High performance simulations using FreeFem++ on mixed distributed- plus shared-memory architecture

Atsushi Suzuki[1]

[1]Cybermedia Center, Osaka University
atsushi.suzuki@cas.cmc.osaka-u.ac.jp

- Pierre Jolivet, CNRS, ENSEEIHT, France
- Daisuke Furihata, Cybermedia Center - Osaka Univ.

description and solution of linear solver in multi-physics problem

- multi-physics problem is modeled by nonlinear partial differential equations
  weak / strong coupling $\longrightarrow$ iteration of solution of linear equations
- Finite element matrices are generated directly from discretization of weak formulation in `FreeFem++` developed by F. Hecht, LJLL UMPC
- domain decomposition solver with/without overlap in MPI distributed memory environment is performed by `HPDDM` developed by P. Jolivet
- Stiffness matrix is factorized by sparse direct solver in shared memory environment by `Dissection` developed A. Suzuki

strategy for linear solver in large scale

# unknowns	1 million	10 million	100 million
solver	direct solver : Pardiso/MUMPS/ Dissection	GMRES + additive Schwarz preconditioner	iterative substructuring + BDD / GenEO
hardware	16core / 64GB	160core / 640GB	512core / 2TB

example of simplest multi-physics problem

non-dimensionalized Rayleigh-Bénard equations at stationary state : (u, p, θ)

$$\frac{1}{Pr} u \cdot \nabla u - 2\nabla \cdot D(u) + \nabla p = Ra\theta \vec{e}_3 \text{ in } \Omega\,,$$

$$\nabla \cdot u = 0 \text{ in } \Omega\,,$$

$$u \cdot \nabla \theta - \triangle \theta = 0 \text{ in } \Omega$$

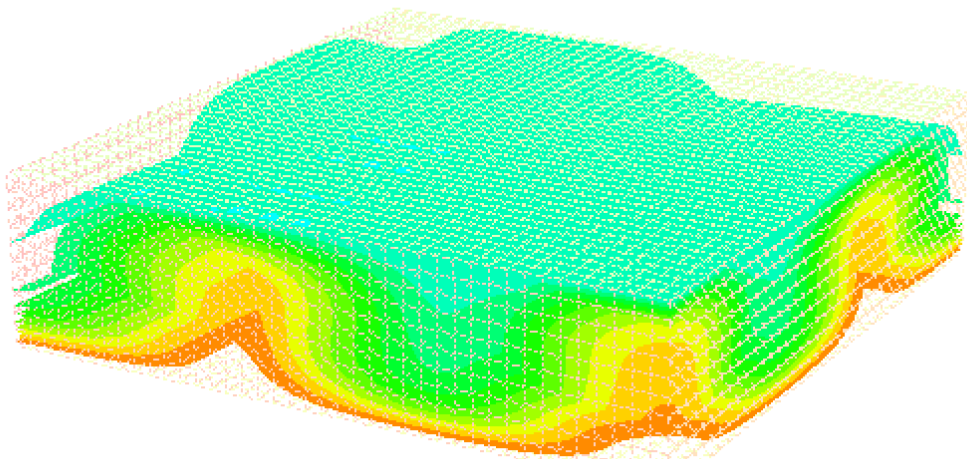
$$u \cdot n = 0 \text{ on } \partial\Omega\,,$$

$$\theta = 1 \text{ on } \Gamma_1\,, \theta = 0 \text{ on } \Gamma_3\,, \partial_n \theta = 0 \text{ on } \Gamma_2 \cup \Gamma_4\,.$$

strong coupling leads to global nonlinear problem

The stiffness matrix at k-th Newton iterator is unsymmetric and indefinite.

