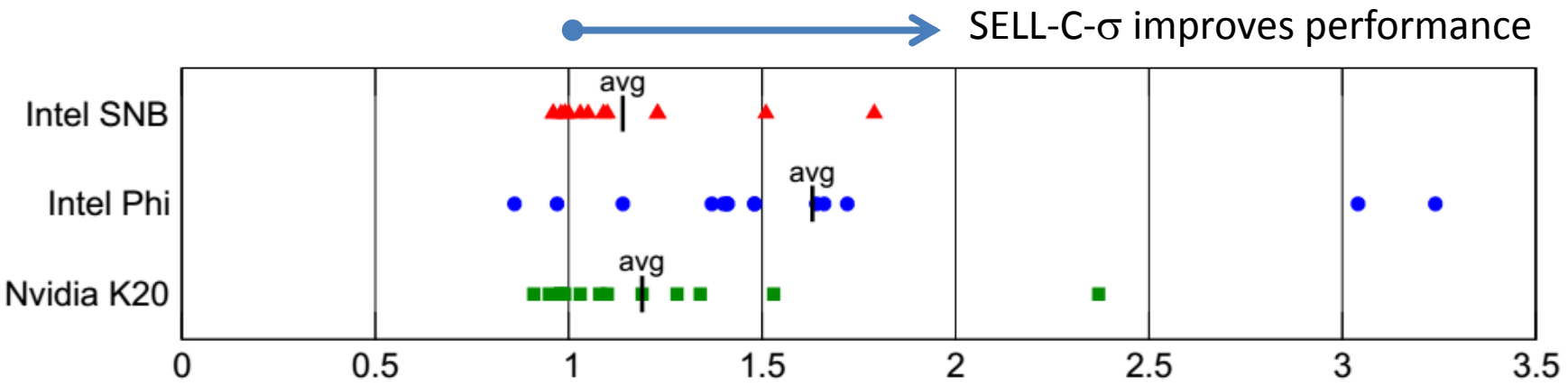
[2]A. Monakov et al.. Automatically tuning sparse matrix-vector multiplication for GPU architectures. LNCS Vol. 5952

SELL-C- σ : Relative Performance



Relative performance as compared to optimized vendor library

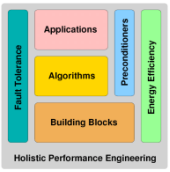
- Intel Sandy Bridge 8-core: Intel mkl (CRS)
- Intel Xeon Phi: Intel mkl (CRS)
- Nvidia K20: Nvidia CUBLAS



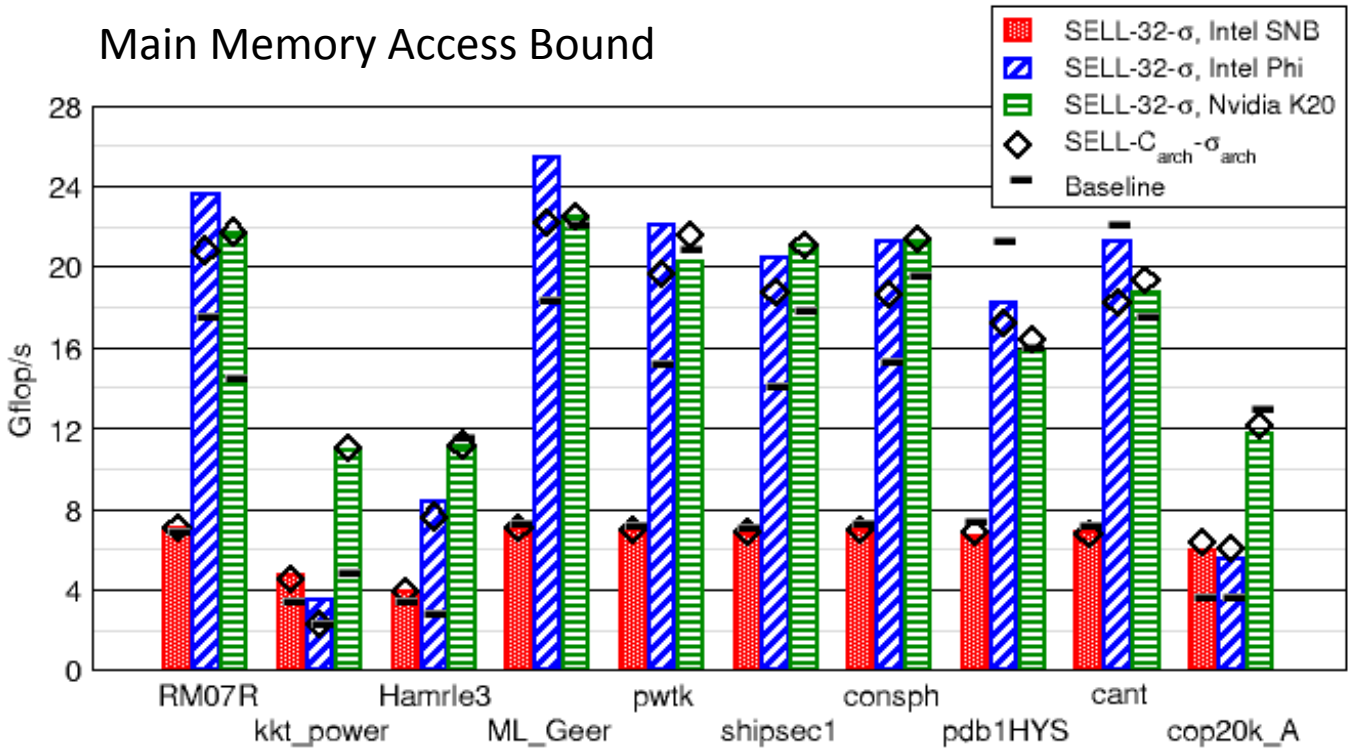
Test set: “Williams group" - University of Florida Sparse Matrix Collection (www.nvidia.com/content/NV_Research/matrices.zip)

