International Conference on Scientific Computing

S. Margherita di Pula, Sardegna, Italy,

October 10-14, 2011

High-performance large eddy simulation of incompressible turbulent flows

Pasqua D'Ambra

Institute for High-Performance Computing and Networking (ICAR-CNR), Naples branch

pasqua.dambra@cnr.it

Joint work with

Andrea Aprovitola, Institute for Combustion Research (IRC), CNR

Filippo Maria Denaro, Second University of Naples

Daniela di Serafino, Second University of Naples

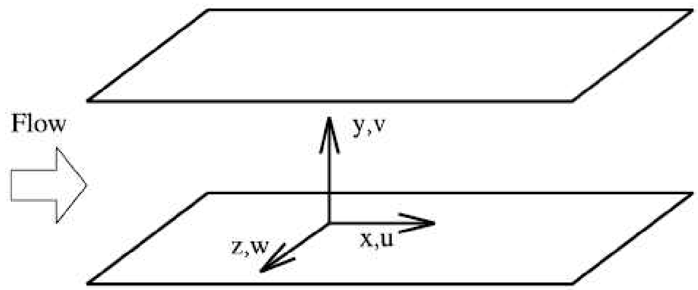
Salvatore Filippone, University of Rome “Tor Vergata”

Outline

- **Background & Motivations**
  - Large Eddy Simulation (LES) of turbulent channel flows
  - Numerical methods & main computational kernels
- **Par-LES**: a Parallel Simulation Code for LES based **on parallel scientific libraries for sparse matrix computations**
  - Grid partitioning and parallelism
  - **PSBLAS** for building and applying discrete operators and for sparse linear solvers
  - **MLD2P4** for building and applying algebraic multilevel preconditioners in sparse linear solvers
- **Performance Results**

LES of turbulent channel flows

Bi-periodical plane channel flow



- **N-S eq. (**$Re_\tau = u_\tau\, \delta/\nu$ = 180, 395, 590)
- **Domain sizes**: $2\pi\,\delta \times 2\delta \times \pi\delta$
- **Boundary conditions**: periodic in the streamwise (x) and spanwise directions (z); no-slip at the solid walls (y)
- **Initial condition**: Poiseuille flow with random Gaussian perturbation
- **Pressure forcing term**: assigned along the streamwise direction to set up a constant mass flow rate in the channel
- **LES approach**: FV formulation in which the filtered velocity is based on a *high-order deconvolution top-hat kernel filter*
- **SGS modelling**: *implicit eddy viscosity* due to multidimensional upwind flux discretization

LES of turbulent channel flows (cont'd)

- **Time integration:**
  - time splitting by **Approximate Projection Method** (**APM**)
  - 2nd-order semi-implicit Adams-Bashforth/Crank-Nicolson (AB/CN) scheme
- **Space discretization:**
  - non-uniform grid along y (stretching cosine law)
  - co-located flow variables (center of FVs)
  - multidimensional 3rd-order upwind scheme for convective fluxes
  - 2nd-order central scheme for diffusive fluxes
  - 4th-order formulas for spatial derivatives in inverse deconvolution

A. Aprovitola, F. M. Denaro, *A non-diffusive, divergence-free, Finite Volume-based double projection method on non-staggered grids*, Int. J. Num. Meth. Fluids, 53, 1127-1172, 2007

LES: APM numerical procedure

Helmholtz-Hodge decomposition

$$\tilde{\mathbf{v}}^{n+1} = \mathbf{v}^* - \Delta t \nabla \Phi$$

predictor-corrector approach (velocity field)

where v* is obtained by solving the semi-discrete **deconvolved momentum eq.** neglecting pressure terms

$$\left(A_x^{-1} - \frac{\Delta t}{2\mathrm{Re}_\tau} D_2\right)\mathbf{v}^* = \left(A_x^{-1} + \frac{\Delta t}{2\mathrm{Re}_\tau} D_2\right)\tilde{\mathbf{v}}^n +$$
$$\frac{\Delta t}{2}\left(3\left(\frac{1}{\mathrm{Re}_\tau}(D_1 + D_3)\tilde{\mathbf{v}}^n + \mathbf{f}_{conv}^n\right) - \left(\frac{1}{\mathrm{Re}_\tau}(D_1 + D_3)\tilde{\mathbf{v}}^{n-1} + \mathbf{f}_{conv}^{n-1}\right)\right)$$

with suitable Dirichlet boundary conditions near the walls.

LES: APM numerical procedure (cont'd)

Φ is obtained by solving the so-called **pressure equation**

$$(D_1 + D_2 + D_3)\Phi = \frac{1}{\Delta t \, |\Omega(\mathbf{x})|} \int_{\partial\Omega(\mathbf{x})} \mathbf{v}^* \cdot \mathbf{n} \, dS$$

with Neumann boundary conditions near the walls, ensuring compatibility (a solution exists up to an additive constant)

$$\nabla\Phi = \nabla p + \mathrm{O}(\Delta t)$$

$\nabla\Phi$ is a first order approximation of the pressure gradient

LES: computational kernels

Solution of the deconvolved momentum eq.:

$$\left(A_x^{-1} - \frac{\Delta t}{2\,\mathrm{Re}_\tau} D_2 \right) \mathbf{v}^* = \left(A_x^{-1} + \frac{\Delta t}{2\,\mathrm{Re}_\tau} D_2 \right) \tilde{\mathbf{v}}^n$$

$$+ \frac{\Delta t}{2} \left(3 \left(\frac{1}{\mathrm{Re}_\tau} (D_1 + D_3) \tilde{\mathbf{v}}^n + \mathbf{f}_{conv}^n \right) - \left(\frac{1}{\mathrm{Re}_\tau} (D_1 + D_3) \tilde{\mathbf{v}}^{n-1} + \mathbf{f}_{conv}^{n-1} \right) \right)$$

