Parallel Hierachical Hybrid Solvers for Petascale and Beyond

Esmond G. Ng
Lawrence Berkeley National Laboratory

May 23, 2011

Background

- Systems of linear equations

  $$A\,x = b$$

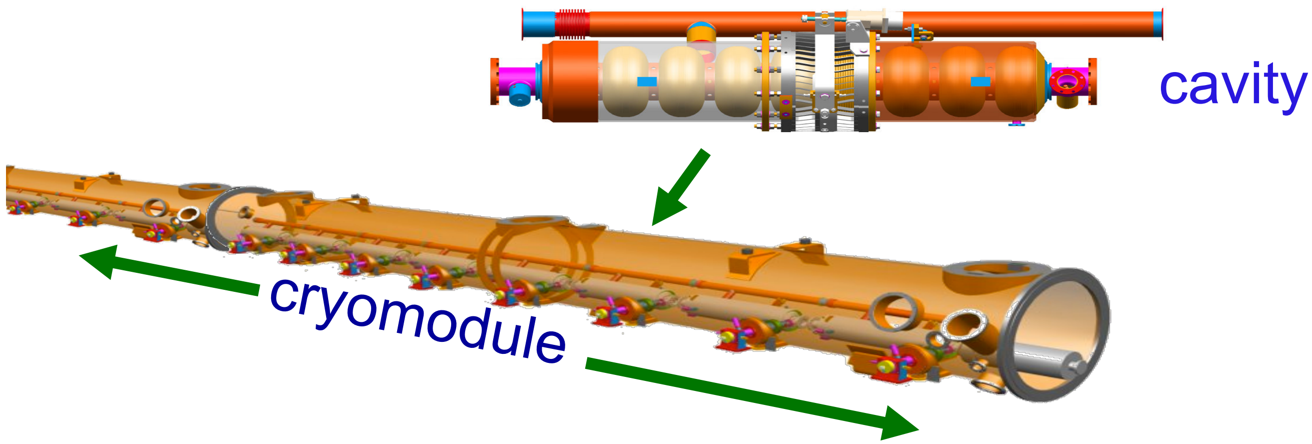
  arise in many scientific and engineering applications.
  - A is a given n by n matrix and b is a given vector.
  - The goal is to compute x.

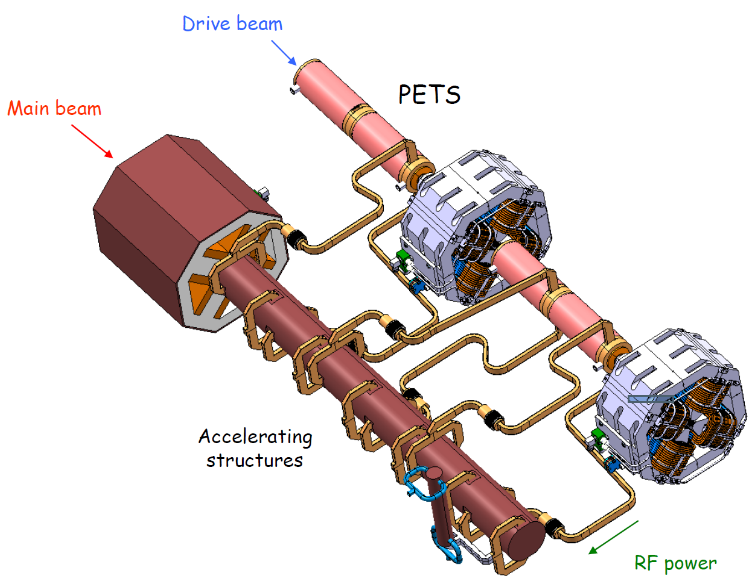
- Occur most often in the inner most loop of numerical simulations
- More complex/accurate simulations require efficient solution of continuously increasing larger problems.

Need for Accurate and Robust Linear Solvers

- Some applications need to compute accurate solutions efficiently.
- Example: modeling of accelerator cavities

International **L**inear **C**ollider (ILC)



