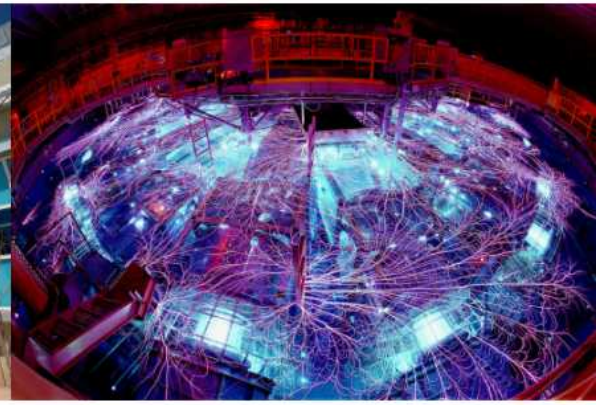
Exceptional service in the national interest

National Laboratories

SAND2019-10585C





Asynchronous Domain Decomposition Solvers

Parallel Solution Methods for Systems Arising from PDEs

Christian Glusa

Center for Computing Research, Sandia National Laboratories

September 16, 2019

U.S. DEPARTMENT OF ENERGY

oratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia,
neywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract

Motivation

Synchronous iterative methods:

- Alternate computation and communication
  $\rightarrow$ High network peak usage, low network usage during computation phases
- Are limited by global synchronization at scale, in particular under load imbalance due to
  - heterogeneous systems,
  - unbalanced local solve costs,
  - (transient) variability in communication cost

Asynchronous methods

- use the network more evenly,
- absorb load imbalance more easily,
- are more resilient to faults,
- sacrifice deterministic behavior.

Can asynchronous linear solvers be used at scale?

Restricted Additive Schwarz iteration (RAS)

Solve

$$\begin{cases} -\Delta u &= f \quad \text{in } \Omega \subset \mathbb{R}^d, \\ u &= 0 \quad \text{on } \partial\Omega. \end{cases}$$

for $k = 1, 2, \ldots$ **do**

Residual $r^{(k)} = f + \Delta u^{(k-1)}$

Solve in parallel

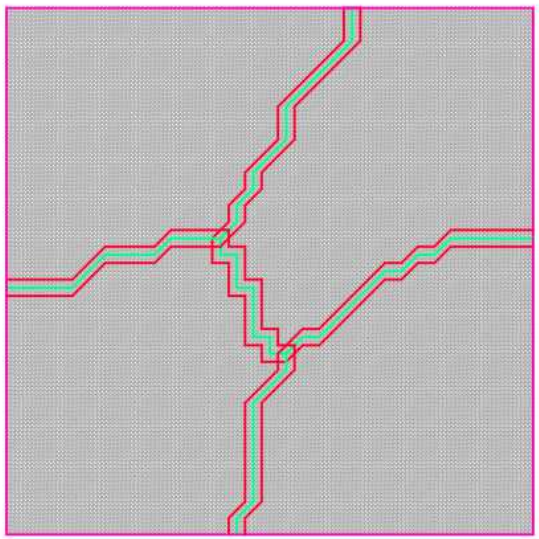
$$\begin{cases} -\Delta d_p^{(k)} &= r^{(k)} \quad \text{in } \Omega_p, \\ d_p^{(k)} &= 0 \quad \text{on } \partial\Omega_p. \end{cases}$$

Update

$$u^{(k)} = u^{(k-1)} + \sum_p \theta_p d_p^{(k)}$$

- $\{\theta_p\}$ is a partition of unity.
- Careful choice of θ_p eliminates need for communication in the update step.

Overlapping partition $\Omega = \cup_p \Omega_p$



Algebraic version of RAS

Solve

$$\boldsymbol{A}\vec{u} = \vec{f}.$$

For $k = 1, 2, \ldots$

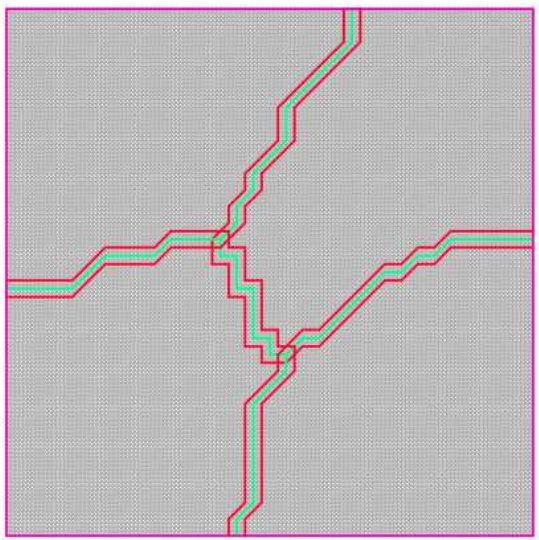
$$\vec{u}^{(k)} = \vec{u}^{(k-1)} + \sum_p \boldsymbol{R}_p^T \boldsymbol{D}_p \boldsymbol{A}_p^{-1} \boldsymbol{R}_p \left(\vec{f} - \boldsymbol{A}\vec{u}^{(k-1)} \right)$$

- $\boldsymbol{R}_p$ is the restriction to subdomain p, $\boldsymbol{A}_p = \boldsymbol{R}_p \boldsymbol{A} \boldsymbol{R}_p^T$.
- The diagonal matrices $\boldsymbol{D}_p$ satisfy discrete partition of unity property

$$\boldsymbol{I} = \sum_p \boldsymbol{R}_p^T \boldsymbol{D}_p \boldsymbol{R}_p.$$

- Similar to Additive Schwarz, but does not require damping to work as an iterative method.
- Unsymmetric iteration.

Overlapping partition $\Omega = \cup_p \Omega_p$



Communication pattern of RAS

$$\vec{u}^{(k)} = \underbrace{\vec{u}^{(k-1)} + \sum_p \boldsymbol{R}_p^T \boldsymbol{D}_p}_{\text{update}} \underbrace{\boldsymbol{A}_p^{-1}}_{\text{local solve}} \underbrace{\boldsymbol{R}_p \left(\vec{f} - \boldsymbol{A}\vec{u}^{(k-1)} \right)}_{\text{residual}}$$

- Computation of the residual requires neighborhood communication (halo values).
- Local solve and update are communication free.