! $N=N_x \, x N_y \, x \, N_z$ total number of grid cells

1. ***compute** discrete inverse deconvolution by 4th-order scheme*

   $A_x^{-1} \rightarrow \mathbf{A} \in \Re^{N \times N}$

2. ***compute** discrete diffusive operators by 2nd-order scheme*

   $D_1, D_2, D_3 \rightarrow \mathbf{D}_i \in \Re^{N \times N}$

⇒ **Build sparse matrices independent of time-step**

LES: computational kernels (cont'd)

Solution of the deconvolved momentum eq. (cont'd):

! $N=N_x \, x N_y \, x \, N_z$ total number of grid cells

3. compute *discrete convective fluxes by multidimensional 3th-order up-wind scheme*

$\mathbf{f}_{conv}^{n}, \mathbf{f}_{conv}^{n-1}$ ➡ $\mathbf{q}_i^{n} = \alpha \mathbf{u}_i^{n}, \; \mathbf{q}_i^{n-1} = \alpha \mathbf{u}_i^{n-1} \qquad i = 1,2,3$ ⇨ **vectors updates**

4. build *rhs and coefficient matrix of the* **velocity systems**

$$\mathbf{M}\mathbf{v}_i^* = \mathbf{w}_i \quad \textit{with} \quad \mathbf{v}^* = (\mathbf{v}_i^*)_{i=1,2,3}$$

$$\mathbf{M} = \mathbf{A} - \beta \cdot \mathbf{D}_2$$

$$\mathbf{w}_i = (\mathbf{A} + \beta \cdot \mathbf{D}_2)\tilde{\mathbf{v}}_i^{\,n} +$$

$$+ \tau\left(3\left(\lambda \cdot (\mathbf{D}_1 + \mathbf{D}_3)\tilde{\mathbf{v}}_i^{\,n} + \mathbf{q}_i^{n}\right) - \left(\lambda(\mathbf{D}_1 + \mathbf{D}_3)\tilde{\mathbf{v}}_i^{\,n-1} + \mathbf{q}_i^{n-1}\right)\right)$$

⇨ **matrices updates/matrix-vector products/vector updates**

N.B. matrices independent of time-step

LES: computational kernels (cont'd)

Solution of pressure equation

$$(D_1 + D_2 + D_3)\Phi = \frac{1}{\Delta t \, |\Omega(\mathbf{x})|} \int_{\partial\Omega(\mathbf{x})} \mathbf{v}^* \cdot \mathbf{n} \, dS$$

! $N=N_x \, x N_y \, x \, N_z$ total number of grid cells

*1. **build** rhs and coefficient matrix of the sparse linear system*

$$\mathbf{D}\,\varphi = \mathbf{s}$$

$$\mathbf{D} = \mathbf{D}_1 + \mathbf{D}_2 + \mathbf{D}_3 \in \Re^{NxN}$$
$$\mathbf{s} \in \Re^{N}$$

matrices updates/vector update

N.B. matrix independent of time-step

LES: time-marching procedure

APM procedure

Build sparse matrices/vectors → **Sparse Matrix Management** (allocate, build, update)

! Nsteps = total number of time steps

for *n = 1, Nsteps* ***do***

1. ***compute*** *convective fluxes*
2. ***compute*** *diffusive fluxes*
3. ***compute*** *deconvolved velocity*

→ **Sparse BLAS** (matrix-vector prod.,vector up.)

4. ***solve*** *deconvolved momentum eq.*
5. ***solve*** *pressure eq.*

→ **Solution of sparse linear systems** (~75% of total cost)

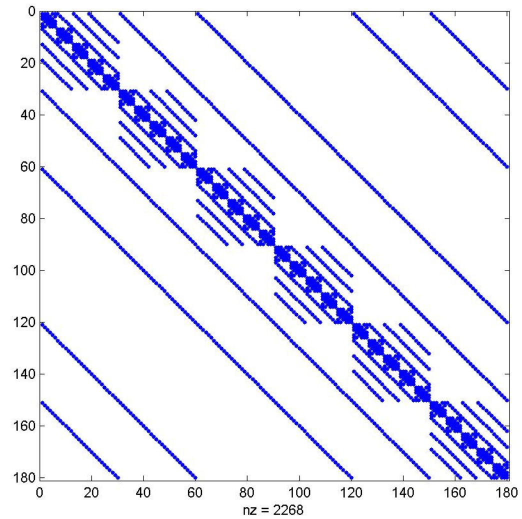
6. ***update*** *velocity*

endfor

Number of time steps and dimensions of matrices/vectors increasing with Reynolds number

LES: sparse linear solvers

Solution of velocity systems:



$$\mathbf{M}\mathbf{v}_i^* = \mathbf{w}_i \quad \textit{with} \quad \mathbf{v}^* = (\mathbf{v}_i^*)_{i=1,2,3}$$

Three linear systems whose matrix is:

✓large and sparse
✓unsymmetric with symmetric sparsity pattern
✓diagonally dominant

