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 II: 2016 – 2018
• Motivation

• Software:
  – Interoperability, portability & performance

• Multicoloring and ILU Preconditioning

• Scaling Results: Eigenvalue Computations
ESSEX project – background

Quantum physics/information applications

\[ \psi(\vec{r}, t) = H \psi(\vec{r}, t) \]
and beyond....

\[ H \, x = \lambda \, x \]

“Few” (1,...,100s) of eigenpairs

\[ \{ \lambda_1, \lambda_2, \ldots, \ldots, \ldots, \lambda_k, \ldots, \ldots, \lambda_{n-1}, \lambda_n \} \]

“Bulk” (100s,...,1000s) eigenpairs

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

\[ \rightarrow \text{Sparse eigenvalue solvers of broad applicability} \]
Software: Interoperability, portability & performance

Kernel library (GHOST) and solver framework (PHIST)
ESSEX-II: Software Packages

**User Applications**
- CRAFT: C/R & Automatic Fault Tolerance lib
- GHOST: General, Hybrid, and Optimized Sparse Toolkit
- PHIST: Pipelined Hybrid Parallel Iterative Solver Toolkit

**3rd-party libraries:** Trilinos,...

**Hardware:** CPU / GPGPU / Xeon Phi

**ScaMaC**
Scalable Matrix Collection

**RACE**
Recursive Adaptive Coloring Engine

**MPI+X**  \( X \in \{\text{CUDA},\text{OpenMP},\text{pthreads}\} \)

Links to open source repositories at [https://blogs.fau.de/essex/code](https://blogs.fau.de/essex/code)
• **Hybrid MPI+X** execution mode (X=OpenMP, CUDA)

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

• **Tall & skinny matrix-matrix kernels** (block orthogonalization)  
  \(\rightarrow 2x – 10x\) speed-up vs. Optimized general building block libraries

• **SELL-C-\(\sigma\) sparse matrix format**

• Open Source code & example applications: [https://bitbucket.org/essex/ghost](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](https://bitbucket.org/essex/[ghost,phist])
- Cmake build system
- Available via Spack
- [https://github.com/spack/spack/](https://github.com/spack/spack/)
- PHIST will join **Extreme-Scale Development Kit**, [https://xSDK.info/](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 \( \rightarrow \) 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 Preconditioning

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)

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)

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

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

**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

BEAST and Z-PARES: shared tools for large EVPs

- BEAST
  - multiple flavors
  - iterative adaptivity
  - ESSEX libraries

- Z-PARES
  - +SSM
  - contour integrals
  - SSM algorithm
  - nonlinear EVPs

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

≈ 880 TFlops on OFP
Visit our homepage: https://blogs.fau.de/essex/

THANK YOU!