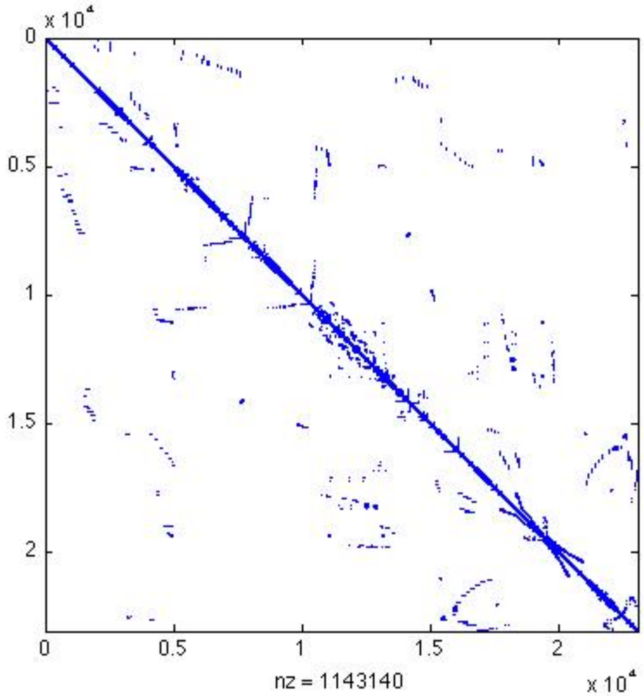


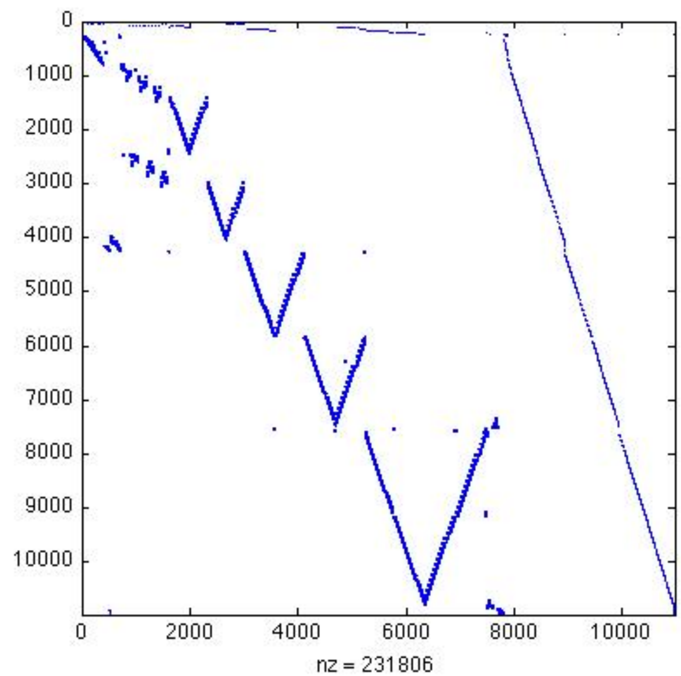
Compact **Li**near **C**ollider (CLIC) -
Two-Beam Accelerator

Background

❑ Characteristics of *A*:
  - The dimension, *n*, can be very large.
  - *A* can be sparse – many of the elements in *A* are zero.
    - “Sparsity” depends on the applications.

⇒ Absolutely important to take advantage of the zero entries for efficient solution of such linear systems.