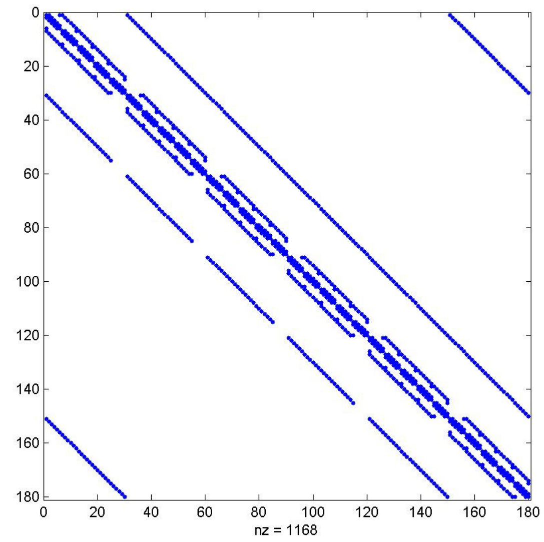
GMRES with point-Jacobi preconditioner

accounts for ~ 40% of the total run-time

LES: sparse linear solvers (cont'd)

Solution of pressure system:

$$\mathbf{D}\,\varphi = \mathbf{s}$$



Singular, but compatible linear system, whose matrix is

✓large & sparse
✓unsymmetric with symmetric sparsity pattern
✓condition number rapidly increasing for decreasing grid sizes

GMRES with Multilevel DD Preconditioners

accounts for ~ 35% of the total run-time

Par-LES

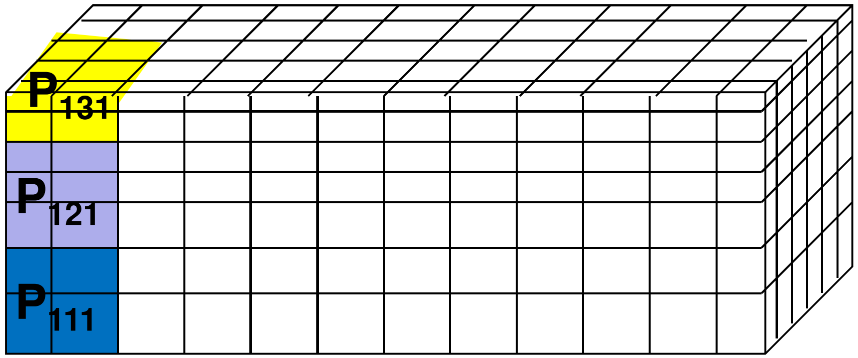
a parallel code for LES

based on portable, efficient and reliable **scientific libraries for parallel sparse matrix computations**

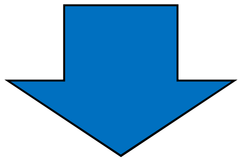
- Simplicity in changing numerical methods/solvers (modularity, flexibility)
- Parallelism encapsulated in library routines and support for distributed data structures creation/management (rapid and "low-cost" introduction of parallelism)
- Up-to-date parallel algorithms and standard base software (efficiency, portability,robustness)