Synchronous residual computation at iteration k:

$$\boldsymbol{R}_p \left(\vec{f} - \boldsymbol{A}\vec{u}^{(k-1)} \right) = \vec{r}_{p,\text{local}}^{(k-1)} + \sum_q \vec{r}_{q,\text{halo}}^{(k-1)}$$

Asynchronous residual computation at time t:

$$\boldsymbol{R}_p \left(\vec{f} - \boldsymbol{A}\vec{u}^{(t)} \right) = \vec{r}_{p,\text{local}}^{(t)} + \sum_q \vec{r}_{q,p,\text{halo}}^{(t)}$$

$\vec{r}_{q,p,\text{halo}}^{(t)}$ can be out of date since it is computed on subdomain q and needs to be sent across the network.

$\rightarrow$ Asynchronous iteration is not mathematically equivalent to the synchronous one.

$\rightarrow$ Special convergence theory is required. (See work by Szyld, Frommer, etc)

Realizing asynchronous communication: One-sided MPI

Sandia National Laboratories

- Allows MPI ranks to directly read/write exposed memory regions on other ranks.
- If the hardware (network interfaces) and MPI implementation support it, can be significantly faster than two-sided communication.

`MPI_Win_create`	Expose a memory region.
`MPI_Put`	Write into a window.
`MPI_Get`	Read from a window.
`MPI_Win_lock` / `MPI_Win_unlock`	Locks/unlocks a window, exclusive or shared access.
`MPI_Win_lock_all` / `MPI_Win_unlock_all`	Locks/unlocks a window on all processes.
`MPI_Win_flush`	Complete all outstanding operations.
⋮	

After testing the different options, we settled on `MPI_Win_lock_all` / `MPI_Win_unlock` and `MPI_Put`.

Convergence detection

Sandia National Laboratories

Convergence detection in synchronous iterations: $\|\vec{r}\| < \varepsilon$.
For asynchronous methods:

- No global iteration count, no synchronized global residual
- Norms and inner products introduce synchronization points.

Our ad-hoc approach for convergence detection:

- Compute norm of local residual vector $\|\vec{r}_p\|$
- Save to window on rank 0
- Rank 0 sums contributions and exposes result in a different window.
- Subdomains retrieve approximate residual norm and terminate if small enough.

Drawbacks:

- Local residual norms are not necessarily monotonically decreasing.
- Might iterate longer than necessary when rank 0 is busy.

Requirements for use at scale

Sandia National Laboratories

- No GMRES acceleration (inner products and norms are synchronization points)
  $\rightarrow$ Need to use the best possible fixed point iteration.
- One-level methods are not scalable
  $\rightarrow$ Asynchronous coarse grid solve for global information exchange
- Convergence is guaranteed.

Optimized RAS [1, 6, 7]

Subdomain problem in RAS

$$\begin{cases} -\Delta d_p^{(k)} &= r^{(k)} & \text{in } \Omega_p, \\ d_p^{(k)} &= 0 & \text{on } \partial\Omega_p. \end{cases}$$

The optimal choice would be

$$\begin{cases} -\Delta d_p^{(k)} &= r^{(k)} & \text{in } \Omega_p, \\ d_p^{(k)} &= 0 & \text{on } \partial\Omega_p \cap \partial\Omega, \\ \frac{\partial d_p^{(k)}}{\partial \vec{n}} - \Lambda_{\Omega_p^c}\left(d_p^{(k)}\right) &= 0 & \text{on } \partial\Omega_p \setminus \partial\Omega, \end{cases}$$

where $\Lambda_{\Omega_p^c}$ is the Poincaré-Steklov or Dirichlet-to-Neumann operator with respect to $\Omega_p^c = \Omega \setminus \Omega_p$. One can show convergence in at most $P - 1$ iterations!

oRAS0

But: evaluation of $\Lambda_{\Omega_p^c}$ is as expensive as the original problem.
Lowest order approximation

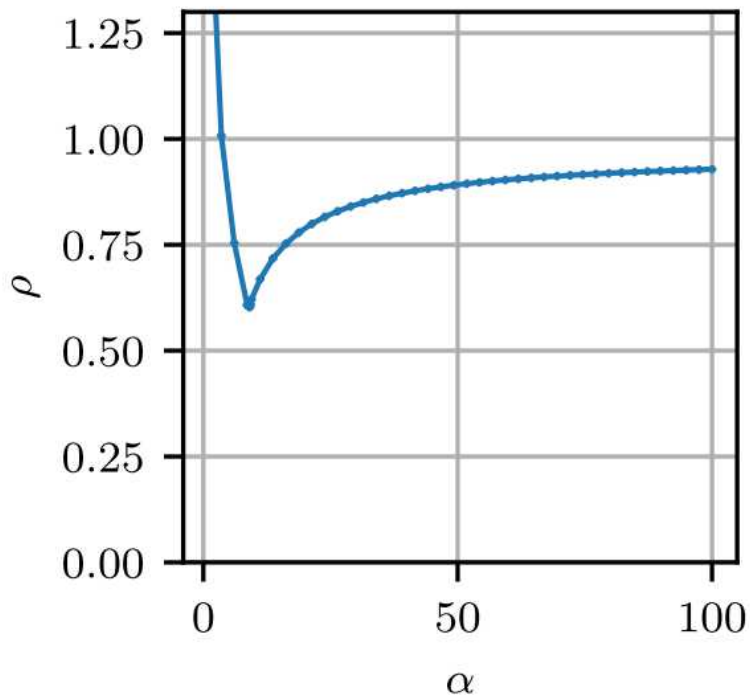
$$\Lambda_{\Omega_p^c}(d) \approx \alpha d, \qquad \alpha > 0,$$

i.e. Robin condition, leads to *optimized Restricted Additive Schwarz iteration of 0th order* (oRAS0).