Accuracy: Double Precision Computations

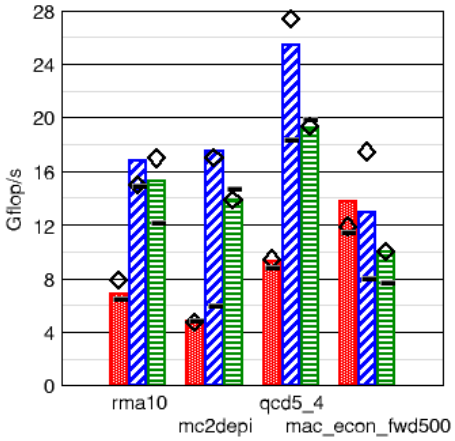
SELL-C- σ : Absolute Performance



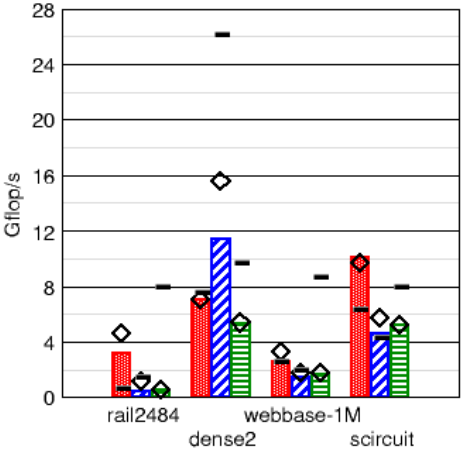
Main Memory Access Bound



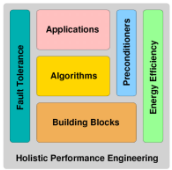
Cache Bound



Strange structures



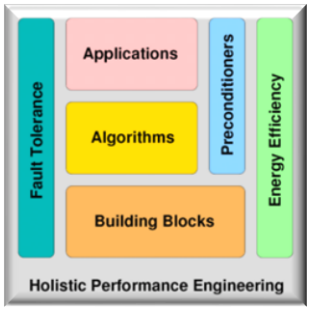
SELL-C- σ: a unified Sparse Matrix Storage Formats



- Moritz Kreutzer: ***A unified sparse matrix storage format for heterogeneous systems*** Poster & talk at the Early Research Showcase track at SC13 (Denver)
- M. Kreutzer, G. Hager, G. Wellein, H. Fehske, and A. R. Bishop: ***A unified sparse matrix data format for modern processors with wide SIMD units***. ACCEPTED for publication: SIAM SISC (arXiv:1307.6209)

- S. Müthing, D. Ribbrock and D. Göddeke**. *Integrating multi-threading and accelerators into DUNE-ISTL***. In: *Proceedings of ENUMATH 2013*. (2014). Accepted. (→ EXADUNE project; extension to small dense blocks)
- H. Anzt, S. Tomov and J. Dongarra. ***Implementing a sparse matrix vector product for the SELL-C/SELL-C-σ formats on NVIDIA GPUs***. Tech. rep., March 2014. http://www.eecs.utk.edu/resources/library/585 (--> MAGMA Library)

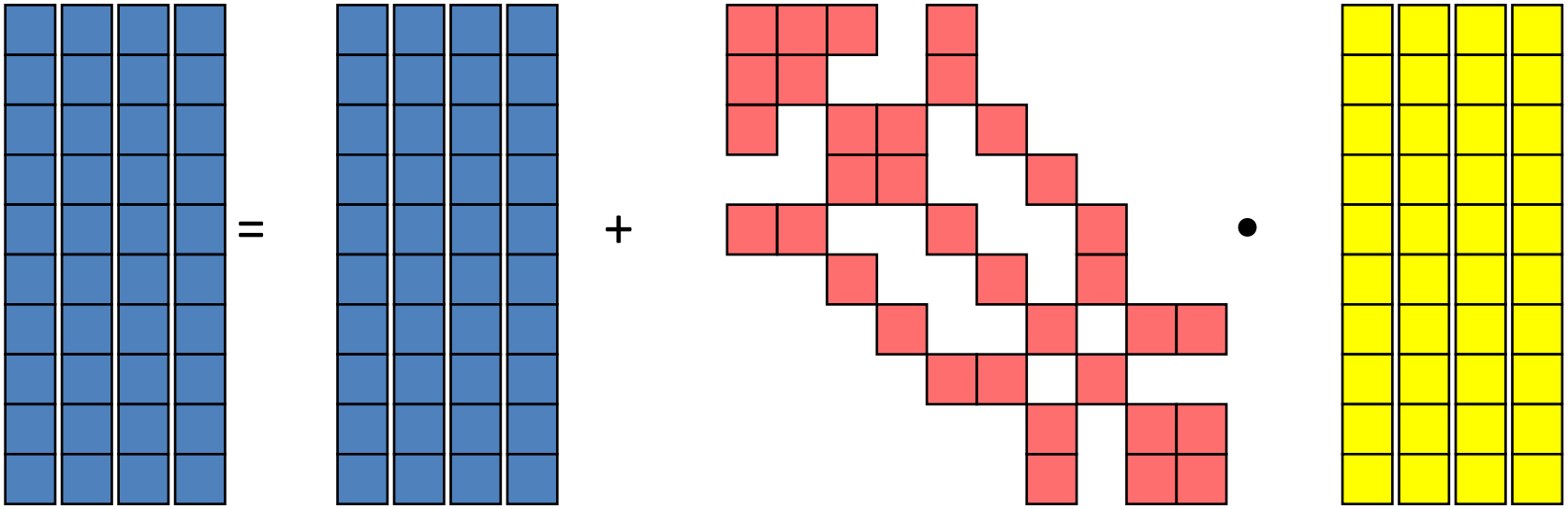
- Work in progress: PETSc, ww.libgeodecomp.org,…