Par-LES: basic choice for parallelism

Data parallelism based on 3D decomposition of computational grid

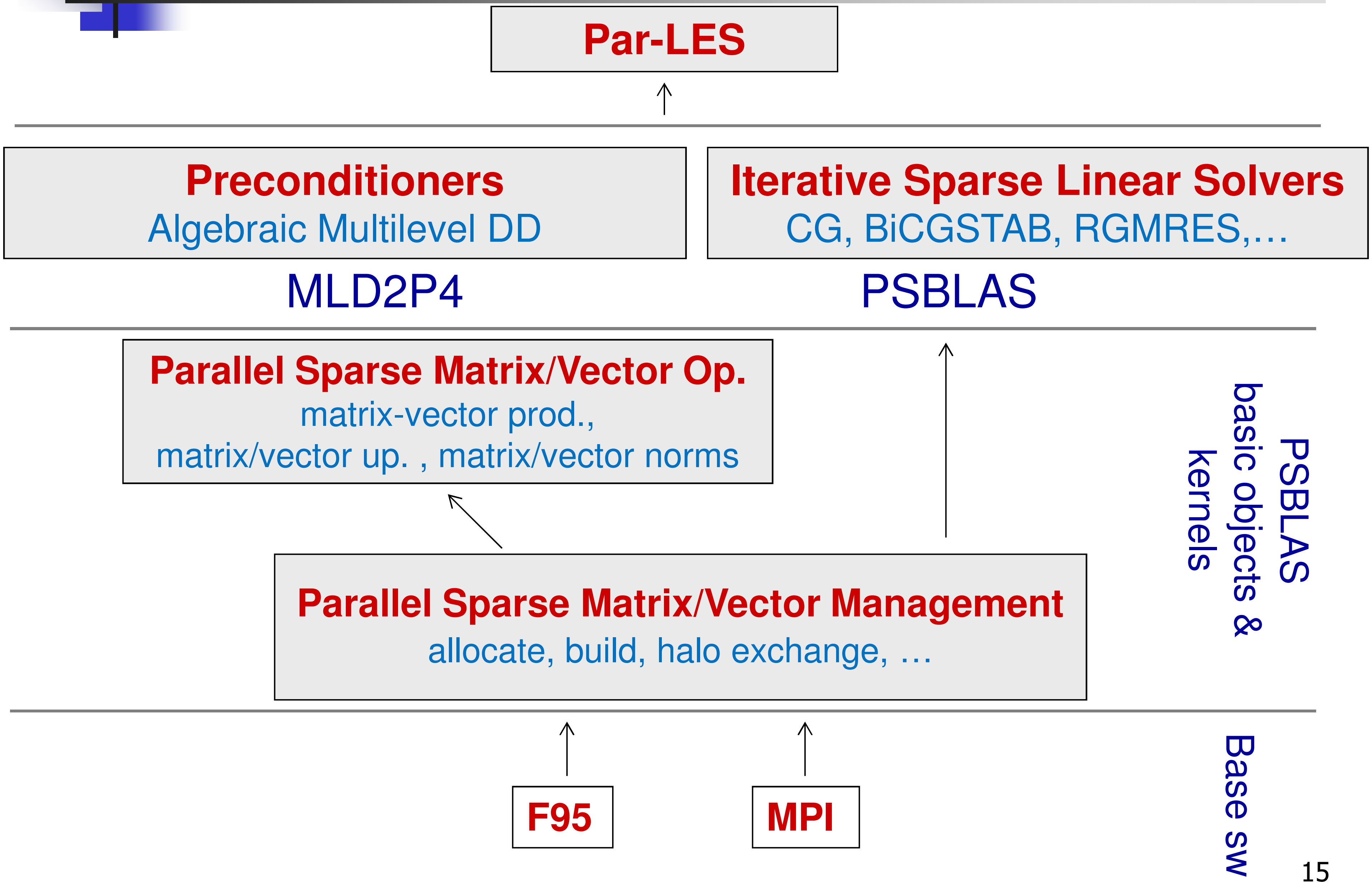


- ✓Processors are connected in a 3D Cartesian virtual topology
- ✓Every processor owns a 3D sub-block of computational grid, i.e. a block of consecutive rows of the matrices/vectors
- ✓Every processor locally computes on its internal points (*volume of sub-block*)
- ✓Nearest-neighbor processors communicate some layers of "halo" points (surface of sub-block)



Surface to volume effect, i.e. minimize T_{com}/T_{calc}

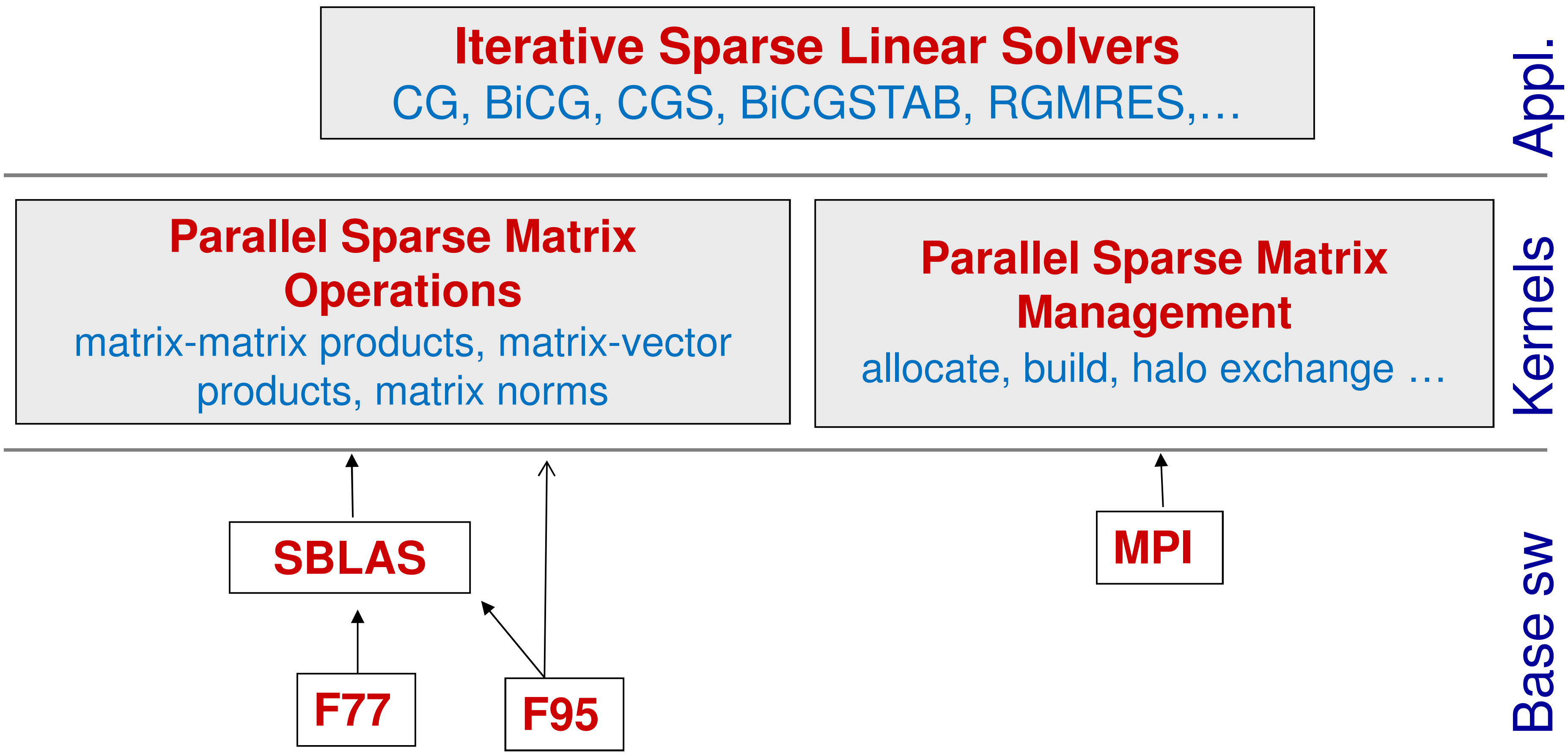
Par-LES: Software Architecture



PSBLAS (Filippone et al., www.ce.uniroma2.it/psblas/)