- Replace subdomain matrices $\boldsymbol{A}_p \to \boldsymbol{A}_p + \alpha \boldsymbol{M}_p^{\partial\Omega_p}$, where $\boldsymbol{M}_p^{\partial\Omega_p}$ is the surface mass matrix.
- $\alpha \to \infty$ recovers usual RAS, $\alpha \to 0$ blows up.

How could we determine the optimal value of α efficiently?

- Fourier-type analysis (e.g. Gander [4], Garay & Szyld),
- Extrapolation from a coarser version of the problem,
- Optimization algorithms



Numerical Experiments with asynchronous oRAS0

- 2D Finite Element Poisson on unstructured mesh
- 64 subdomains
- # DoFs: $\approx$ 261,000
- Subdomain problem solved using direct factorization
- 5 samples per value of α
- Synchronous communication: non-blocking two-sided MPI
- Asynchronous communication: one-sided MPI

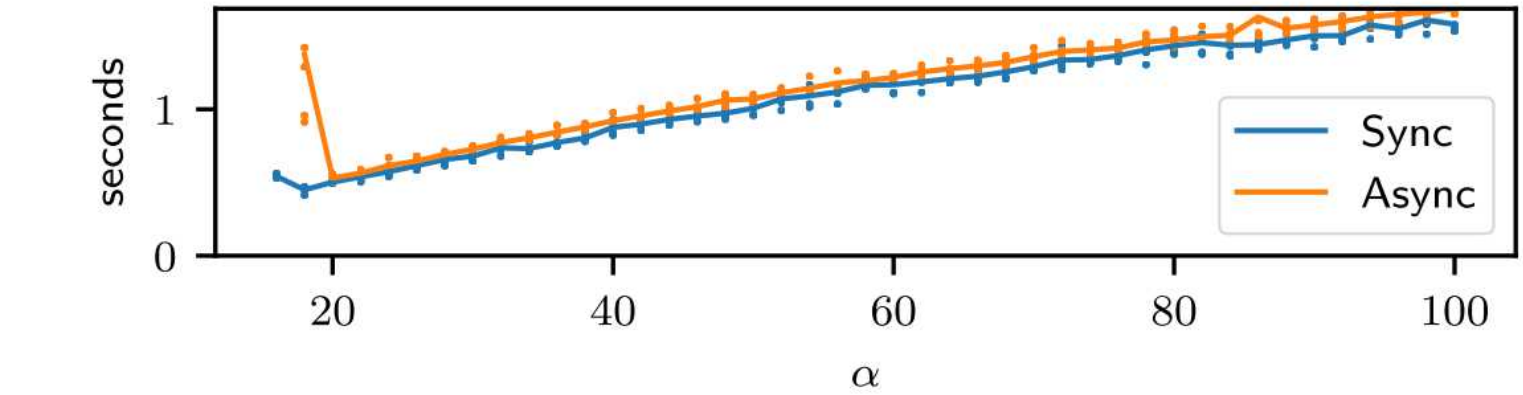
- Is the asynchronous method converging?
- How does the asynchronous iteration compare to the synchronous one? Observed contraction factor:

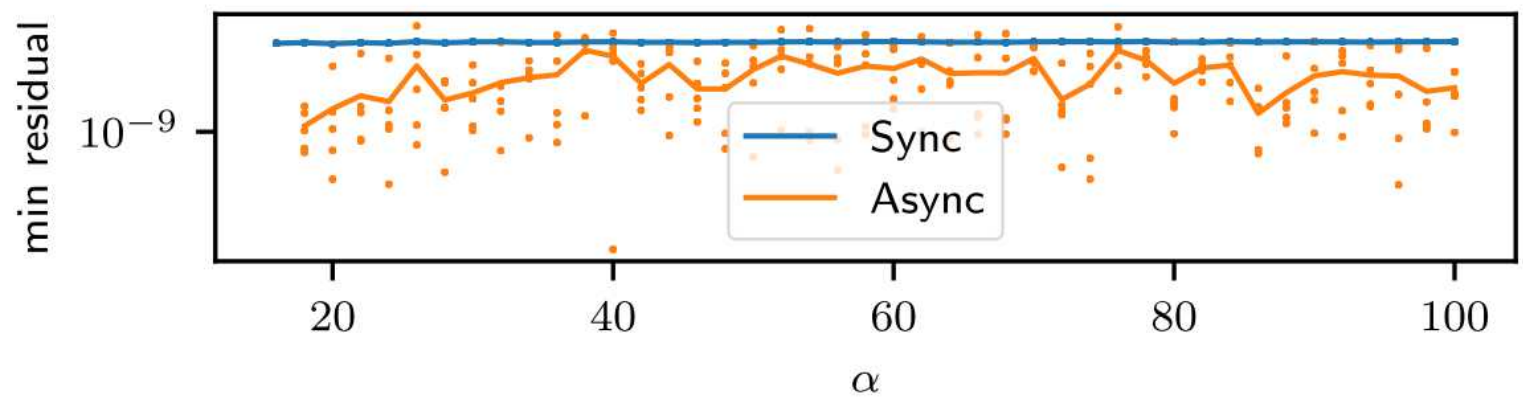
$$\hat{\rho} = \left(\frac{\text{Final residual}}{\text{Initial residual}} \right)^{\frac{\text{Average time per synchronous iteration}}{\text{Solve time}}}$$