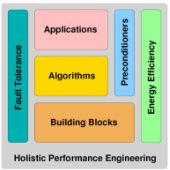
BASIC BUILDING BLOCKS: SPARSE MATRIX VECTOR MULTIPLICATION

1) A UNIFIED STORAGE FORMAT

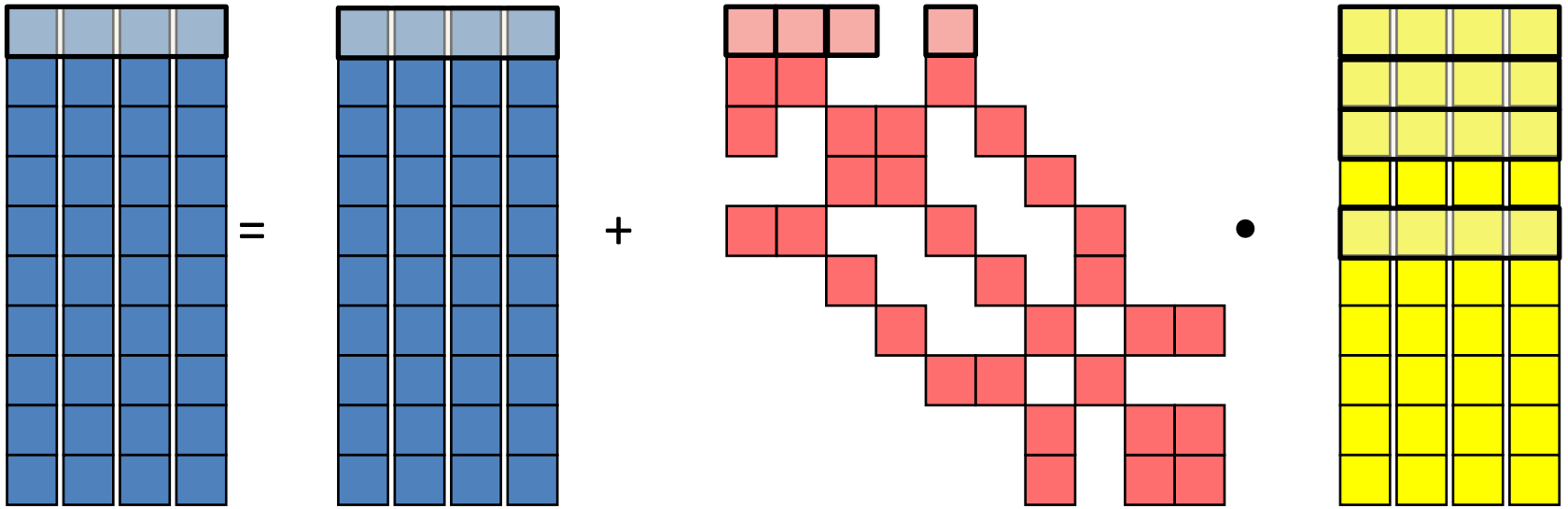
2) BLOCK VECTOR OPERATIONS



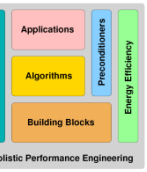
BLOCK VECTOR Operations – from scattered access towards streaming



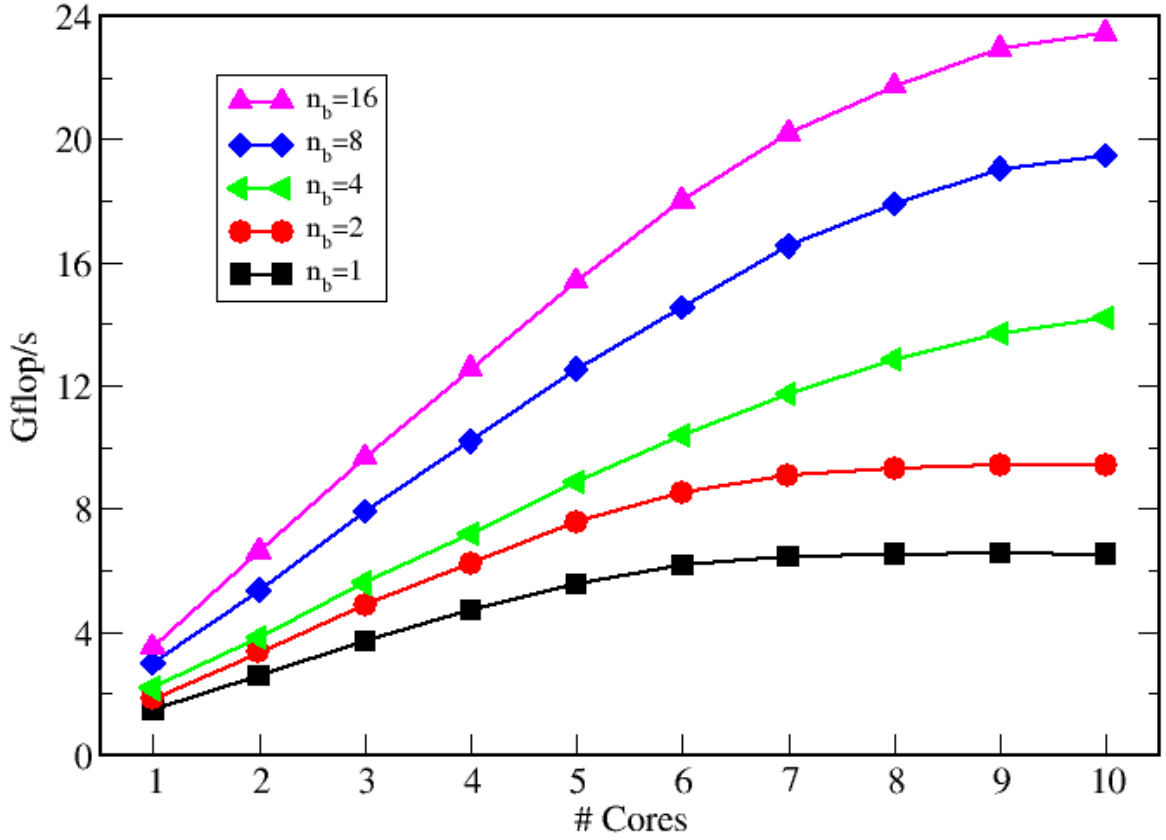
- Applying sparse matrix to several vectors at the same time:
  - Reduces overall traffic for loading sparse matrix data
  - Increases message length for parallelization
  - Row major/interleaved storage scheme for vectors: Linear (streaming) data access patterns for “larger” vector blocks (best case: multiple of 8 vectors)