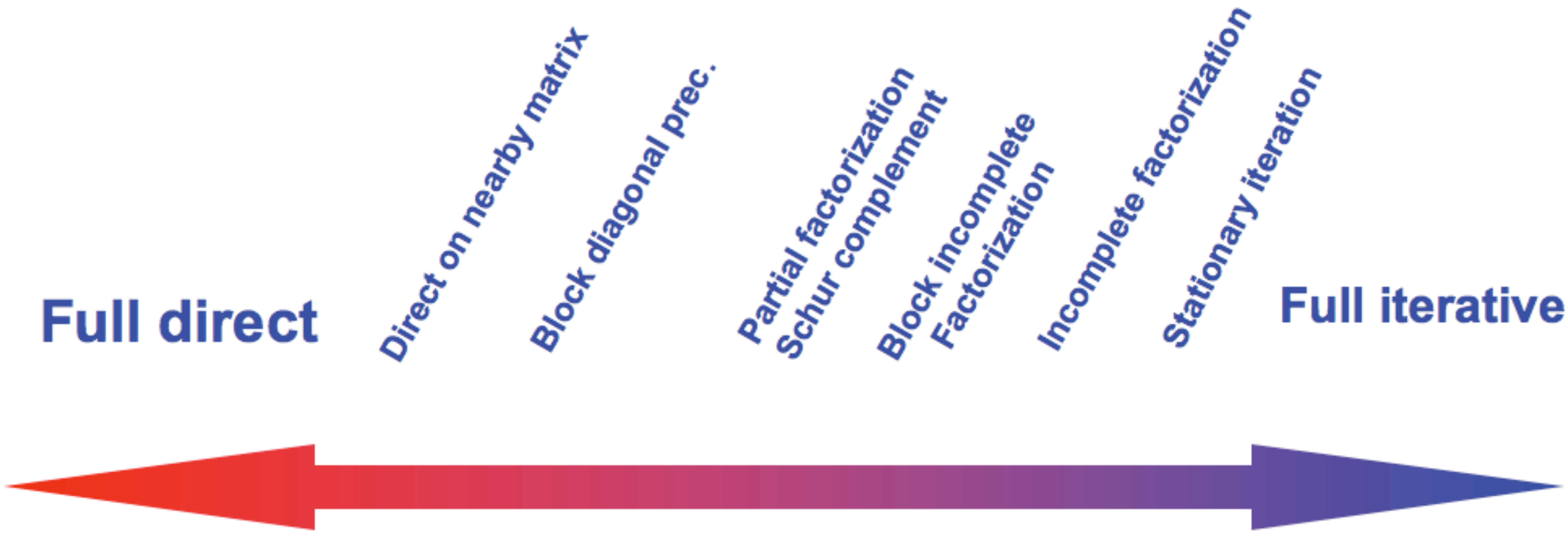
Background

- Features of emerging computer architectures –
  - Hierarchical – multi-/many-core
  - Heterogeneous – CPU + accelerators
- This provide several levels of parallelism

⇒ Need to design novel hierarchical solution schemes that naturally match the features of computing platforms

NERSC Hopper – a 1.28 petaflops/sec Cray XE6 with 6,384 compute nodes (24 cores/node, totaling 153,216 cores) and 212 TB of memory

Sparse Linear Equations Solver Spectrum



Direct Methods

- Based on factoring the matrix A into product of a lower triangular matrix L and an upper triangular matrix U using Gaussian elimination:

  $$A = LU$$

  (pivoting may be needed for stability)

- Then the solution is obtained by solving two triangular linear systems

  $Ly = b$ and $Ux = y$

  (permutation due to pivoting is not shown)

Direct Methods

- Positives:
  - Robust – termination after a finite number of operations.
  - Accurate – Gaussian elimination is known to be backward stable.
  - Efficient – almost all implementations take advantage of BLAS-3 operations.
- Negatives:
  - Sparsity issues – Gaussian elimination will destroy some of the zero entries.
    - Coping with *fill* is part of the solution process.
    - Memory/computing becomes prohibitive for 3D problems.
  - Limited scalability due to communication requirements.

Iterative Methods

- Based on generating a sequence of approximations
  - Many algorithms available for generating the approximations:
    - Basic methods:
      - Jacobi, Gauss-Seidel, successive overrelaxation
    - Projection methods:
      - Steepest descent, minimal residual
    - Krylov subspace methods:
      - Arnoldi's, generalized minimal residual, conjugate gradient, conjugate residual, biconjugate gradient, quasi-minimal residual

Iterative Methods

- Positives:
  - Relative easy to implement, requiring sparse computational kernels.
    - Possibly few operations.
    - Typically require just matrix-vector multiplications.
  - Small storage requirements.
- Negatives:
  - Convergence is not guaranteed.
  - Convergence rate may be slow.
  - Both depends on the spectral radius of the "iteration matrix".
    - Problem dependent.
  - Possible better weak scalability.

Preconditioned Iterative Methods

- Improving convergence rate:
  - Find nonsingular matrices P and Q.
  - Consider the equivalent linear system $(PAQ)(Q^{-1}x) = (Pb)$.
  - The goal is to reduce the spectral radius of PAQ.
  - P and Q are called the left and right "preconditioners", respectively.
    - P and Q should be easy to apply.

- Preconditioning is a research area of its own.
  - Some recent work makes use of techniques from sparse direct methods in constructing preconditioners.

- A step further is to combine sparse direct methods and iterative methods in a more intelligent way.

Hybrid Methods for Solving Sparse Linear Systems

- Goal:
  - Combine direct methods and iterative methods in an intelligent way to create a new class of hybrid methods for solving sparse linear systems on hierarchical, and possibly heterogeneous, high performance computer architectures.
    - Exploit the positives of both direct and iterative methods.

- Concept is not new …
  - Example:
    - Apply techniques developed for direct methods to compute an incomplete factorization.
    - Then use the incomplete factors as preconditioners for iterative methods.

- … but it is the details that make a difference.