$$K = \begin{bmatrix} A_0 + A_1(\vec{u^k}) & B^T & D_{Ra} \\ B & 0 & 0 \\ C_2(\vec{\theta^k}) & 0 & C_0 + C_1(\vec{u}^k) \end{bmatrix}$$



temperature distribution of a stationary state

FreeFem++ script for Rayleigh-Bénard equations : shared memory

```
load "Dissection"                    // usage of sparse direct solver
defaulttoDissection();
real Pr = 7.0;  real Ra = 1500.0;
macro d11(u1)      dx(u1) //
macro d12(u1,u2)   (dy(u1) + dx(u2))/2.0 //
macro ugrad(u1,u2,u3,v) (u1*dx(v) + u2*dy(v) + u3*dz(v)) //
macro Ugrad(u1,u2,u3,v1,v2,v3) [ugrad(u1,u2,u3,v1),
                  ugrad(u1,u2,u3,v2), ugrad(u1,u2,u3,v3)] //
mesh3 Th=readmesh3("mesh.data"); // prepared in advance
func Pk = [P2,P2,P2,P1,P2];          // P2-P1 for fluid,
fespace Wh(Th,Pk);                   // P2 for temperature
varf vDRB ([u1,u2,u3,p,th],[v1,v2,v3,q,psi]) =
  int3d(Th)( 2.0*(d11(u1)*d11(v1)+d22(u2)*d22(v2)+d33(u3)*d33(v3)+
                  2.0*d12(u1,u2)*d12(v1,v2)+
                  2.0*d13(u1,u3)*d13(v1,v3)+
                  2.0*d23(u2,u3)*d23(v2,v3))
    - p * div(v1, v2, v3) - q * div(u1, u2, u3)
    + (Ugrad(u1,u2,u3,up1,up2,up3)'*[v1,v2,v3] // '
    + Ugrad(up1,up2,up3,u1,u2,u3)'*[v1,v2,v3]) / Pr) //'
  - int3d(Th)( Ra * th * v3 )
  + int3d(Th)( dx(th)*dx(psi)+dy(th)*dy(psi)+dz(th)*dz(psi) )
  + int3d(Th)( ugrad(up1,up2,up3,th)*psi + ugrad(u1,u2,u3,thp)*psi )
  + on(1,3,u2=1.0) + on(2,4,u1=1.0) + on(5,6,u3=1.0)
  + on(5,6,th=1.0);
matrix K=vDRB(Wh,Wh,solver=sparsesolver);
real[int] b=vRHS(0, Wh);
Wh [u1,u2,u3,p,th];                  // data on finite eleement mesh
u1[] = K^-1 * b                      // solution by Dissection
```

FreeFem++ script for Rayleigh-Bénard equations : distributed memory

```
load "hpddm"                              // for additive Schwarz solver
mpiComm comm(mpiCommWorld,0,0);  // MPI communicator
real Pr = 7.0;  real Ra = 1500.0;
macro d11(u1)     dx(u1) //
// ....
mesh3 Th=readmesh3("mesh.data"); // prepared in advance
func Pk = [P2,P2,P2,P1,P2];       // P2-P1 for fluid,
fespace Wh(Th,Pk);                  // P2 for temperature
varf vDRB ([u1,u2,u3,p,th],[v1,v2,v3,q,psi]) =
  int3d(Th)( 2.0*(d11(u1)*d11(v1)+d22(u2)*d22(v2)+d33(u3)*d33(v3)+
// ....
    + Ugrad(up1,up2,up3,u1,u2,u3)'*[v1,v2,v3]) / Pr) //'
  - int3d(Th)( Ra * th * v3 )
  + int3d(Th)( dx(th)*dx(psi)+dy(th)*dy(psi)+dz(th)*dz(psi) )
  + int3d(Th)( ugrad(up1,up2,up3,th)*psi + ugrad(u1,u2,u3,thp)*psi )
  + on(1,3,u2=1.0) + on(2,4,u1=1.0) + on(5,6,u3=1.0)
  + on(5,6,th=1.0);
int[int][int] intersection;     // map for index numbering
real[int] D;                          // partition of unity
int s = 1;                              // width of overlap of domains
build(Th, s, intersection, D, Pk, comm); //domain decomposition
matrix K=vDRB(Wh,Wh);             // local stiffness matrix
real[int] b=vRHS(0, Wh);          // local RHS vector
schwarz dK(K, itersection, D, comm);
set(dK, sparams="-hpddm_schwarz_method ras -hpddm_tol 1.e-12");
Wh [u1,u2,u3,p,th];                 // data on local finite element mesh
u1[] = dK^-1 * b                      // solution in parallel
```

additive Schwarz preconditioner for overlapping subdomains

linear equation : $A\,x = b$, A : stiffness matrix from FEM,
symmetric or unsymmetric, large sparse.
overlapping decomposition of the matrix $\Lambda = \bigcup_{p=1}^{P} \Lambda_p,\ \Lambda_p \cap \Lambda_q \neq \emptyset$