BLOCK VECTOR Operations – performance impact



Performance of block spMVM using n_b = 1,2,4,8,16 vectors



Matrix properties:

- Dimension: $D=10^7$
- Avg. non-zeros/row: n_{nzr} =14

Test system:

- Intel Xeon E5-2660 v2
- Single socket – 10 cores
- Clock: 2.2 GHz
- LD-Bandwidth: 47 GB/s

Analysis of block spMVM performance: M. Röhrig-Zöllner et al., *INCREASING THE PERFORMANCE OF THE JACOBI-DAVIDSON METHOD BY BLOCKING*, submitted.

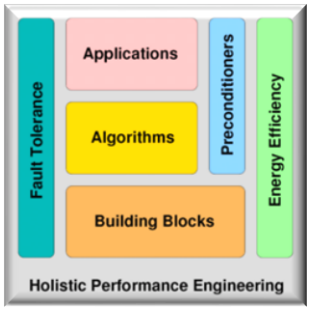
Basic building blocks library: GHOST

General, Hybrid and Optimized Sparse Toolkit

- Basic **tailored** sparse matrix / vector **operations**
- CRS or **SELL-C-σ** (unified format) storage schemes
- (Block-)SpMVM: **SIMD intrinsic** (AVX, SSE, MIC) & **CUDA kernels**
- Dense vector /matrices: row-/column-major storage

- Supports data & task parallelism (up to application level)
- MPI + OpenMP + tasks for concurrent execution
- Generic and hardware-aware (w/ hwloc) task management

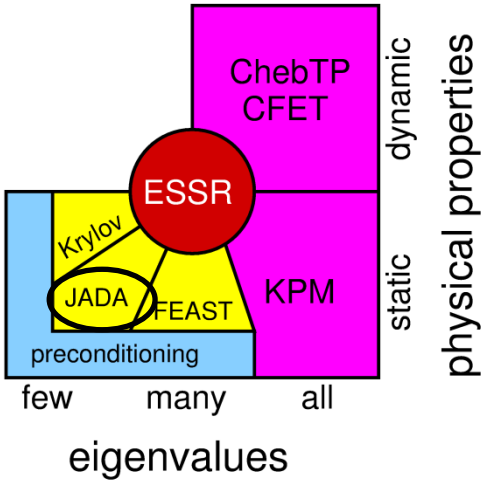
- Application layer triggered checkpoint / restart
- Asynchronous checkpointing via tasks
- Various checkpoint locations (node, filesystem)



ALGORITHMS: BLOCKED JACOBI-DAVIDSON (JADA) METHOD

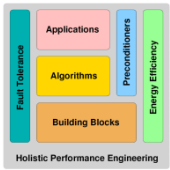
Compute l extreme eigenvalues/vectors$\{(\lambda_i, v_i), i = 1, \ldots, l\}$ of sparse matrix A:

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



Analysis of block spMVM performance: M. Röhrig-Zöllner et al., *INCREASING THE PERFORMANCE OF THE JACOBI-DAVIDSON METHOD BY BLOCKING*, submitted.

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):

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

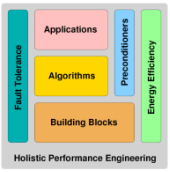
Scalar

Dense matrix (Tall & skiny)