Berkeley-INRIA Team

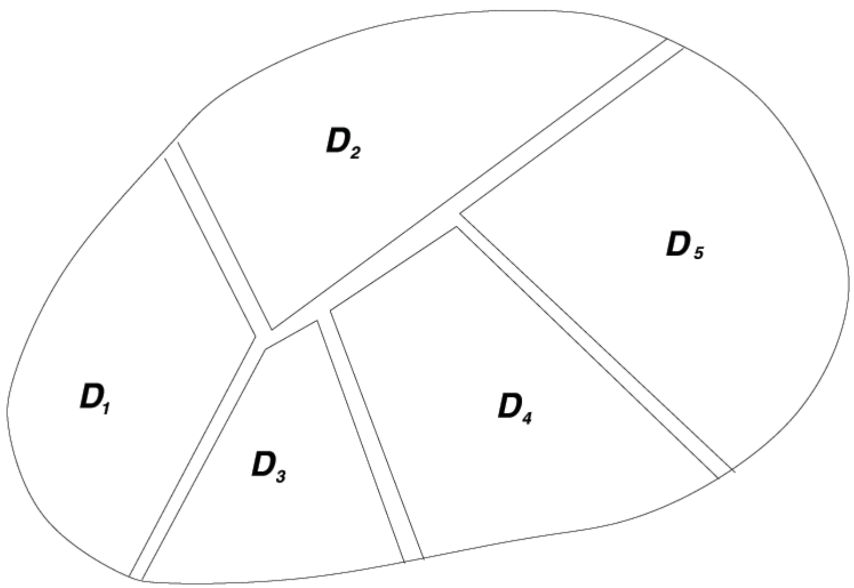
- Lawrence Berkeley National Laboratory
  - Xioaye Sherry Li and Esmond G. Ng
  - Ichitaro Yamazaki
  - Experience in sparse direct methods and incomplete factorization

- INRIA-Bordeaux
  - Luc Giraud and Jean Roman
  - Experience in iterative methods, prconditioning techniques, and sparse direct methods

- Both teams have independently engaged in R&D to create “hybrid” solvers that combine the advantages of sparse direct solvers and the advantages of iterative solvers.

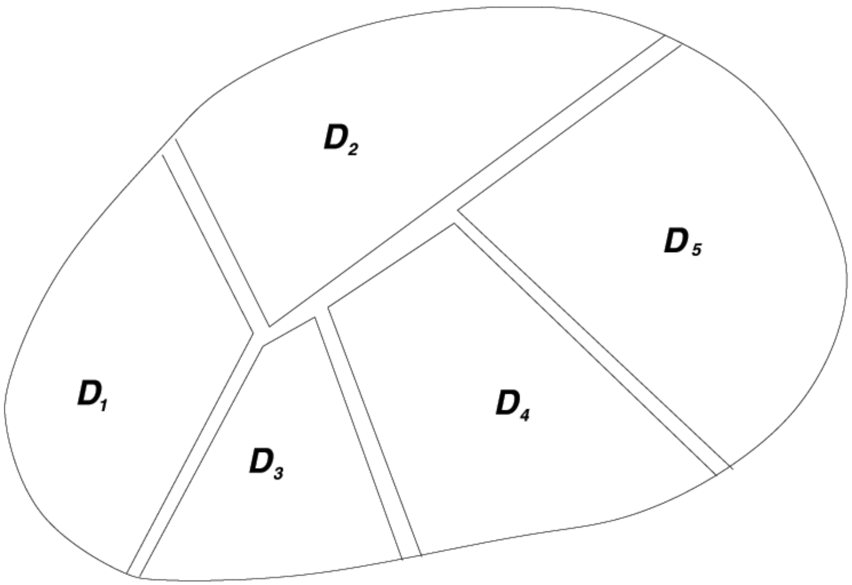
Domain Decomposition Based Hybrid Linear Solver

- Given a matrix A.
- Consider a graph representation of the sparsity of A.
  - Symmetric A … undirected graph
  - Nonsymmetric A … undirected graph of $A+A^T$
- Compute a partitioning of the graph (using, e.g., PT-SCOTCH and ParMETIS).



Domain Decomposition Based Hybrid Linear Solver

- Desirable properties of the partitioning …
  - The subdomains are balanced in size.
  - The “separator” (or “interface”) is small.



- Number the vertices of the subdomains (one by one) before those on the separator.
  - This is equivalent to a permutation of the rows and columns of A.