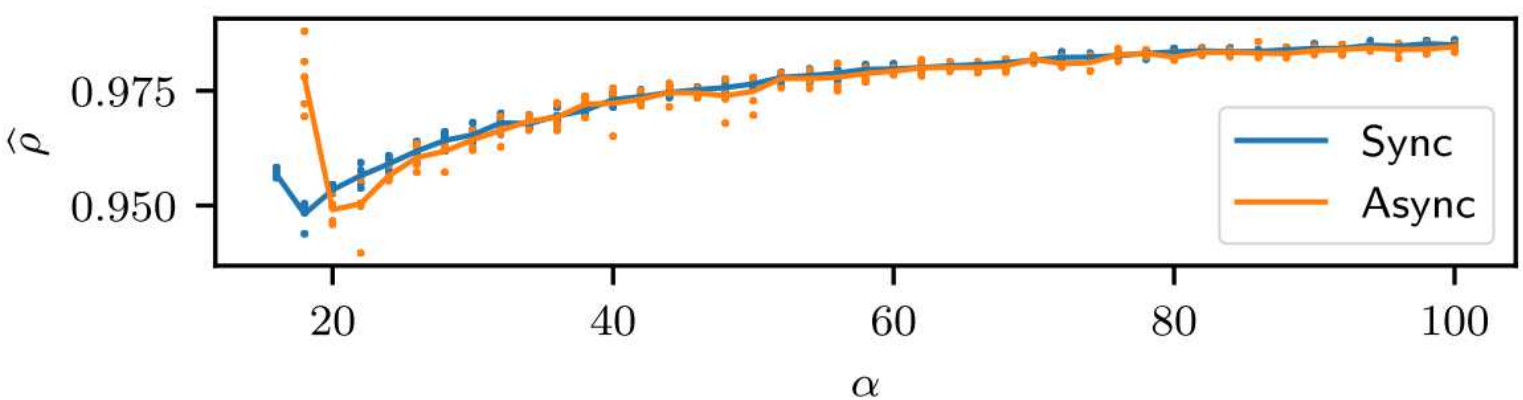
  (Reduces to contraction per iteration in synchronous case.)
- Is the dependence on α different from the synchronous case?

(See also results in Magoulès, Szyld, and Venet [5])

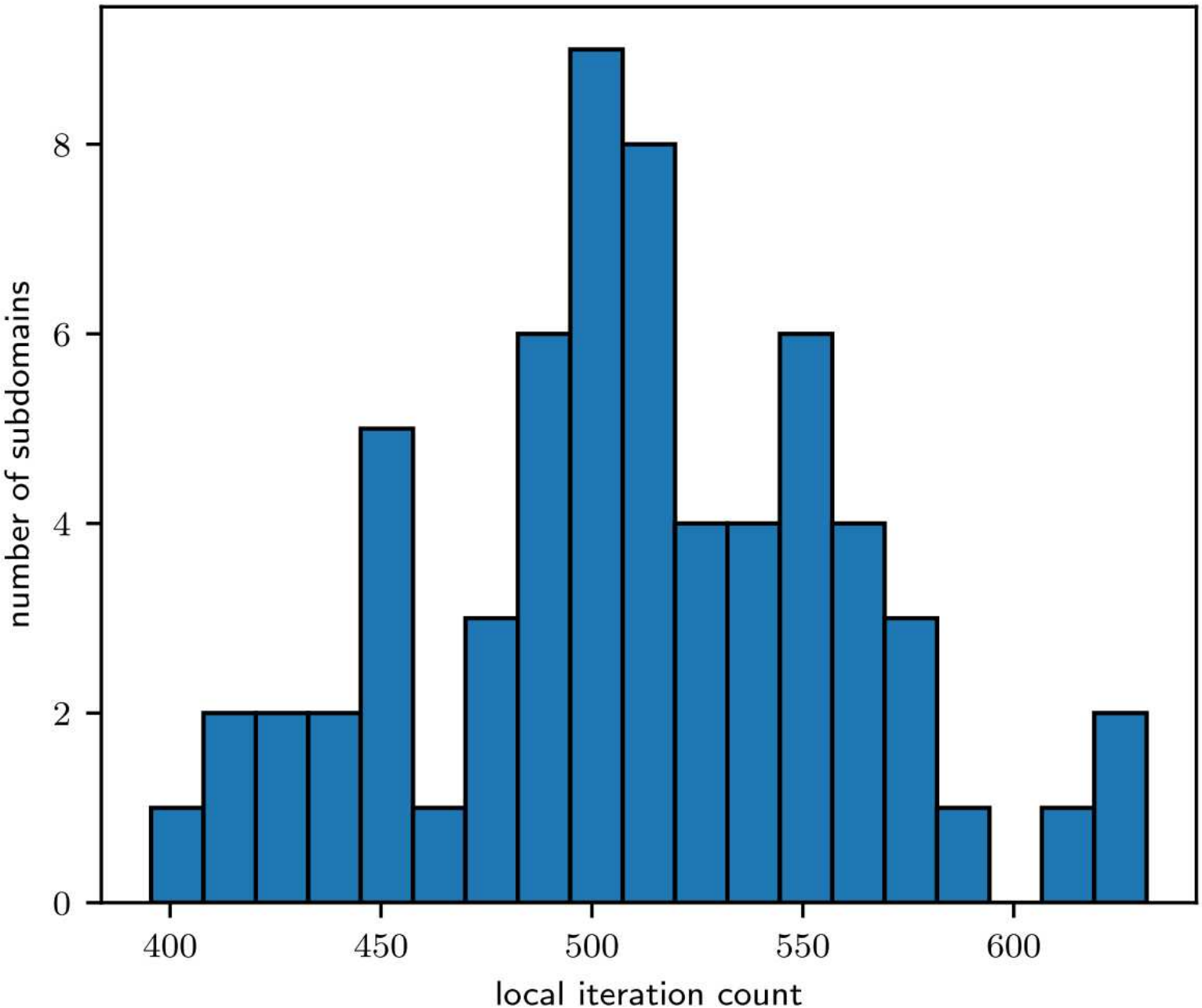
Numerical Experiments with oRAS0, sync vs async



