# Schwarz for the “outer-loop”

Xiao-Chuan Cai

Department of Computer Science
University of Colorado Boulder

Happy 20th birthday PETSc

Thanks Bill, Barry, ... the whole PETSc team!

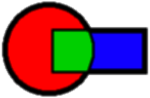
# The pre-PETSc pre-MPI days

I feel like I have been using “PETSc” for 26 years – since 1989 at Yale

Bill’s “tile” algorithms ... “tile” codes. There was a communication piece and a DD piece ...

# Outline – Schwarz for the outer-loops

- Nonlinear solver loop
- Multi-physics loop
- Time integration loop
- Optimization loop
- Some final remarks



# Differences between inner solve and outer loop

- Sophisticated parallel preconditioned Krylov subspace methods (such as Krylov-Schwarz or Krylov-MG) are often used for the inner solve

- In most existing approaches, the outer loops are handled sequentially (time integration, optimization, etc)
- Simple algorithms are often used for the outer loops (fixed-point iteration for electronic structures, GS iteration for optimization, etc)

- What we want are both or one of the following:
  - More parallelism for some of the outer loops
  - More robustness for the outer loop solver
- Can we include some of the outer loops inside the Schwarz domain to increase parallelism and robustness (only for large machines)?

Some old and new examples

# A question asked in 1992-1994

To design a general purpose parallel nonlinear solver using three components: Schwarz, Newton, Krylov, (like the one in PETSc) which one should be used for the outer loop?

Choice#1 (Schwarz-Newton-Krylov algorithm):

- *Schwarz loop*
  - *Newton loop*
    - *Krylov loop*

Supporting paper: Cai and Dryja (1994)

Choice#2 (Newton-Krylov-Schwarz algorithm):

- *Newton loop*
  - *Krylov loop*
    - *Schwarz loop*

Supporting paper: Cai, Gropp, Keyes, Tidriri (1994)

Conclusion: Unless the problem is a simple elliptic equation, Schwarz should not be used for the “out-most” outer loop since it doesn’t have a global view of the problem. Newton is a better choice

# Use Schwarz to reduce the layer of outer loops

Consider a multi-physics problem: fluid-structure interaction

- *Time loop*
  - $3 \times 3$ *block Gauss-Seidel loop*
    - *Solid loop*
    - *Fluid loop*
    - *Moving mesh loop*

This can be reduced to

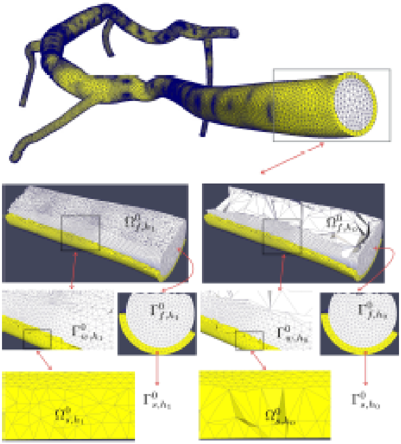
- *Time loop*
  - *Krylov + monolithic Schwarz loop*

This improves the robustness of the algorithm, increases the parallelism

# A fully coupled fluid-structure interaction problem

$$
\begin{cases}
\rho_f \dfrac{\partial \boldsymbol{u}_f}{\partial t}\Big|_{\boldsymbol{X}_f} - \nabla \cdot \boldsymbol{\sigma}_f + \rho_f \left[ \left( \boldsymbol{u}_f - \dfrac{\partial \boldsymbol{d}_m}{\partial t} \right) \cdot \nabla \right] \boldsymbol{u}_f = \rho_f \boldsymbol{f}_f & \text{in } \Omega_f^t, \\
\nabla \cdot \boldsymbol{u}_f = 0 & \text{in } \Omega_f^t, \\
\boldsymbol{u}_f = \boldsymbol{v}_f^d & \text{on } \Gamma_{f,d}^t, \\
\boldsymbol{\sigma}_f \boldsymbol{n}_f = \boldsymbol{g}_f & \text{on } \Gamma_{f,n}^t, \\
 & \\
-\nabla \cdot \boldsymbol{\sigma}_m = 0 & \text{in } \Omega_f^0, \\
\boldsymbol{d}_m = 0 & \text{on } \Gamma_f^0, \\
 & \\
\rho_s \dfrac{\partial^2 \boldsymbol{d}_s}{\partial t^2} + \eta_s \dfrac{\partial \boldsymbol{d}_s}{\partial t} - \nabla \cdot \boldsymbol{\Pi}_s = \rho_s \boldsymbol{f}_s & \text{in } \Omega_s^0, \\
\boldsymbol{d}_s = 0 & \text{on } \Gamma_{s,d}^0, \\
\boldsymbol{\Pi}_s \boldsymbol{n}_s = \boldsymbol{g}_s & \text{on } \Gamma_{s,n}^0, \\
 & \\
\boldsymbol{u}_f = \dfrac{\partial \boldsymbol{d}_s}{\partial t} & \text{on } \Gamma_w^t, \\
\boldsymbol{\sigma}_s \hat{\boldsymbol{n}}_s = -\boldsymbol{\sigma}_f \boldsymbol{n}_f & \text{on } \Gamma_w^t, \\
\boldsymbol{d}_m = \boldsymbol{d}_s & \text{on } \Gamma_w^0.
\end{cases}
$$