Parallel Sparse BLAS rel. 2.4

Basic Linear Algebra Operations with Sparse Matrices
on MIMD Architectures



Multilevel DD preconditioners

Domain Decomposition (DD) preconditioners:

- divide the PDE domain (the matrix) into subdomains (submatrices)
- apply a "local preconditioning" in each subdomain
- build the global preconditioner from the local ones

Additive Schwarz preconditioners – pros & cons:

- naturally fit in a parallel environment → good implementation scalability
- # iterations of the preconditioned solver grows with # subdomains
  → poor algorithmic scalability

Use of multiple level corrections to obtain optimal preconditioners
(# iterations bounded independently of grid size and subdom. diameter)

MLD2P4 (www.mld2p4.it)

Multi-**L**evel **D**omain **D**ecomposition
Parallel **P**reconditioners **P**ackage based on **PSBLAS**

mld_precbld(A,M,…)
A, distributed sparse matrix (input)
M, distributed sparse preconditioner (output)

mld_precaply(M,x,y,…)
M, distributed sparse preconditioner (input)
x,y, distributed vectors (input/output)

Diagonal
Block-Jacobi
Additive Schwarz with arbitrary overlap
Algebraic multi-level Schwarz based on aggregation

PSBLAS 2.0
extended version of PSBLAS 1.0

P. D'Ambra, D. di Serafino, S. Filippone, *MLD2P4: a Package of Parallel Algebraic Multilevel Domain Decomposition Preconditioners in Fortran 95*, ACM Transactions on Mathematical Software, Vol. 37, N. 3, 2010.

Preconditioners available in MLD2P4

Any combination of the following components

- Base (1-lev) preconditioner (smoothers):
  point-Jacobi, block-Jacobi, additive Schwarz (AS, RAS, ASH) with
  LU (UMFPACK) or ILU fact. (ILU(p), MILU(p), ILU(t,p)) of the blocks
- Multilevel type:
  additive or multiplicative, with pre-, post- or two-side smoothing
- Coarsening strategy:
  decoupled classical or unsymmetric smoothed aggregation
- Coarsest-level matrix:
  distributed or replicated
- Coarsest-matrix solver:
  ILU, sparse LU (UMFPACK, SuperLU, SuperLU_Dist), block-Jacobi with
  ILU or LU on the blocks, pont-Jacobi

Par-LES: Test Case

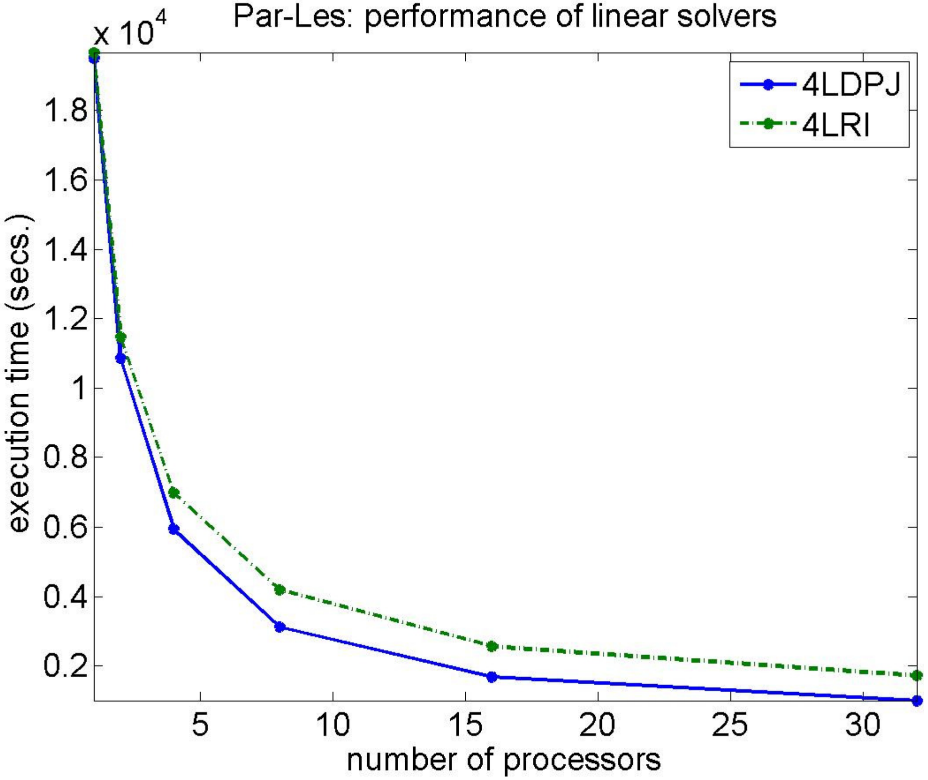
- $Re_\tau = 590$
- Number of grid cells: 48x100x48
  (providing a resolved boundary layer with minimum cell size $\Delta y+ = 0.145$)
- $\Delta t = 10^{-5}$ (estimated on the base of linear stability constraints)
- Matrices Dimension: N=228096
  - Nonzeros for velocity 2960640, Nonzeros for pressure 1589902
- Preconditioners for pressure system:
  - 4-lev V cycle, with RAS(1) as smoother, replicated coarse matrix: 4 sweeps of Block Jacobi with ILU(0) on diagonal blocks **(4LRI)**
  - 4-lev V cycle, with Block Jacobi as smoother, distributed coarse matrix: 4 sweeps of Point Jacobi **(4LDPJ)**
- Stopping criterion for linear solvers:: $\| r_k \| / \| r_0 \| \leq 10^{-7}$ or maxiter