Bordered Block Form

$$
\begin{bmatrix}
D_1 & & & & & E_1 \\
& D_2 & & & & E_2 \\
& & D_3 & & & E_3 \\
& & & D_4 & & E_4 \\
& & & & D_5 & E_5 \\
F_1 & F_2 & F_3 & F_4 & F_5 & A_{\Gamma\Gamma}
\end{bmatrix}
\begin{bmatrix}
x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6
\end{bmatrix}
=
\begin{bmatrix}
b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6
\end{bmatrix}
$$

- Each D_i corresponds to a subdomain.
- $A_{\Gamma\Gamma}$ corresponds to the "separator".
- E_i and F_j corresponds to the connections between the subdomains and the separator.

Block Factorization

$$
\begin{bmatrix}
D_1 & & & & & E_1 \\
& D_2 & & & & E_2 \\
& & D_3 & & & E_3 \\
& & & D_4 & & E_4 \\
& & & & D_5 & E_5 \\
F_1 & F_2 & F_3 & F_4 & F_5 & A_{\Gamma\Gamma}
\end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}
=
\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{bmatrix}
$$

Block elimination

$\Rightarrow$

$$
\begin{bmatrix}
D_1 & & & & & E_1 \\
& D_2 & & & & E_2 \\
& & D_3 & & & E_3 \\
& & & D_4 & & E_4 \\
& & & & D_5 & E_5 \\
& & & & & S
\end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ y \end{bmatrix}
=
\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ c \end{bmatrix}
$$

The *Schur complement* is given by

$$S = A_{\Gamma\Gamma} - \sum_{l} F_l D_l^{-1} E_l$$

Block Triangular Solution

- The Schur complement is given by

$$S = A_{\Gamma\Gamma} - \sum_{l} F_l D_l^{-1} E_l$$

- Also, c is given by

$$c = b_{\Gamma} - \sum_{l} F_l D_l^{-1} b_l$$

$$\begin{bmatrix} D_1 & & & & & E_1 \\ & D_2 & & & & E_2 \\ & & D_3 & & & E_3 \\ & & & D_4 & & E_4 \\ & & & & D_5 & E_5 \\ & & & & & S \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ y \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ c \end{bmatrix}$$

- Once S and c have been computed, the solution can be obtained via a block substitution process.

$$Sy = c$$

$$D_l x_l = b_l - E_l y, \quad \text{for } l = 1, 2, \ldots$$

Schur Complement Method

- Assume that we can solve $Sy = c$.

- Then each (smaller) linear system

  $$D_l x_l = b_l - E_l y$$

  can be solved relatively easily.
  - Either serially on a single processor or in parallel on multiple processors.
    - This can be accomplished using a number of existing sparse direct solvers.

- How about the linear system $Sy = c$?

$$\begin{bmatrix} D_1 & & & & & E_1 \\ & D_2 & & & & E_2 \\ & & D_3 & & & E_3 \\ & & & D_4 & & E_4 \\ & & & & D_5 & E_5 \\ & & & & & S \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ y \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ c \end{bmatrix}$$

can be processed using a parallel sparse direct solver

Schur Complement Method

- How about the linear system $Sy = c$?
- Possible challenges:
  - Size of S depends on the quality of the partitioning.
    - In terms of number of unknowns associated with S and number of nonzero entries in S.

$$\begin{bmatrix} D_1 & & & & & E_1 \\ & D_2 & & & & E_2 \\ & & D_3 & & & E_3 \\ & & & D_4 & & E_4 \\ & & & & D_5 & E_5 \\ & & & & & S \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ y \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ c \end{bmatrix}$$

- This linear system can be solved in a number of ways.

Handling the Schur complement

- Possibilities for solving $Sy = c$ … $S = A_{\Gamma\Gamma} - \boxed{\sum_l F_l D_l^{-1} E_l}$

  can be done in parallel

- Performing LU on S.
  - Result in a truly direct solution for the original system.
  - S tends to be quite dense and its LU factorization usually suffers from a lot of fill.

- Perform incomplete factorization on S and use the incomplete factors as preconditioners for solving $Sy = c$.

- Compute an approximation of S via drop tolerance, perform LU factorization on the approximation, and use the LU factors as preconditioners for solving $Sy = c$.

- Compute an approximation of S via drop tolerance, perform incomplete factorization on the approximation, and use the incomplete factors as preconditioners for solving $Sy = c$.