- R_p: restriction from the total DOF to sub-matrix : $\Lambda \to \Lambda_p$
- D_p : discrete representation of partition of the unity

$$\sum_{p=1}^{P} R_p^T D_p R_p = I_N, \quad [D_p]_{kk} = \begin{cases} 1 & k \in \Lambda_p,\ k \notin \Lambda_q, \forall q \neq p, \\ 1/\#\{p; k \in \Lambda_p\} & \text{otherwise} \end{cases}$$

restricted additive Schwarz (RAS) preconditioner

$$M_{\text{RAS}}^{-1} = \sum_{p=1}^{P} R_p^T D_p (R_p A R_p^T)^{-1} R_p$$

RAS converges much faster than block preconditioner without overlap,
$\sum_{p=1}^{P} \widehat{R}_p^T (\widehat{R}_p A \widehat{R}_p^T)^{-1} \widehat{R}_p$.

- graph partitioner METIS provides non-overlapping decomposition of graph corresponding to finite element nodes and elements: $\Lambda_p \cap \Lambda_q = \emptyset$, restriction operator $\widehat{R}_p$, $\widehat{R}_p^T \widehat{R}_q = 0$.
- overlapping decomposition $\{\Lambda_p\}$ is obtained by adding neighboring nodes of $\widehat{\Lambda}_p$

iterative substructuring method with preconditioner

nonoverlappig decomposition of the matrix $\bigcup_{p=1}^{P} \bar{\Omega}_p = \Omega$, $\Omega_p \cap \Omega_q = \emptyset$, $p \neq q$.
interface problem to find Neumann data on $\partial\Omega_p \cap \partial\Omega_q$.
local stiffness matrix with Neumann boundary condition

$$A^{(p)} = \begin{bmatrix} A_{II}^{(p)} & A_{IB}^{(p)} \\ A_{BI}^{(p)} & A_{BB}^{(p)} \end{bmatrix} = \begin{bmatrix} A_{II}^{(p)} & 0 \\ A_{BI}^{(p)} & S^{(p)} \end{bmatrix} \begin{bmatrix} E_{II}^{(p)} & A_{II}^{(p)-1} A_{IB}^{(p)} \\ 0 & E_{BB}^{(p)} \end{bmatrix}$$

$S^{(p)} = A_{BB}^{(p)} - A_{BI}^{(p)} A_{II}^{(p)-1} A_{IB}^{(p)}$: local Schur complement
global problem $A\,x = b$ is converted to interface problem $S\,y = f$ with global Schur complement

$$S = \sum_{p=1}^{P} R^{(p)T} S^{(p)} R^{(p)}$$

local Schur complement $S^{(p)}$: singular for floating subdomain Ω_p,
$\partial\Omega_p \cap \partial\Omega = \emptyset$
Neumann-Neumann preconditioner

$$Q_{NN} = \sum_{1 \leq p \leq p} R^{(p)T} D^{(p)} S^{(p)\dagger} D^{(p)} R^{(p)}$$

$S^{(p)\dagger}$: generalized inverse = inverse of $S^{(p)}$ on Im($S^{(p)}$)
Dissection can detect Ker($S^{(p)}$) from $A^{(p)}$, numerically

Balancing Neumann-Neumann and GenEO preconditioner

- Neumann-Neumann preconditioner drops global information due to restriction to local problem
- global information is recovered by introduction of coarse space
  - BDD : coarse space from collection of zero energy mode from kernel of local Schur complement
  - GenEO : enrichment of coarse space from eigenvalue analysis of local Schur complement

coarse space and preconditioner of BDD and preconditioner

$$W = \{v\,; v = \sum_{1\leq p\leq P} R^{(p)T} D^{(p)} v^{(p)}, \quad v^{(p)} \in \mathrm{Ker} S^{(p)}\}$$

$$Q_{NN} = \sum_{1\leq p\leq p} R^{(p)T} D^{(p)} S^{(p)\dagger} D^{(p)} R^{(p)}, \quad P: \text{ orthogonal projection onto } W$$

enriched coarse space of GenEO with local eigenvectors $(\lambda^{(p)}, v^{(p)})$

$$W^{(p)} = \{v^{(p)}\,; S^{(p)} v^{(p)} = \lambda^{(p)} v^{(p)}\}, \quad W = \{v\,;\, v = \sum_{1\leq p\leq P} R^{(p)T} D^{(p)} v^{(p)},\, v^{(p)} \in W^{(p}$$