# A fully coupled fluid-structure interaction problem



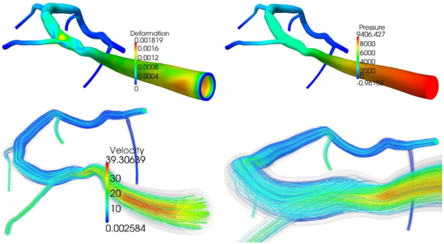


**Table :** The strong scalability results. A FSI problem with 280,806,789 unknowns is solved by a three-level monolithic Schwarz method

| *np* | subsolve | NI | iter | time(s) | *sp* | eff |
|---|---|---|---|---|---|---|
| 4,096 | ILU(0) | 2 | 3.8 | 539.64 | 1 | 100% |
| 4,096 | ILU(1) | 2 | 4.5 | 616.16 | 1 | 100% |
| 4,096 | ILU(2) | 2 | 3 | 645.82 | 1 | 100% |
| 6,144 | ILU(0) | 2 | 3.8 | 368.3 | 1.47 | 98% |
| 6,144 | ILU(1) | 2 | 3.8 | 403.12 | 1.5 | 100% |
| 6,144 | ILU(2) | 2 | 3 | 446.38 | 1.45 | 96% |
| 8,192 | ILU(0) | 2 | 3.8 | 286.51 | 1.88 | 94% |
| 8,192 | ILU(1) | 2 | 3 | 307.23 | 2 | 100% |
| 8,192 | ILU(2) | 2 | 3 | 346.80 | 1.86 | 93% |
| 10,240 | ILU(0) | 2 | 4.2 | 248.33 | 2.17 | 87% |
| 10,240 | ILU(1) | 2 | 2.8 | 243.95 | 2.5 | 100% |
| 10,240 | ILU(2) | 2 | 3.8 | 312.51 | 2.07 | 83% |

# Loop for time integration

- In traditional parallel methods for solving time dependent PDEs, the parallelism is obtained by partitioning the spatial mesh, the time integration is handled sequentially
  - When the number of time steps is large, there are lots of sequential steps
- When the number of processors is large, the sequential time integration part is becoming a bottleneck
- There are several attempts to develop parallel-in-time algorithms such as multiple shooting, parareal (Lions, Maday, Turinici 2001), etc.

Consider a parabolic equation

$$\begin{cases} \dfrac{\partial u}{\partial t} + Lu &= f(x) \quad \text{in} \quad D \times T \\ u(x,0) &= u^0(x) \quad \text{in} \quad D \\ u(x,t) &= 0 \quad \text{on} \quad \partial D \times T, \end{cases}$$

Let $L_h(\cdot)$ be the 5-point finite difference discretization. We have

$$\begin{cases} \dfrac{u_h^{k+1} - u_h^k}{\Delta t} + L_h(u_h^{k+1}) = f_h^{k+1} \\ u_h^0 \quad \text{given} \end{cases}$$

We denote by $u_h^k = \{u_{i,j}^k\}$ the solution vector consisting of all nodal values at the $k^{th}$ time level

## Coupling space and time into a large linear system

Stacking $s$ levels of solutions into a single vector, $U = (u_h^1, u_h^2, \cdots, u_h^s)^T$, which satisfies the space-time system of equations, $A_s U = B$

$$\begin{pmatrix} L_h & & & & & \\ -I & L_h & & & & \\ & & \ddots & & & \\ & & -I & L_h & & \\ & & & & \ddots & \\ & & & & -I & L_h \end{pmatrix} \begin{pmatrix} u_h^1 \\ u_h^2 \\ \vdots \\ u_h^k \\ \vdots \\ u_h^s \end{pmatrix} = \begin{pmatrix} u_h^0 + \Delta t\ f^1 \\ \Delta t\ f^2 \\ \vdots \\ \Delta t\ f^k \\ \vdots \\ \Delta t\ f^s \end{pmatrix}$$