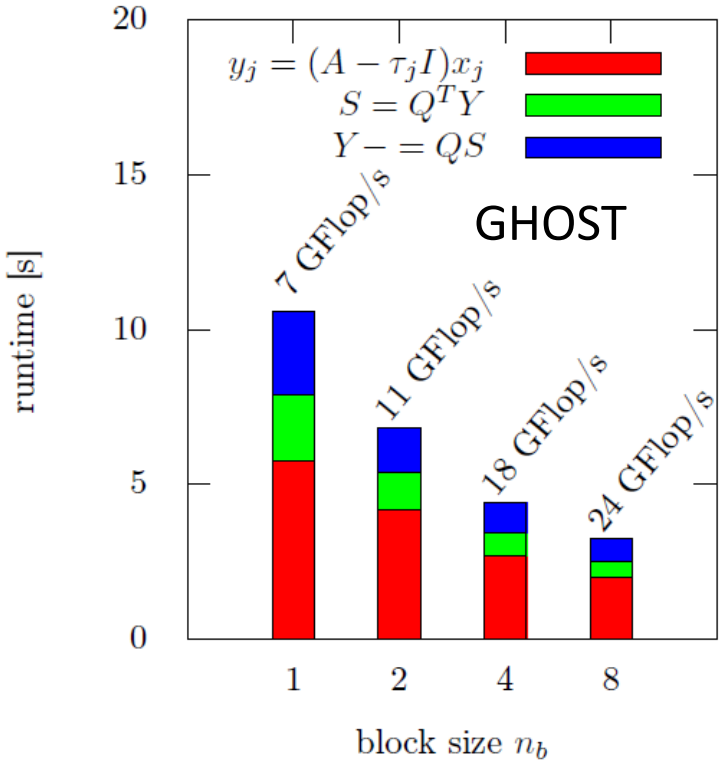
Sparse Matrix

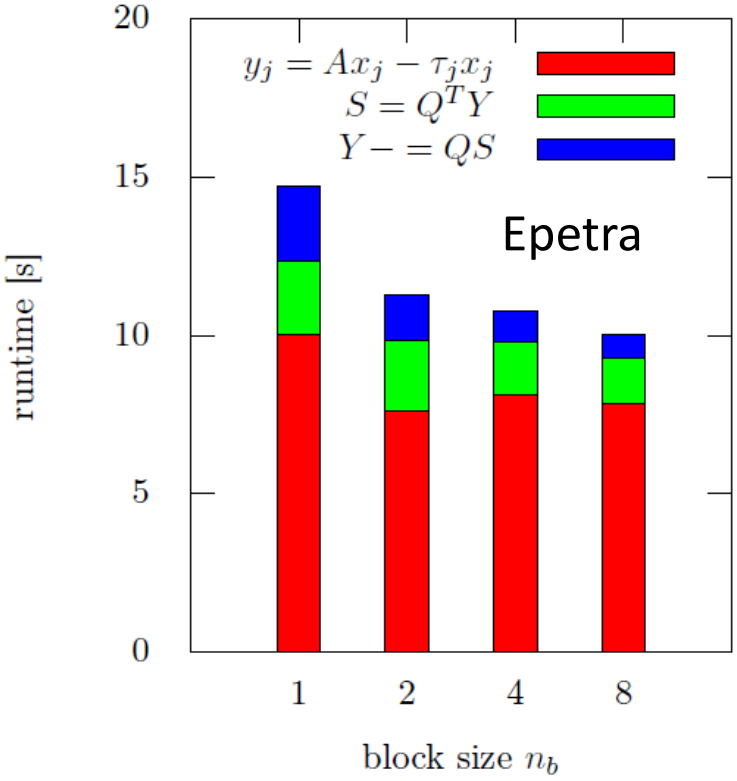
BLOCKED JADA operation available in GHOST for CPU - GPGPU & Xeon Phi: work in progress.

BLOCKED JADA – performance of basic operation



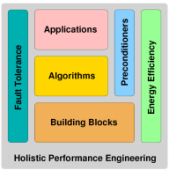
- Matrix: $D=10^7$; $n_{nzr}=14$ – Test system: Intel Xeon E5-2660 v2
- Runtime for 120 JADA operations





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

BLOCKED JADA – overall runtime

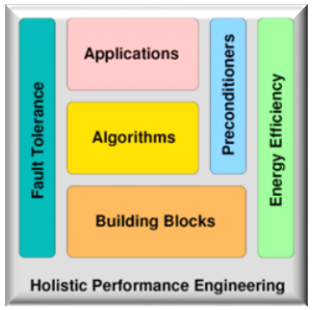


- Determine 20 lowest eigenvalues/vectors
- Compare with latest version of PRIMME[1] (optimized for symmetric problems) + basic operations from Epetra

method	n_b	matvecs	walltime [s]	time/spMVM [ms]
PRIMME (a)	1	1374	183	49.7* (37.3%)
	2	1569	203	50.6 (39.1%)
	4	1899	236	49.9 (40.2%)
	8	2323	286	49.7 (40.4%)
PRIMME (b)	1	1377	180	49.6 (37.9%)
	2	1553	202	50.8 (39.0%)
	4	1680	210	50.2 (40.1%)
	8	1989	260	50.3 (38.4%)
ESSEX – basic algorithm	1	1426	228	39.2** (25%)
	2	≤1591	174	22.1 (20%)
	4	≤1887	172	15.2 (17%)
	8	≤2463	219	11.9 (13%)

Still uses solver for general non-symmetric problems

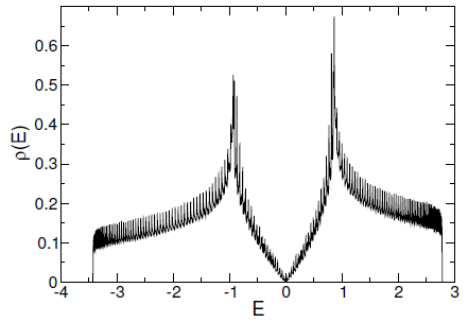
[1] A. Stathopoulos and J. R. McCombs. 2010. PRIMME: preconditioned iterative multimethod eigensolver—methods and software description. *ACM TOMS.* 37, 2, Article 21 (April 2010)