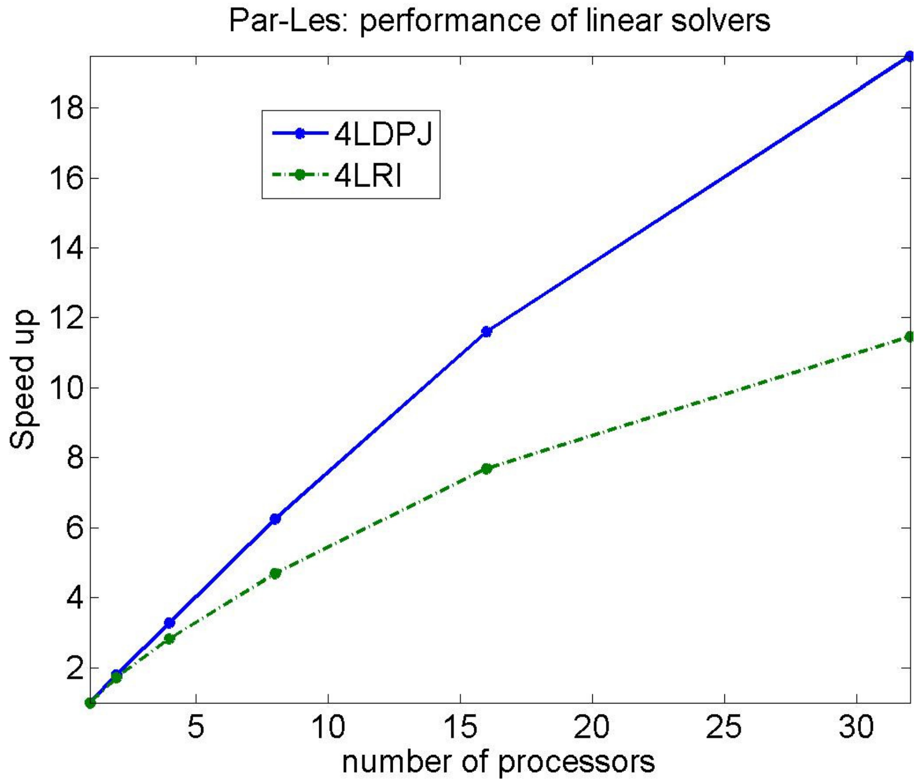
Par-LES: computational environment

HP XC 6000 Linux cluster

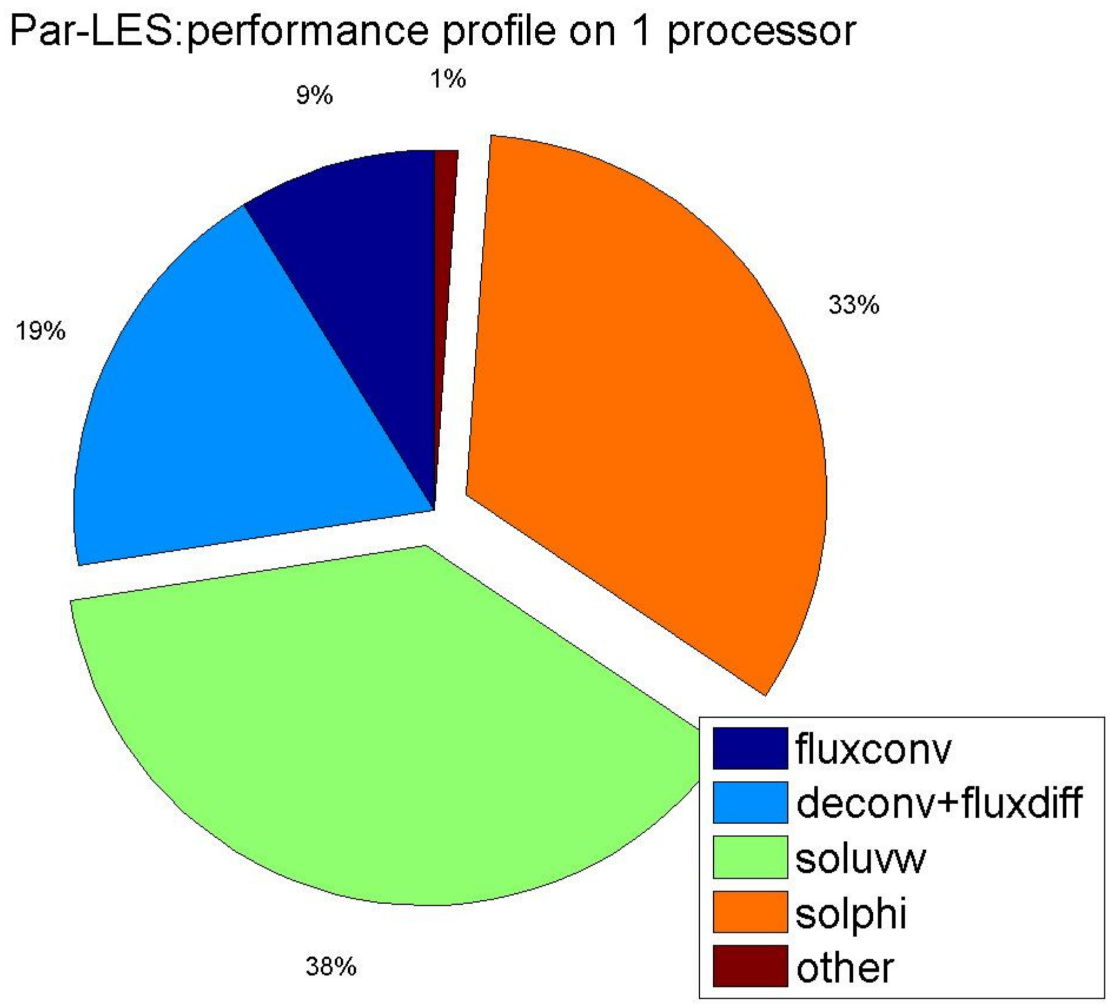
- 64 Intel Itanium 2 Madison (1.4 GHz) bi-processor nodes with 4 GB of RAM
- Quadrics QsNetII Elan 4 interconnection network (900 MB/sec. of bandwidth and 5μsec. of latency)
- HP Linux for High Performance Computing, based on Red Hat Enterprise Linux AS 3 with Kernel 2.4.21
- GNU 4.6 compilers
- HP MPI implementation
- BLAS implementation provided by ATLAS 3.6.0
- PSBLAS 2.4