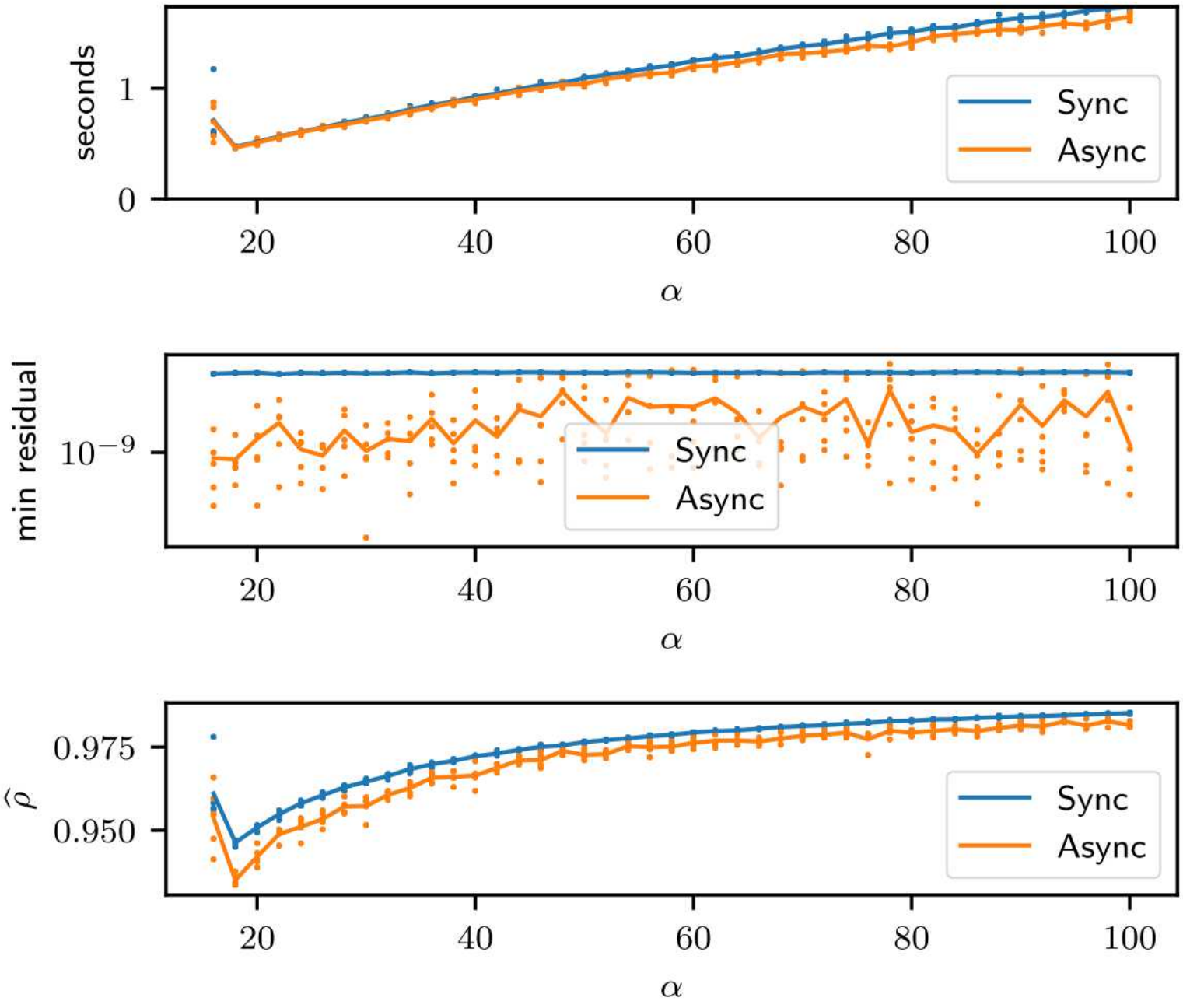




Local iteration counts for asynchronous oRAS0



Numerical Experiments with oRAS0, one subdomain 50% bigger



Synchronous Coarse Grid correction

- To achieve scalability, a mechanism of global information exchange is required.
- $\boldsymbol{A}_0$ coarser version of $\boldsymbol{A}$, and $\boldsymbol{R}_0$ restriction to coarse DoFs.
- Synchronous additive coarse grid correction to standard RAS:

$$\vec{u}^{(k)} = \vec{u}^{(k-1)} + \left(\frac{1}{2} \sum_p \boldsymbol{R}_p^T \boldsymbol{D}_p \boldsymbol{A}_p^{-1} \boldsymbol{R}_p + \frac{1}{2} \boldsymbol{R}_0^T \boldsymbol{A}_0^{-1} \boldsymbol{R}_0 \right) \left(\vec{b} - \boldsymbol{A}\vec{u}^{(k-1)} \right)$$

(Similar for oRAS0, but lack of a strategy for selection of α, use RAS for now.)

- Communication pattern

$$\underbrace{\frac{1}{2}\boldsymbol{R}_0^T}_{\text{prolongation}} \quad \underbrace{\boldsymbol{A}_0^{-1}}_{\text{coarse grid solve}} \quad \underbrace{\boldsymbol{R}_0 \left(\vec{b} - \boldsymbol{A}\vec{u}^{(k-1)} \right)}_{\text{residual}}$$

  - Residual *and* prolongation require communication between *all* subdomains and coarse grid.
  - Coarse grid solve can be distributed itself and internally require communication, but we keep it synchronous.

Asynchronous coarse grid cannot be handled in the same way as the subdomains.

Asynchronous Coarse Grid correction

On coarse grid

- **while** *not converged* **do**
  - **if** $\texttt{RHSisReady}_0[p]$ $\forall$*subdomains p* **then**
    - Solve $\mathbf{A}_0\texttt{x} = \texttt{rhs}_0$
    - **for** *subdomain p* **do**
      - $\texttt{RHSisReady}_0[p] \leftarrow \texttt{False}$
      - $\texttt{canWriteRHS}_p \leftarrow \texttt{True}$
      - $\texttt{solution}_p \leftarrow \texttt{x}$
      - $\texttt{solutionIsReady}_p \leftarrow \texttt{True}$