# Space-time Schwarz

One-level additive Schwarz preconditioner

$$M_{one}^{-1} = M_1^{-1} + M_2^{-1} + \cdots + M_N^{-1}$$

Two-level additive Schwarz preconditioner

$$M_{two}^{-1} = I_c^f \, M_c^{-1} \, (I_c^f)^T + M_{one}^{-1}$$

Two-level hybrid Schwarz preconditioner

$$M_{hyb}^{-1} = I_c^f \, M_c^{-1} \, (I_c^f)^T + M_{one}^{-1} \left( \mathbf{I} - A_s \, I_c^f \, M_c^{-1} \, (I_c^f)^T \right)$$

Here $M_c$ is a coarse preconditioner and $I_c^f$ is a coarse to fine interpolation operator in space-time

# Theory for space-time additive Schwarz

Let $s$ be the window size, $U = (u^1, u^2, \cdots, u^s)^T$ and

$$A_{\tau,s}(U, V) = \tau \sum_{k=1}^{s} a(u^k, v^k) + \sum_{k=1}^{s}(u^k, v^k)$$

Let $P$ be the two-level space time additive Schwarz operator, and if the overlap is large enough, then

$$A_{\tau,s}(PU, PU) \leq CA_{\tau,s}(U, U)$$

$$A_{\tau,s}(U, PU) \geq cA_{\tau,s}(U, U)$$

Here $c$ and $C$ are independent of $\tau, h, H, H_c$ and $s$.

**Table :** The number of iterations for the two-level hybrid preconditioning with different mesh size, overlapping size, and number of processors. The coarse mesh is $32 \times 32$

| mesh-window size | overlap | number of processors | | | |
|---|---|---|---|---|---|
| | | 128 | 256 | 512 | 1024 |
| $249 \times 249 \times 8$ | 4 | 4.68 | 4.72 | 4.70 | 4.69 |
| $373 \times 373 \times 16$ | 6 | 5.83 | 5.87 | 5.90 | 5.96 |
| $497 \times 497 \times 32$ | 8 | 5.62 | 5.49 | 5.58 | 5.54 |

**Table :** Computing time (sec) per window size and number of iterations

| $497 \times 497$ | window size | | | | |
|---|---|---|---|---|---|
| | 4 | 8 | 16 | 32 | 64 |
| iter | 4.44 | 3.64 | 4.38 | 5.54 | 6.65 |
| time/window size | 79 | 64 | 80 | 126 | 167 |

# Time dependent optimization problems

Application areas: parabolic optimization problems, inverse problems, fluid control problems, etc
Current approach:

- *Optimization loop (sequential)*
  - *Solve a forward-in-time simulation problem*
    - *Loop in time (sequential)*
      *- Loop in space (parallel)*
    - *End loop in time*
  - *Solve a backward-in-time simulation problem*
    - *Loop in time (sequential)*
      *- Loop in space (parallel)*
    - *End loop in time*
  - *Solve objective equation*
- *End optimization loop*

# Observations and proposals

- There are two sequential loops inside the sequential loop $\longrightarrow$ the total number of sequential steps is the product
- The optimization loop is essentially a $3 \times 3$ (nonlinear) Gauss-Seidel $\longrightarrow$ slow and some sometimes difficult to converge $\longrightarrow$ many iterations

- On small machines, only parallelize the inner loop(s) and use Gauss-Seidel type methods for the outer loop
- On large machines, parallelize some of the outer loop(s) and replace Gauss-Seidel by Krylov-Schwarz

# A time dependent inverse problem

Consider a time dependent convection-diffusion equation

$$\begin{cases} \dfrac{\partial C}{\partial t} = \nabla \cdot (a(\mathbf{x})\nabla C) - \nabla \cdot (\mathbf{v}(\mathbf{x})C) + f(\mathbf{x}, t), & 0 < t < T\,,\ \mathbf{x} \in \Omega \\ C(\mathbf{x}, t) = p(\mathbf{x}, t), \quad \mathbf{x} \in \Gamma_1 \\ a(\mathbf{x})\dfrac{\partial C}{\partial \mathbf{n}} = q(\mathbf{x}, t), \quad \mathbf{x} \in \Gamma_2 \\ C(\mathbf{x}, 0) = C_0(\mathbf{x}), \quad \mathbf{x} \in \Omega \end{cases}$$

where $\Omega \in \mathbf{R}^3$, and $\partial\Omega = \Gamma_1 \bigcup \Gamma_2$