Par-LES: performance results (2000 time steps)

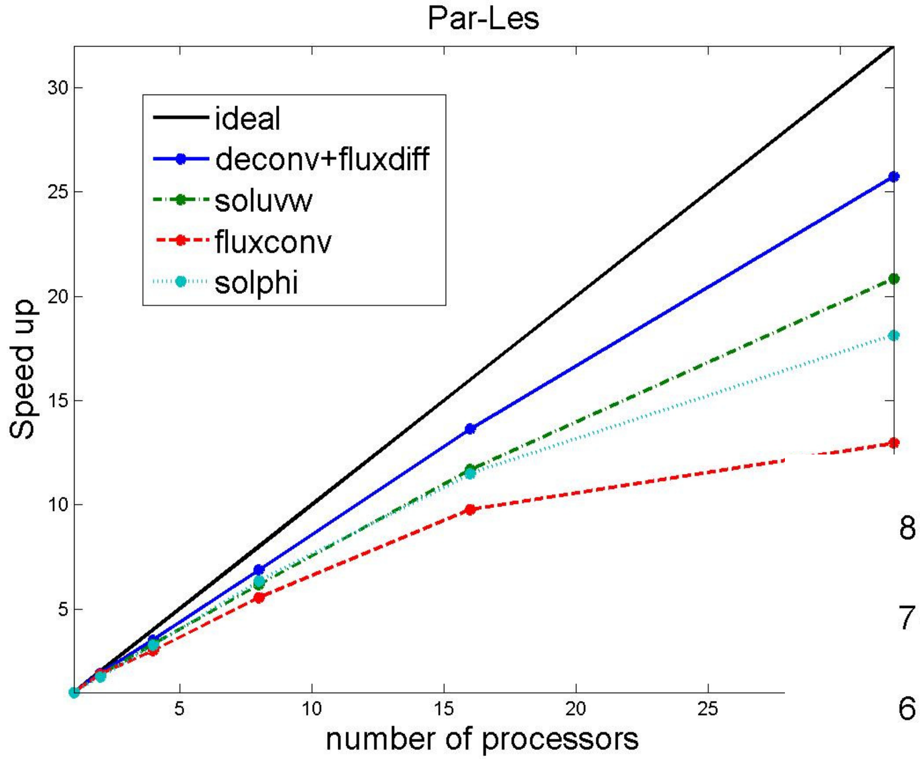


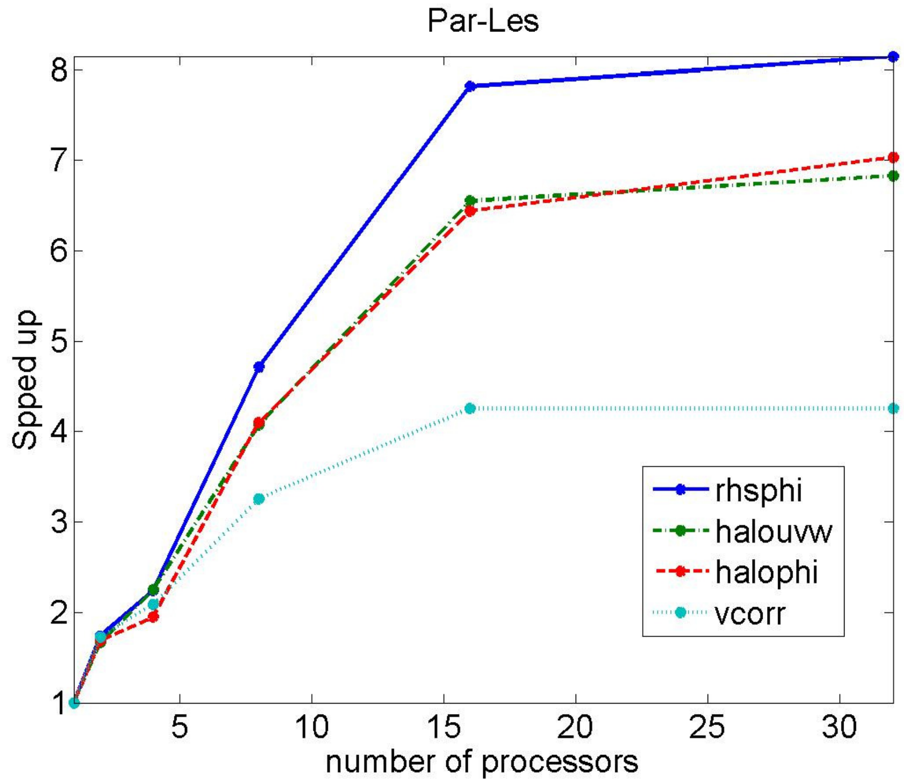


Par-LES: performance results (4LDPJ-2000 time steps)