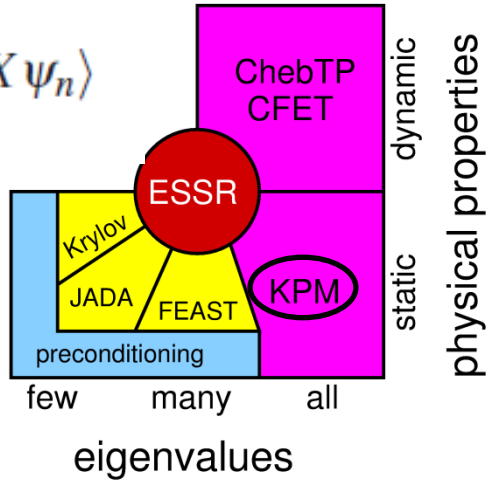


FROM APPLICATION DOWN TO BUILDING BLOCKS: KERNEL POLYNOMIAL METHOD (KPM) FOR DOS COMPUTATIONS

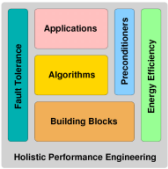
Compute approximation to the complete eigenvalue spectrum of large sparse matrix A (with $X = I$)

$$X(\omega) = \frac{1}{N}\mathrm{tr}[\delta(\omega - H)X] = \frac{1}{N}\sum_{n=1}^{N}\delta(\omega - E_n)\langle\psi_n, X\psi_n\rangle$$





KPM to approximate density of states



- Basic idea: Compute polynomial approximation to $X(\omega)$ using Chebyshev polynomials of first kind:

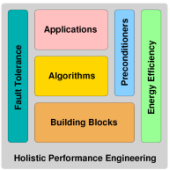
$$|v_0\rangle = |v\rangle \;, |v_1\rangle = \tilde{H}|v_0\rangle \;, |v_{m+1}\rangle = 2\tilde{H}|v_m\rangle - |v_{m-1}\rangle$$

- Sparse matrix $\tilde{H}$ is rescaled to eigenvalues $[-1, 1]$
- The complete spectrum can be approximated via the moments:

$$\mu_m = \langle v|T_m(\tilde{H})X|v\rangle = \langle v_m|X|v_0\rangle$$

- Even for large matrix dimensions M=10^3 ,…, 10^4 moments are sufficient
- Averaging over R random initial configurations is required

KPM basic implementation




```
for r = 0 to R-1 do
    |v⟩ = |rand()⟩;
    Initialization &
    computation of μ0, μ1
    for m = 1 to M/2 do
        swap(|w⟩, |v⟩);
        |u⟩ = H|v⟩ ;
        |u⟩ = |u⟩ – b|v⟩ ;
        |w⟩ = –|w⟩ ;
        |w⟩ = |w⟩ + 2a|u⟩ ;
        η2m = ⟨v|v⟩ ;
        η2m+1 = ⟨w|v⟩ ;
    end
end
```

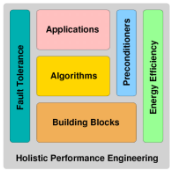
Application: R random configurations

Algorithm: Compute Chebyshev moments

Basic building blocks: spMVM and sparseBLAS1

KPM approach can be implemented with only one global communication step

KPM with augmented spMVM routine

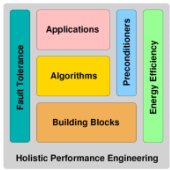


- All sparseBLAS1 operations can be done in spMVM when data is available at CPU

for $r = 0$ *to* R-1 **do**
  $|v\rangle = |\text{rand}()\rangle$;
  $|w\rangle = a(H-b)|v\rangle$ & $\mu_0 = \langle v|v\rangle$ & $\mu_1 = \langle w|v\rangle$;
  for $m = 1$ to $M/2$ **do**
    swap($|w\rangle$, $|v\rangle$);
    $|w\rangle = 2a(H-b)|v\rangle - |w\rangle$ & $\eta_{2m} = \langle v|v\rangle$ & $\eta_{2m} = \langle w|v\rangle$
  end
end

Basic building blocks: All computation can be put in spMVM at no cost

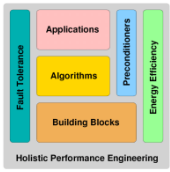
KPM with augmented block spMVM



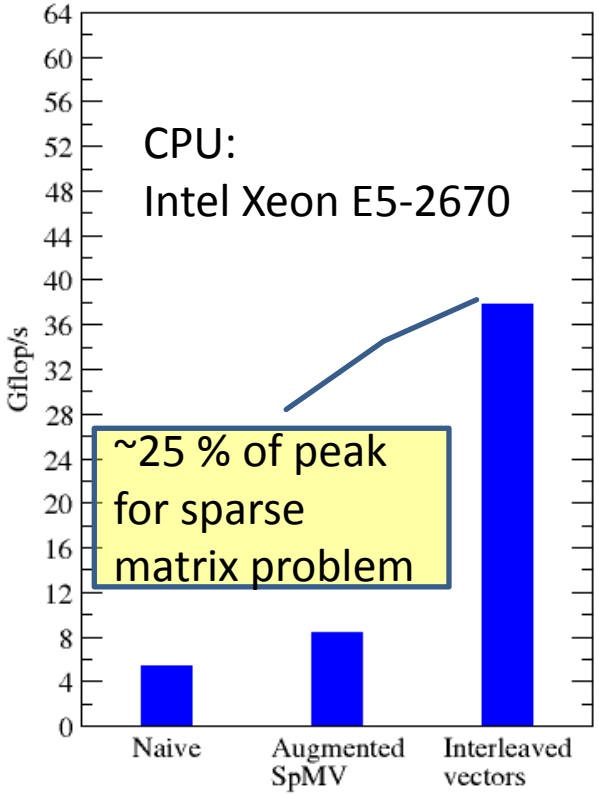
- Compute several interleaved initial configurations at the same time – vectors become tall & skinny dens matrices