Goal: Compute $f(\mathbf{x}, t)$ with certain measured values of $C$

# PDE-constrained optimization problem

- We formulate the inverse problem as an output least-squares problem:

$$\text{Min}_{\mathbf{f}} J(\mathbf{f}) = \frac{1}{2}\int_0^T \int_\Omega A(\mathbf{x})(C(\mathbf{x},t) - C^\epsilon(\mathbf{x},t))^2 \, d\mathbf{x}dt + N_\beta(f)$$

  where $A(\mathbf{x})$ is the data range indicator function, $A(\mathbf{x}) = 1$ or $A(\mathbf{x}) = \sum_{i=1}^{N_s} \delta(\mathbf{x} - \mathbf{x}_i)$

- $N_\beta(\mathbf{f})$ denotes the Tikhonov regularization terms

$$N_\beta(f) = \frac{\beta_1}{2}\int_0^T \int_\Omega |f_t(\mathbf{x},t)|^2 d\mathbf{x}dt + \frac{\beta_2}{2}\int_0^T \int_\Omega |\nabla_{\mathbf{x}} f|^2 d\mathbf{x}dt$$

  Here $\beta_1$ and $\beta_2$ are regularization parameters

# KKT system

We use the optimize-then-discretize (OTD) approach, by introducing a corresponding Lagrange multiplier $G \in W^{1,p}(\Omega)$ and the following Lagrange functional:

$$\mathcal{J}(C, f, G) = \frac{1}{2}\int_0^T \int_\Omega A(\mathbf{x})(C(\mathbf{x}, t) - C^\epsilon(\mathbf{x}, t))^2 d\mathbf{x}dt \\ + N_\beta(f) + (G, L(C, f))$$

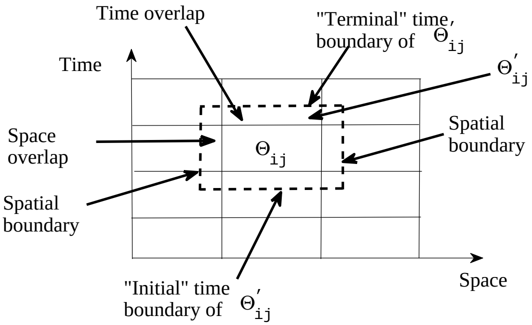
where $L(C, f)$ denotes the convection-diffusion operator. We compute the first-order optimality condition; i.e. the KKT system by taking the directional derivative of $\mathcal{J}$ as follows:

$$\begin{cases} \mathcal{J}_G(C, f, G)v = 0 \\ \mathcal{J}_C(C, f, G)w = 0 \\ \mathcal{J}_f(C, f, G)g = 0 \end{cases}$$

Multiplying a test function, and integrating by part, we obtain the weak form of the KKT system as follows:

$$\begin{cases} \left(\dfrac{\partial C}{\partial t}, v\right) + (a\nabla C, \nabla v) + (\nabla \cdot (\mathbf{v}C), v) \\ -(f(\mathbf{x}, t), v) - \langle q, w \rangle_{\Gamma_2} = 0 \\ \\ (\chi_{[T-\gamma, T]} A(\mathbf{x})(C(\mathbf{x}, t) - C^{\epsilon}(\mathbf{x}, t)), w) - \left(\dfrac{\partial G}{\partial t}, w\right) \\ +(a(\mathbf{x})\nabla G, \nabla w) + (\nabla \cdot (\mathbf{v}(\mathbf{x})w), G) = 0 \\ \\ \beta_1(f_t, g_t) + \beta_2(\nabla f, \nabla g) - (G, g) = 0 \end{cases}$$

# Space-time Schwarz



# Two-level space-time preconditioning

- We introduce $I_H^h$ as a linear interpolation operator from the coarse grid to the fine grid, and the restriction operator $I_h^H$ satisfying $I_h^H = (I_H^h)^T$
- We revise the preconditioner as a multiplicative type two-level Schwarz one:

$$\begin{cases} y = I_H^h M_H^{-1} I_h^H x \\ M_{two-level}^{-1} x = y + M_{ras}^{-1}(x - F_h y) \end{cases}$$

  where $F_h$ denotes the KKT matrix on the fine grid
- For the coarse solver, we use the one-level space-time additive Schwarz preconditioner with LU factorization as the subdomain solve