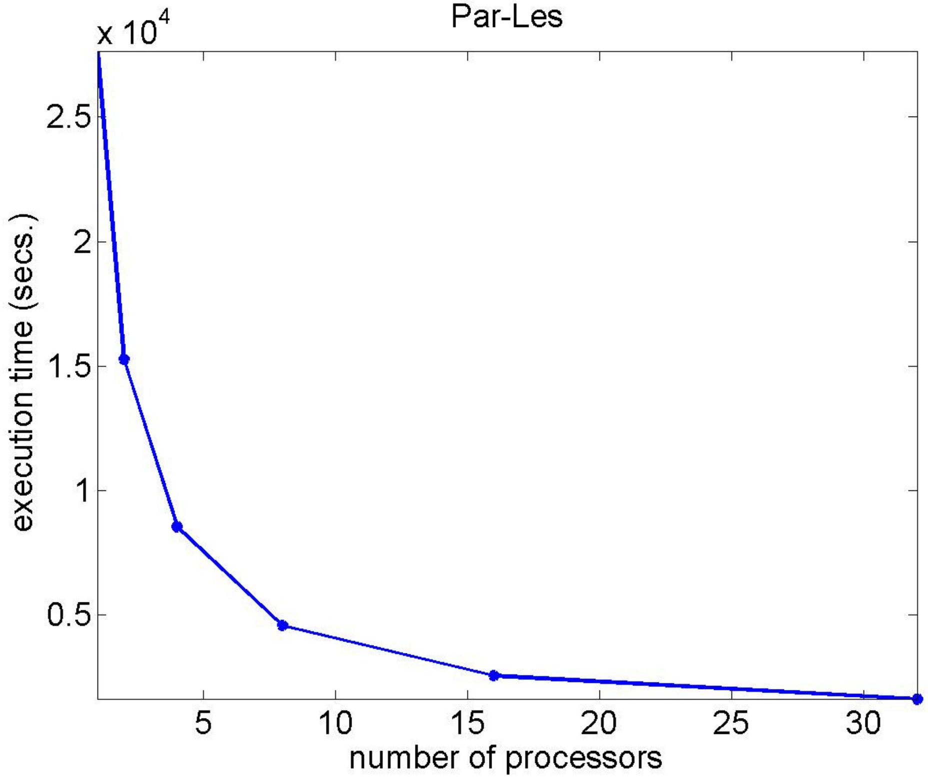


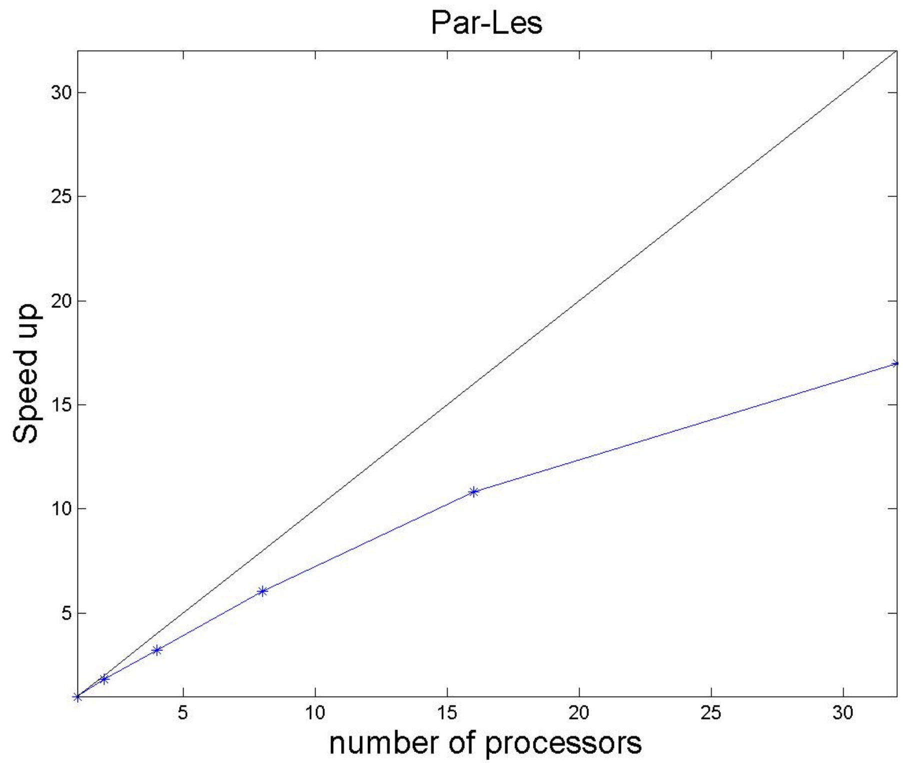
Par-LES: performance results (4LDPJ-2000 time steps)





Par-LES: performance results (4LDPJ-2000 time steps)





Concluding remarks

- **Par-LES** shows good **strong scalability** for medium size Reynolds numbers
- **Par-LES is based on portable and reliable** open-source **parallel scientific libraries** based on MPI, thus it can be run on different architectures on which MPI is available.
- **Many variants of linear solvers and preconditioners can be tested** by changing only input parameters
- **Improvements or extensions to PSBLAS/MLD2P4 libraries** (shift through many-core/GPU architectures) **will be available to Par-LES with no effort**

Thank you for your attention