On subdomain *p*

- **while** *not converged* **do**
  - Compute residual asynchronously
  - **if** $\texttt{canWriteRHS}_p$ **then**
    - Compute local part of coarse grid residual $\texttt{rhs}_p$
    - $\texttt{rhs}_0 \leftarrow \texttt{rhs}_0 + \texttt{rhs}_p$
    - $\texttt{canWriteRHS}_p \leftarrow \texttt{False}$
    - $\texttt{RHSisReady}_0[p] \leftarrow \texttt{True}$
  - Solve local problem
  - **if** $\texttt{solutionIsReady}_p$ **then**
    - Update local iterate with $\texttt{solution}_p$ and solution of local problem (weights $(1/2, 1/2)$)
    - $\texttt{solutionIsReady}_p \leftarrow \texttt{False}$
  - **else**
    - Update local iterate with solution of local problem (weight 1)

Types of variables:

`window on remote rank`

`window on local rank`

`local variable`

Convergence of the method

For process $1 \leq p \leq P$ and time instances $n \geq 1$, define the asynchronous iteration

$$\vec{u}_{p,n} = \begin{cases} \mathcal{T}_{p,n}\left(\vec{u}_{1,n}^{(p)}, \ldots, \vec{u}_{P,n}^{(p)}\right) & \text{if } p \in \sigma_n, \\ \vec{u}_{p,n-1} & \text{if } p \notin \sigma_n, \end{cases}$$

where

- σ_n are the sets of processes performing an update at time n,
- $\mathcal{T}_{p,n}$ are the rules for updating the unknowns on process p at time n, and
- $\vec{u}_{q,n}^{(p)}$ is the data received by process p from process q available at time n.

Theorem (Mimicking Frommer&Szyld [3, 2])

The two-level method converges, provided that $\mathbf{A}$ *is a non-singular M-matrix and provided that*

- *causality principle holds,*
- *no process will ever stop updating its components,*
- *no process will ever stop receiving data.*

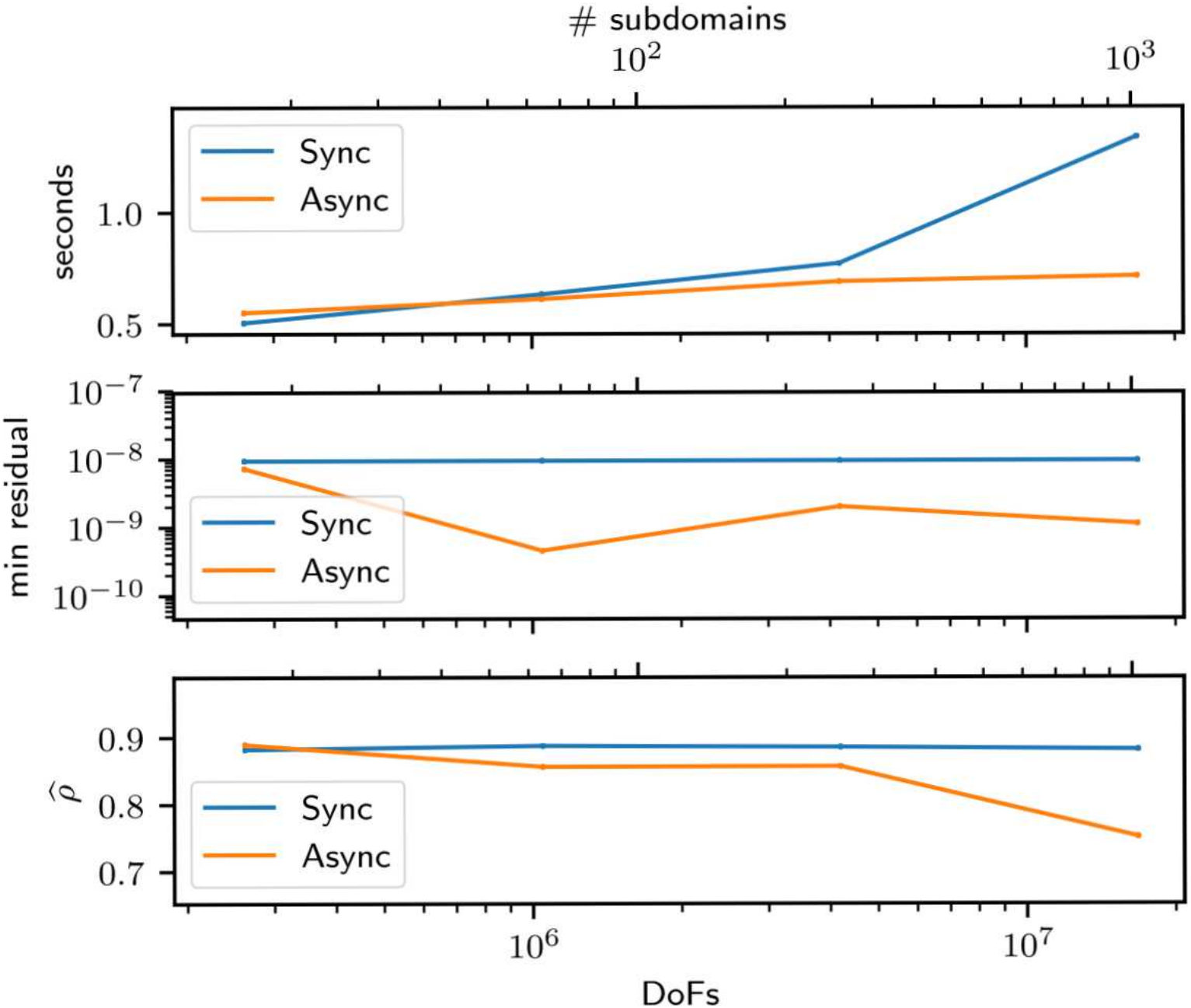
2D Numerical Experiments, RAS w/ additive coarse grid

- 2D Finite Element Poisson
- Weak scaling experiment:

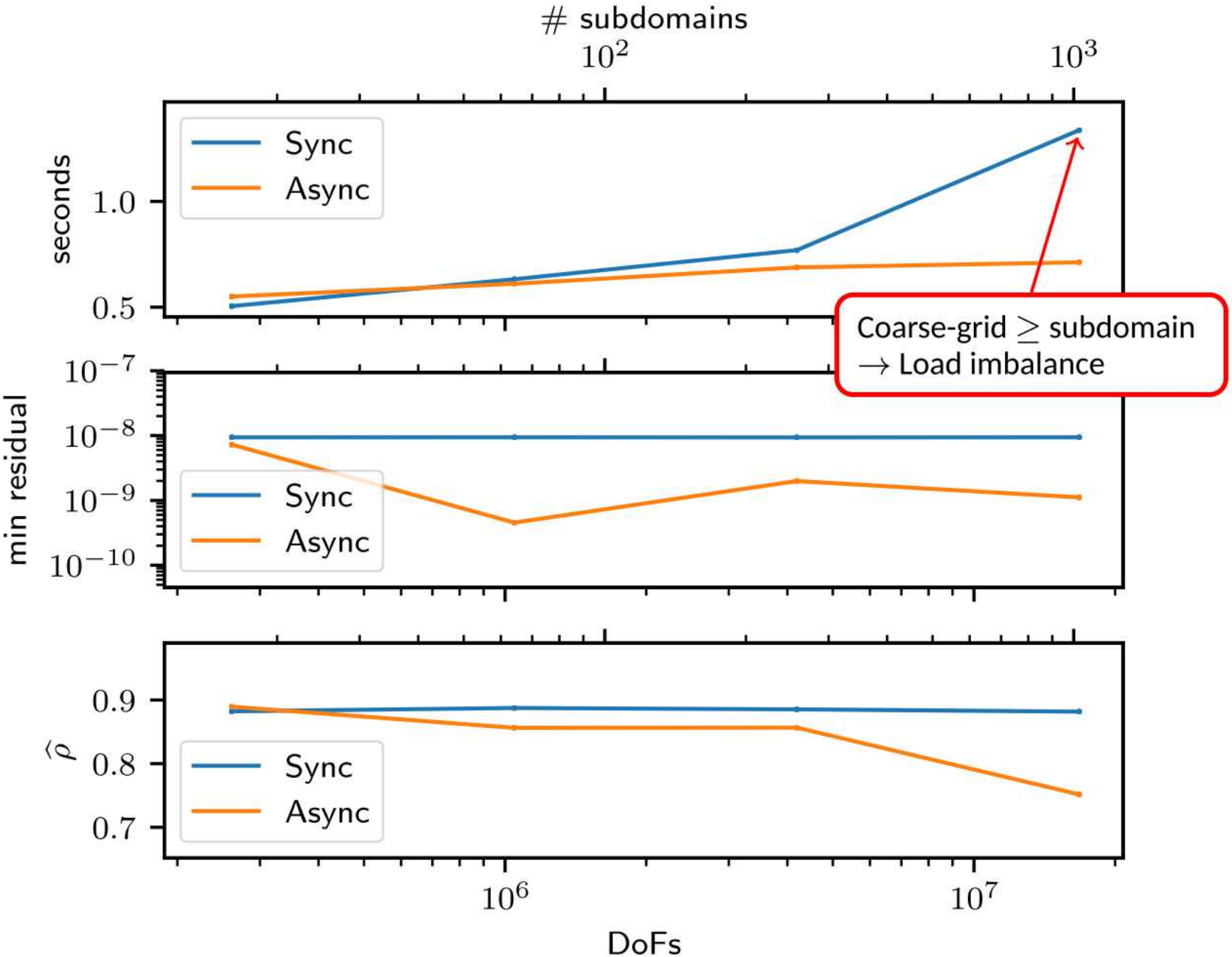
subdomains:	16	64	256	1024
unknowns:	256k	1M	4M	16M

- # DoFs per subdomain: $\approx$ 20,000
- # DoFs for coarse grid: $\approx 16 \times$ #subdomains
- Subdomain and coarse grid solves using direct factorization.
- Coarse grid solve on a single MPI rank.
- Haswell partition on Cori @ NERSC, 1 rank/core

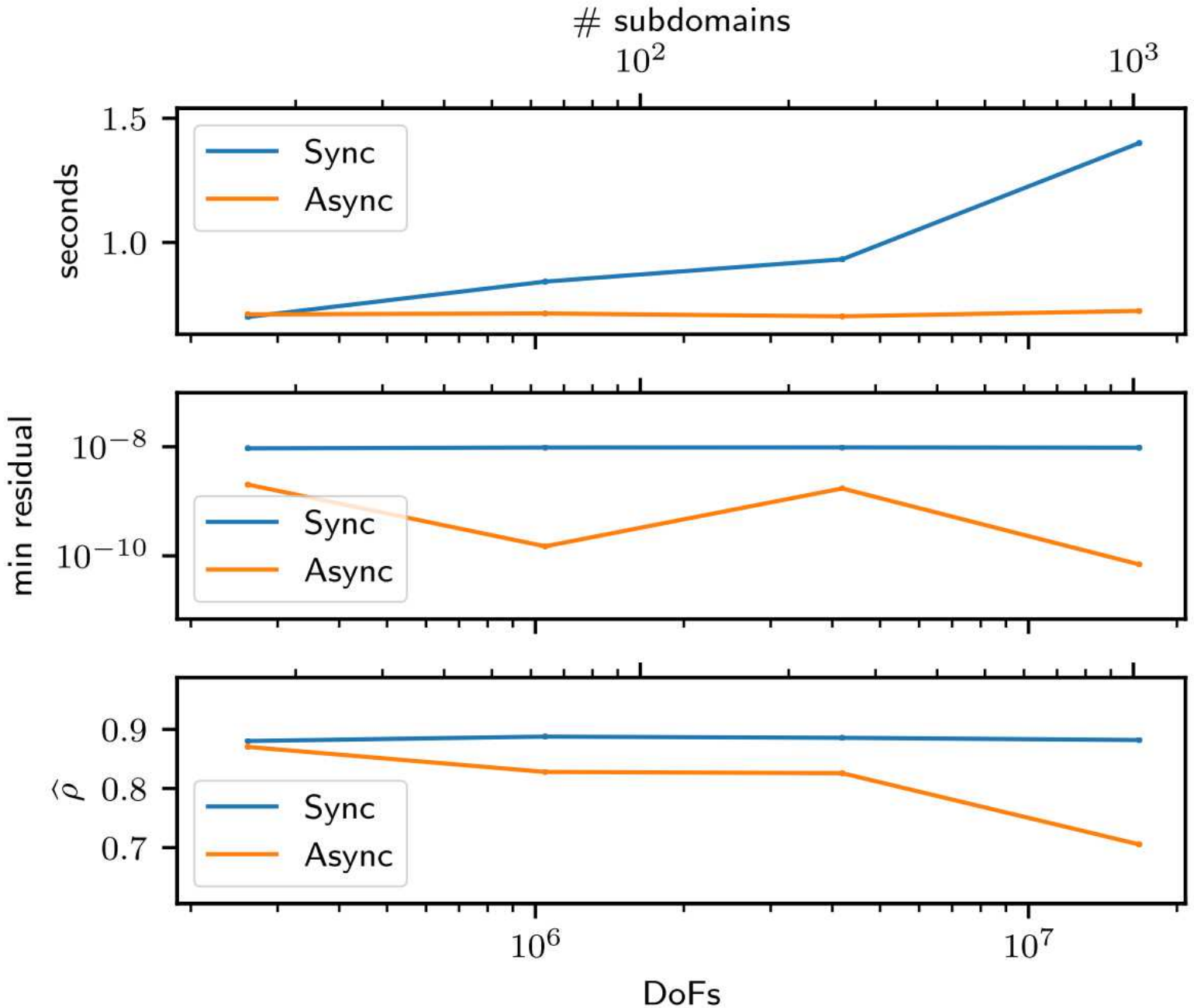
2D, RAS w/ additive coarse grid correction, sync vs async