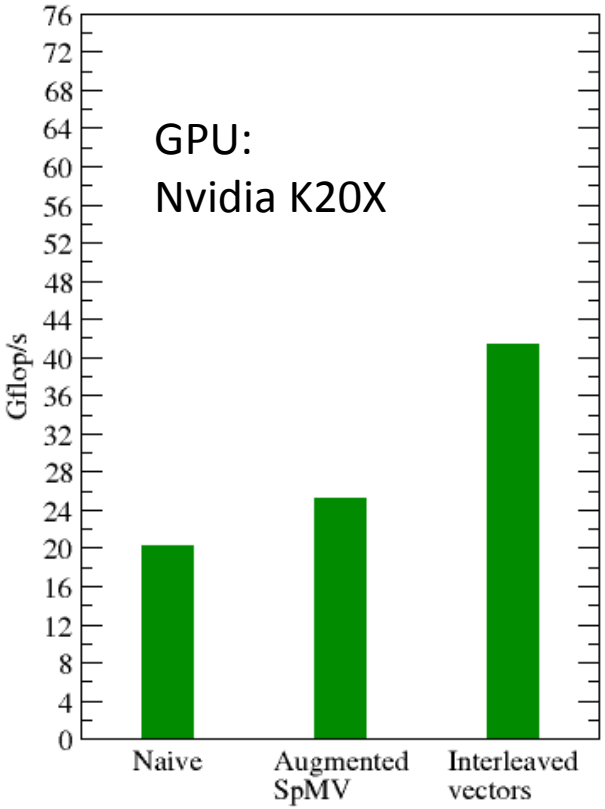
$|v_{0..R-1}\rangle = |\text{rand}()\rangle;$
$|w_{0..R-1}\rangle = a(H-b)|v_{0..R-1}\rangle$ & $\mu_{0,0..R-1}=\langle v_{0..R-1}|v_{0..R-1}\rangle$ &
$\mu_{1,0..R-1}=\langle w_{0..R-1}|v_{0..R-1}\rangle;$
for $m = 1$ to $M/2$ **do**
swap($|w_{0..R-1}\rangle, |v_{0..R-1}\rangle$);
$|w_{0..R-1}\rangle = 2a(H-b)|v_{0..R-1}\rangle - |w_{0..R-1}\rangle$ &
$\eta_{2m} = \langle v_{0..R-1}|v_{0..R-1}\rangle$ & $\eta_{2m+1} = \langle w_{0..R-1}|v_{0..R-1}\rangle$
end

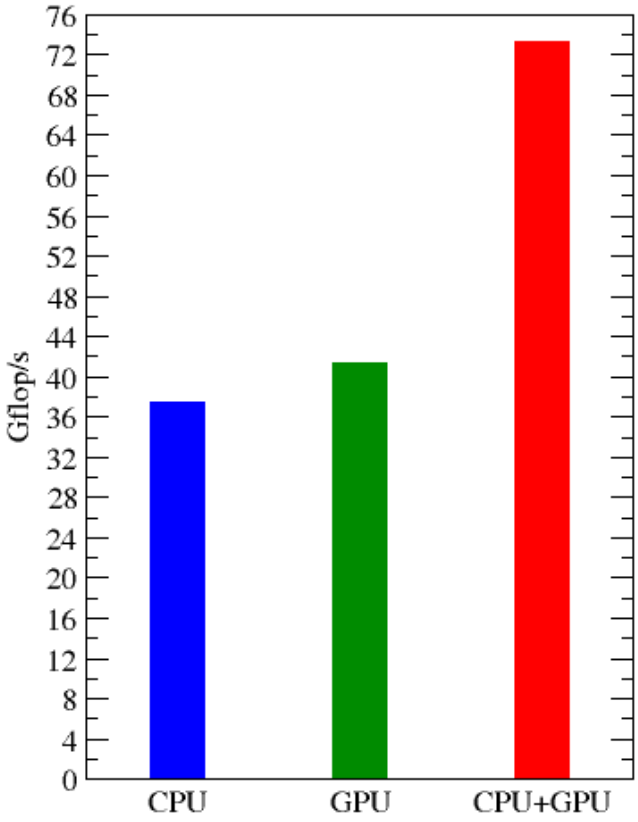
KPM performance improvements



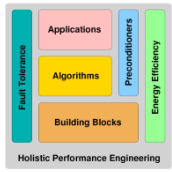
Application: Graphene matrix – $D=9*10^7$ - $R=32$ ($\leftarrow\rightarrow n_b$)
Computer: Piz Daint at CSCS Lugano – CRAY XC30 – node architecture: 1 CPU + 1 GPGPU