Parallel Hierarchical Implementation

- Parallelism between sub-graphs treatment and within the treatment of each individual sub-graph (coarse grain parallelism using MPI between sub-graphs, medium/fine grain parallelism using threads on many-core multiprocessor SMP nodes)

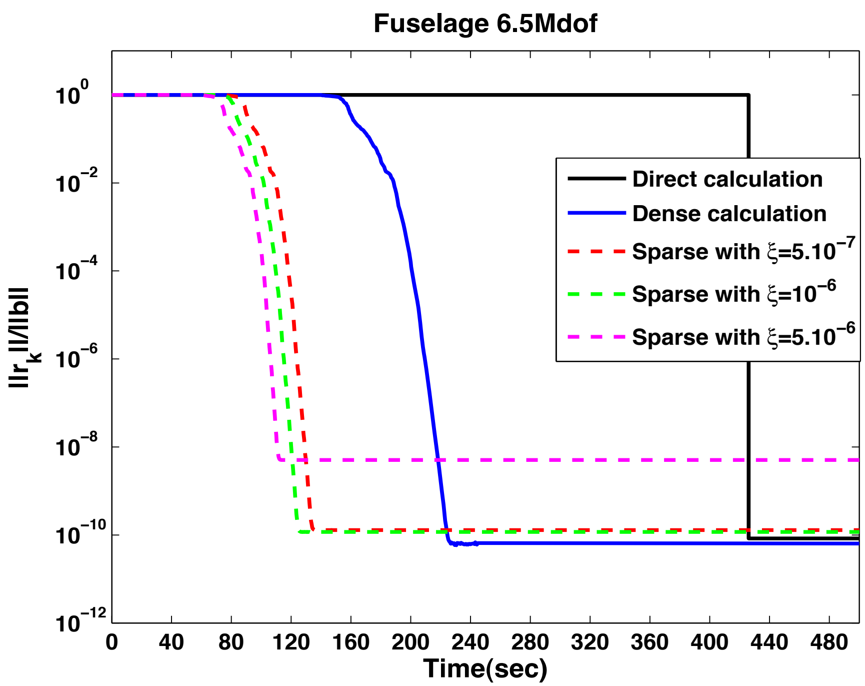
- Natural two/three-levels of parallelism with different granularity ⇨ flexibility to map on parallel platforms to best comply with architecture features

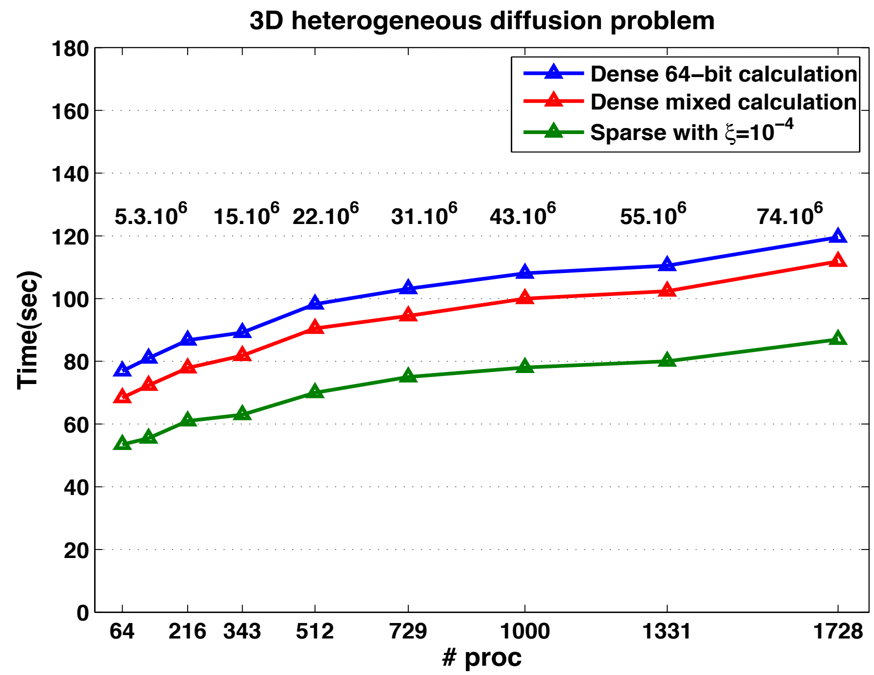
Berkeley and INRIA Approaches

- Lawrence Berkeley National Laboratory approach:
  - Use approximations to the Schur complement as preconditioners.
  - PDSLin package.
  - Funded by the TOPS (Towards Optimal Petascale Simulation) Project under the DOE SciDAC (Scientific Discovery Through Advanced Computing) Program.

- INRIA approach
  - Part of HiePACS project.
  - Parallel additive Schwarz preconditioner.
  - MaPHyS package (Massively Parallel Hybrid Solver).
  - INRIA – CERFACS Joint Laboratory on High Performance Computing (https://inria-cerfacs.inria.fr/).