- local Schur complement needs to be formed explicitly
- ARPACK to solve eigenvalue problem with dense local Schur complement

Dissection solves sparse multiple RHS problem

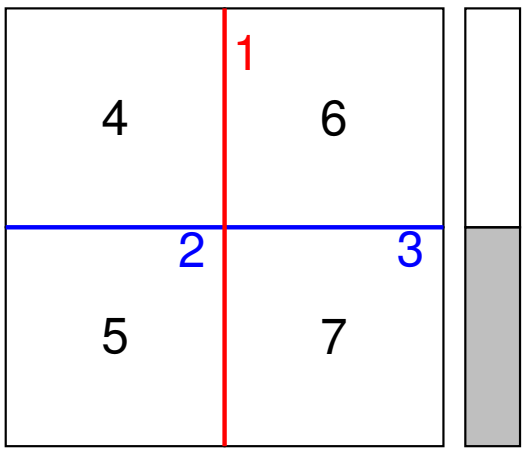
Dissection solves to form local Schur complement matrix

local Schur complement matrix needs to be explicitly formed for GenEO preconditioner

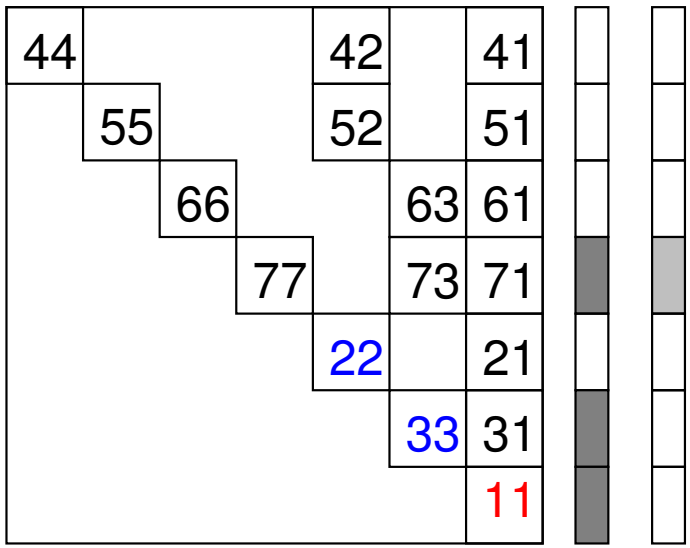
$A_{II}^{(p)}$ is factorized as $L_{II}^{(p)} D_{II}^{(p)} U_{II}^{(p)}$ by using nested-dissection ordering

Schur complement is formed with two forward substitution, lower and upper

$$
\begin{aligned}
S^{(p)} &= A_{BB}^{(p)} - A_{BI}^{(p)} A_{II}^{(p)-1} A_{IB}^{(p)} \\
&= A_{BB}^{(p)} - (U_{II}^{(p)} A_{BI}^{(p)T})^T D_{II}^{(p)-1} (L_{II}^{(p)-1} A_{IB}^{(p)})
\end{aligned}
$$



nested-dissection of $\Omega^{(p)}$ and $A_{IB}^{(p)}$



sparse RHS solution

- forward substitution can be performed by half of the bisection tree with grouping of RHS $A_{IB}^{(p)}$

conclusion

- nonlinear solution of multi-physics problem is well described by FreeFem++ domain specific language
- HPDDM manages local stiffness matrix by domain decomposition from METIS graph partitioner
- additive Schwarz preconditioner can handle parallelization with moderate number of overlapping subdomains using MPI combined with multi-threaded direct solver on shared memory node
- balancing Neumann-Neumann preconditioner uses coarse space from zero energy mode, which is numerically detected during LDU-factorization process in Dissection solver
- GenEO preconditioner uses enriched coarse space by eigenvalue analysis with explicit formation of Schur complement, for which Dissection solver can reduce complexity of multiple RHS solution by employing sparsity pattern of the stiffness matrix
- FreeFem++ is implemented on NEC SX-ACE without facility of dynamic loading expansion due to lack of shared library