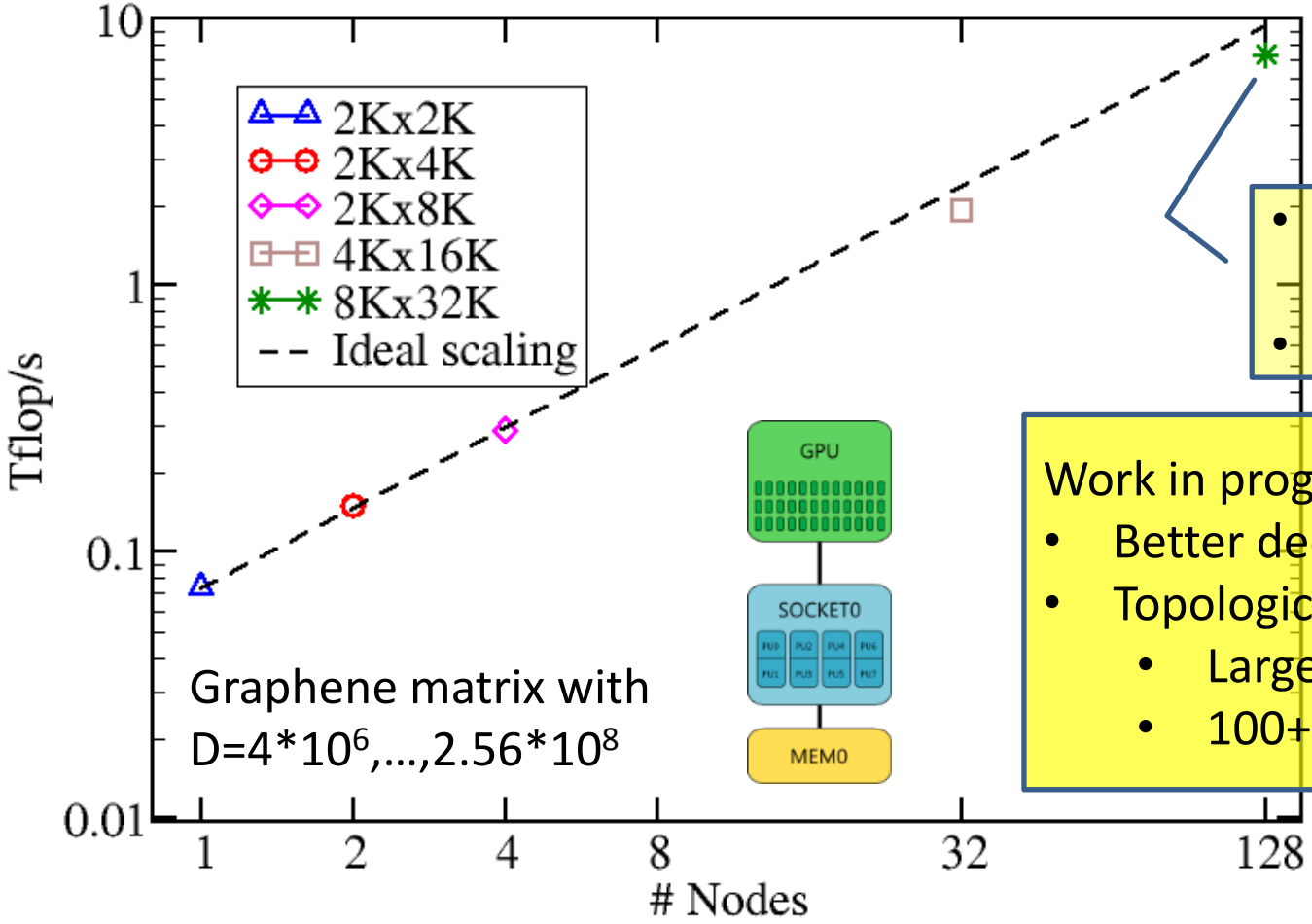




KPM: heterogeneous parallel sparse matrix scheme



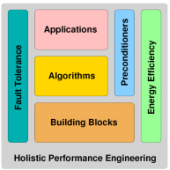
First multi-node weak scaling results from PizDaint



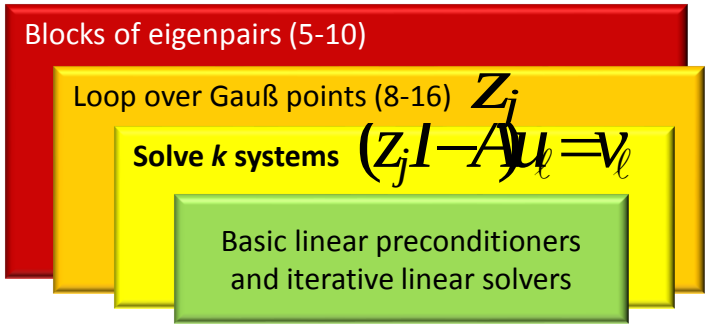
- Naïve problem decomposition
- Still small matrix size

Work in progress
- Better decomposition
- Topological insulator application:
  - Larger problems
  - 100+ GF/s single node

Selected topics not covered in this talk



- FEAST eigensolver:
  Compute blocks
  of interior eigenvalues



- Work on checkpointing/restart & Fault Tolerance (GPI miniapps)

- Application guided matrix bandwidth reduction

GHOST & JADA will become public available this year

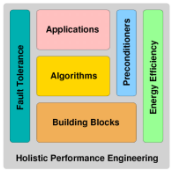
–

If you are interested in testing, you are very welcome

–

ask us: https://blogs.fau.de/essex/

Upcoming ESSEX / EXAHD workshop



Stiftung Alfried Krupp Kolleg Greifswald

Alfried Krupp Wissenschaftskolleg Greifswald

Workshop on

"Sparse Solvers for Exascale: From Building Blocks to Applications"

Alfried Krupp Wissenschaftskolleg Greifswald

22. – 26.03.2015, Greifswald, Germany

Organizer: G. Wellein, H. Fehske, H.-J. Bungartz

Invited speakers:

E. Chow (Georgia Tech)

M. Hochbruck (Karlsruhe)

S. Matsuoka (Tokyo)

Y. Saad (Minnesota)

H. D. Simon (Berkeley)