MaPhyS Parallel Performance



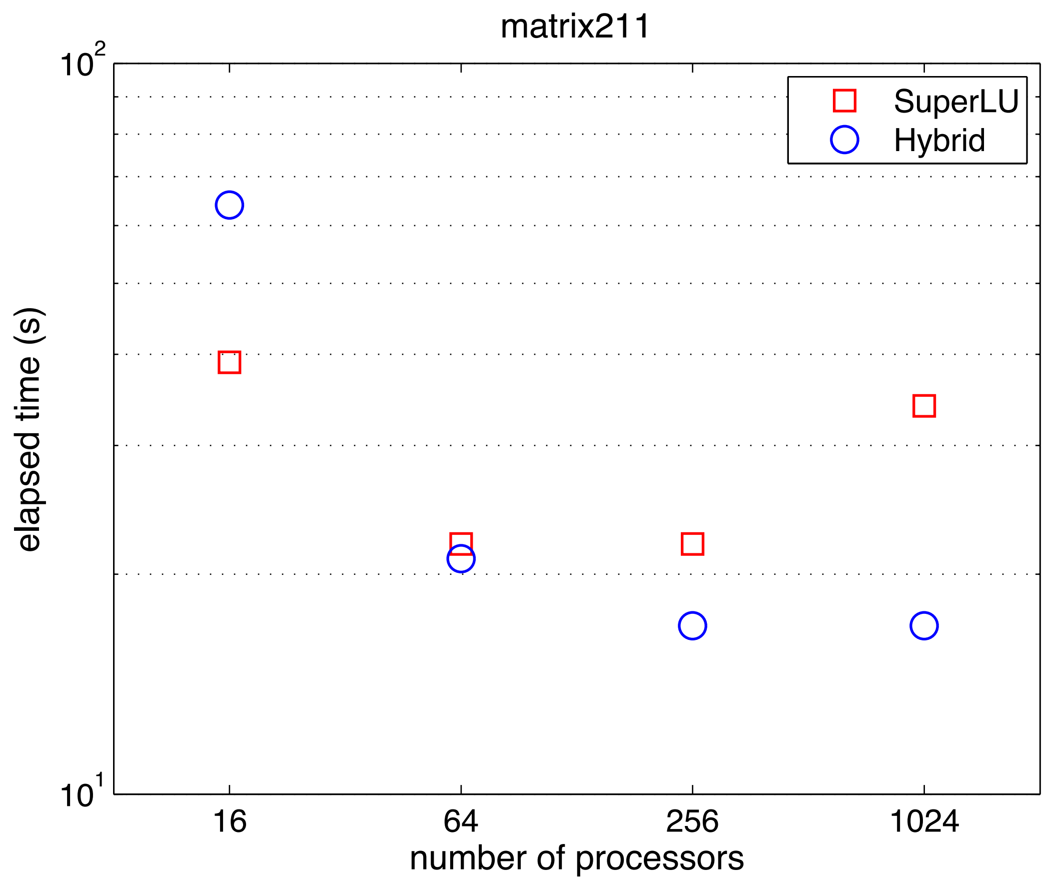


Elasticity problem on 32 cores

Weak scalability

PDSLin Parallel Performance

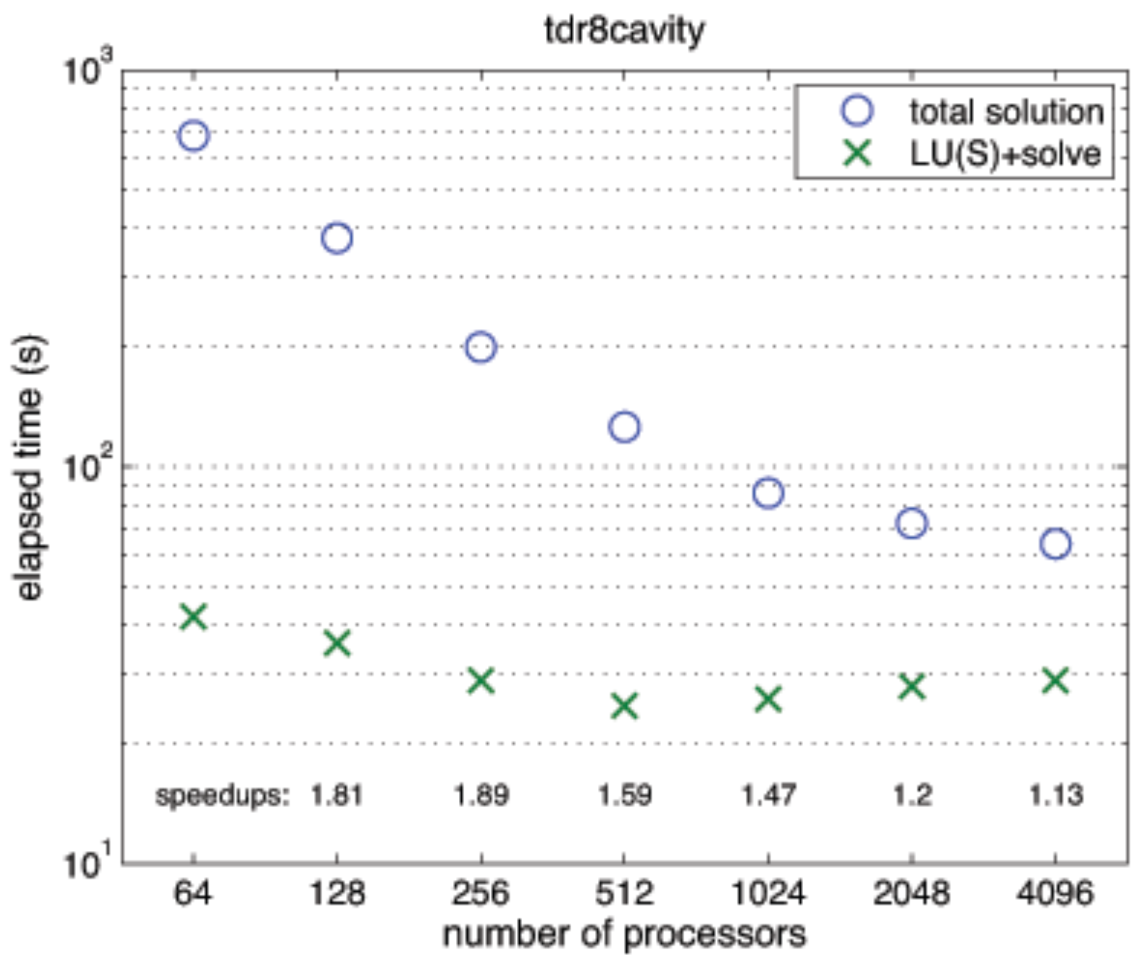
- Fusion problem:
  - Dimension = 801,378 (real unsymmetric, indefinite)
- Experimental setup:
  - PT-SCOTCH to extract 8 domains, each of size ~99K
  - SuperLU_DIST to factor each domain.
  - SuperLU_DIST to compute LU (S'), with S' ≈ S of size 13K, using 64 processors.
  - BICGStab from PETSc to solve Sy = c until rel residual < 10^{-12} (converged in ~10 iterations).



(on NERSC Cray XT-4)

PDSLin Parallel Performance

- ILC cavity problem:
  - Dimension = 17,799,228 (real symmetric, highly indefinite)
- Experimental setup:
  - PT-SCOTCH to extract 64 domains, each of size ~277K
  - SuperLU_DIST to factor each domain.
  - SuperLU_DIST to compute LU (S’), with S’ ≈ S of size 57K, using 64 processors.
  - BICGStab from PETSc to solve Sy = c until rel residual < 10^{-12} (converged in ~10 iterations).



(on NERSC Cray XT-4)

The France-Berkeley Fund Project

- Title : *Scalable Hybrid Solvers for Large Sparse Linear of Equations on Petascale Computing Architectures*
  - A project that provides travel funds to enable collaboration.
- Main focusses
  - Exploit hybrid programming models on NUMA clusters
  - Design parallel numerical techniques for augmented systems
- Start date: January 2011
- Duration: 1-2 years

- One visit by the French team
  - Luc Giraud & Jean Roman (February 14-16, 2011)
  - Emmanuel Agullo (February 14-25, 2011)
- Current research activity: perform a comparative study of the MaPHyS and PDSLin solver

Challenges

- Further algorithmic improvements are needed in both PDSLin and MaPhyS.

- Scalability (numerical and implementation) on $O(10^4\text{-}10^5)$ cores.
- Efficient implementation on heterogeneous many-core (CPU, GPGPU, …).

- Resilience embedded in solvers
  - In particular, preliminary investigations are ongoing at an ANR project:
    - Topics on the agenda of the INRIA's Large-Scale Initiative on "Very High Performance Computing for Computational Sciences".
- Deployment of solvers
  - A focus in the DOE SciDAC (Scientific Discovery Through Advanced Computing) Program.