# Numerical examples

Example 1: Two Gaussian source inversion (with 1%, 5% and 10% measurement noise)
Example 2: Four Gaussian source inversion
Example 3: Eight Gaussian source inversion
The centers of the peaks move along the following blue and red curves respectively

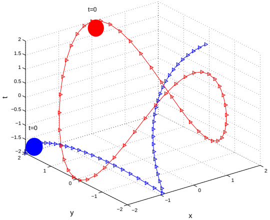



**Figure :** The two source moving traces in 3D $\Omega$

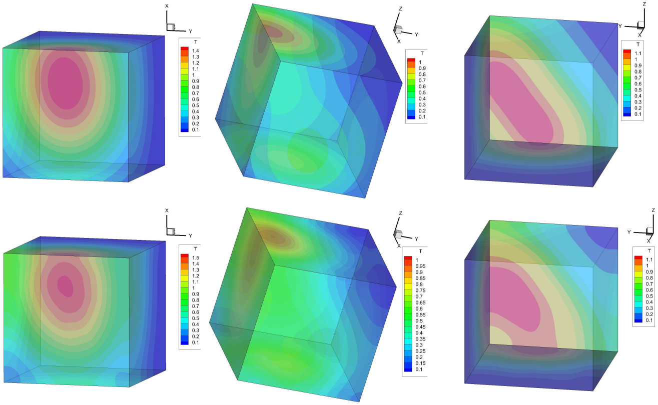



**Figure :** Ex1: The source reconstruction at three moments $t = 10/39, 20/39, 30/39$ with $14 \times 14 \times 14$ (bottom) sensors and 1% data noise, the results are comparable with the exact source distribution (top)

We add some noise to the data $\epsilon = 5\%$ and $\epsilon = 10\%$ and show the reconstruction results

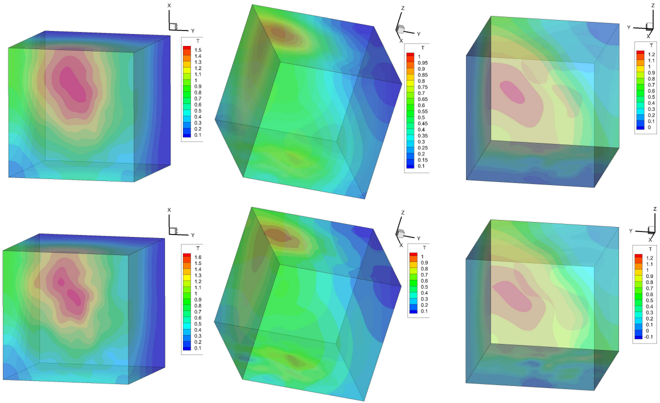



**Figure :** Ex1: The source reconstruction at noise level $\epsilon = 5\%$ (top) and $\epsilon = 10\%$ (bottom)

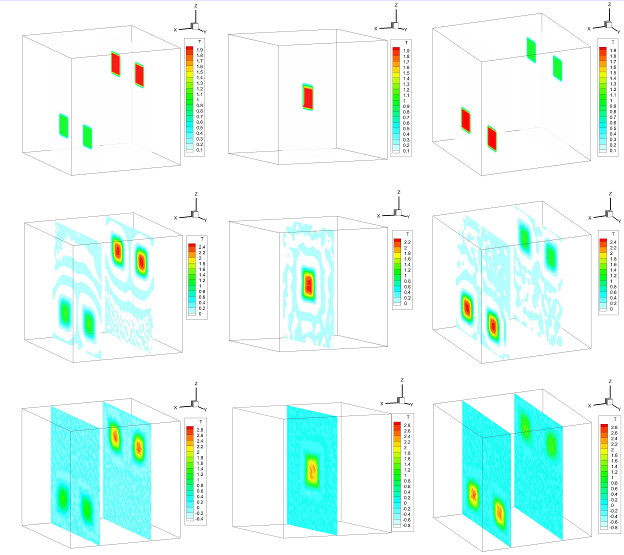
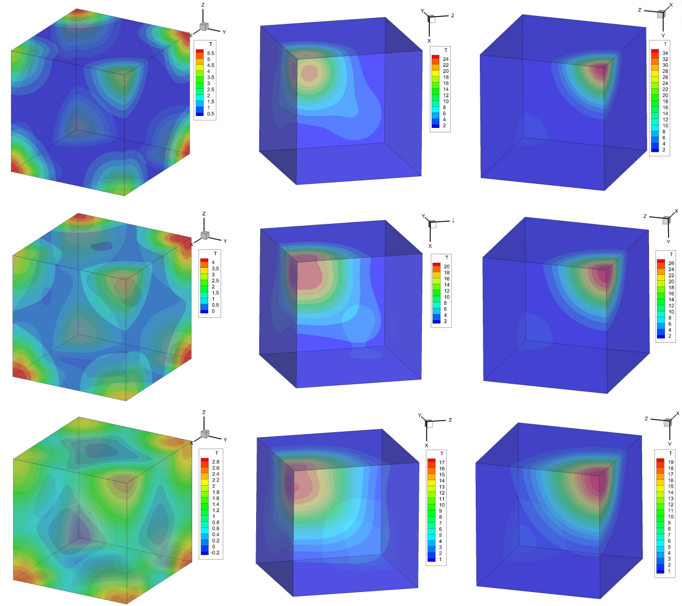


**Figure :** Ex2: The source reconstruction with $N_\beta(f)$ ($H^1$-$H^1$) regularization (top) and $\tilde{N}_\beta(f)$ ($H^1$-$L^2$) regularization (bottom) at three moments $t = 10/39, 20/39, 30/39$



T
5.5
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
T
24
22
20
18
16
14
12
10
8
6
4
2
T
34
32
30
28
26
24
22
20
18
16
14
12
10
8
6
4
2
T
4
3.5
3
2.5
2
1.5
1
0.5
0
T
20
18
16
14
12
10
8
6
4
2
T
26
24
22
20
18
16
14
12
10
8
6
4
2
T
2.8
2.6
2.4
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
T
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
T
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1

We still use a space mesh $49 \times 49 \times 49$ and time step 49, $DOF = 1.73 \times 10^7$, the ILU level and the overlap size on the fine and coarse grid are both set to be 0 and 1 respectively. We test with different number of processors for Example 1

| *np* | *level* | its | Time(sec) |
|---|---|---|---|
| 128 | 1 | 346 | 200.664 |
| | 2 | 83 | 47.812 |
| 256 | 1 | 343 | 127.035 |
| | 2 | 82 | 24.744 |
| 512 | 1 | 343 | 69.482 |
| | 2 | 82 | 16.461 |
| 1024 | 1 | 351 | 41.821 |
| | 2 | 85 | 10.132 |

With the same settings, we test with different number of processors for Example 2

| *np* | *level* | its | Time(sec) |
|---|---|---|---|
| 128 | 1 | 365 | 214.815 |
| | 2 | 85 | 47.072 |
| 256 | 1 | 363 | 152.334 |
| | 2 | 87 | 26.424 |
| 512 | 1 | 363 | 95.707 |
| | 2 | 101 | 19.453 |
| 1024 | 1 | 393 | 58.785 |
| | 2 | 100 | 11.352 |

With the same settings, we test with different number of processors for Example 3

| *np* | *level* | its | Time(sec) |
|---|---|---|---|
| 128 | 1 | 405 | 238.712 |
| | 2 | 93 | 57.244 |
| 256 | 1 | 408 | 145.213 |
| | 2 | 90 | 36.307 |
| 512 | 1 | 400 | 101.343 |
| | 2 | 100 | 18.611 |
| 1024 | 1 | 433 | 59.534 |
| | 2 | 104 | 15.815 |

# A comparison with a traditional reduced space method and a space-time reduced space method, for a 2D problem

| $np$ | $n_t$ | $n_x \times n_y$ | Solver | Time(sec) |
|---|---|---|---|---|
| 64 | 40 | $40 \times 40$ | FS | 12.064 |
| | | | RS(1) | 418.580 |
| | | | RS(2) | 125.484 |
| 128 | 80 | $80 \times 80$ | FS | 15.525 |
| | | | RS(1) | 682.794 |
| | | | RS(2) | 99.528 |
| 256 | 160 | $80 \times 80$ | FS | 23.736 |
| | | | RS(1) | 994.962 |
| | | | RS(2) | 200.543 |
| 512 | 320 | $160 \times 160$ | FS | 136.717 |
| | | | RS(1) | 7240.881 |
| | | | RS(2) | 1094.886 |

# Concluding remarks

- Moving some of the outer loops to the inside solver
  - increases the memory requirement
  - decreases the total compute time
  - increases the level of parallelism
  - increases the robustness
- Schwarz is capable of handling the added difficulties
- All examples are implemented using PETSc
- Some papers are available at

*www.colorado.edu/cs/users/cai*