2D, RAS w/ additive coarse grid correction, sync vs async



2D, RAS w/ additive coarse grid correction, one subdomain 50% bigger



3D Numerical Experiments, RAS w/ additive coarse grid

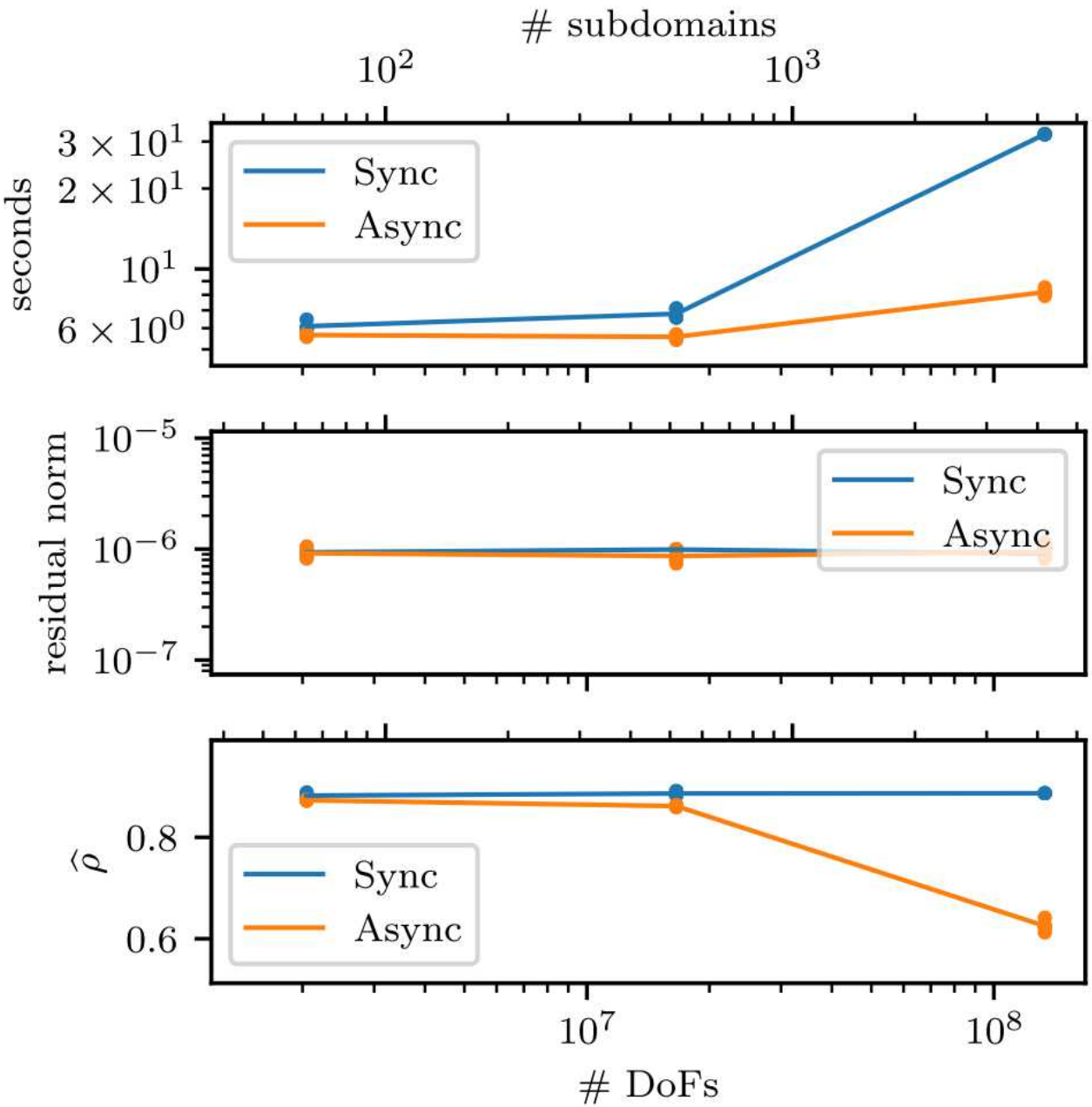
- 3D Poisson equation on a box, structured partitioning.
- Weak scaling experiment:

subdomains:	64	512	4096
unknowns:	2M	16M	128M

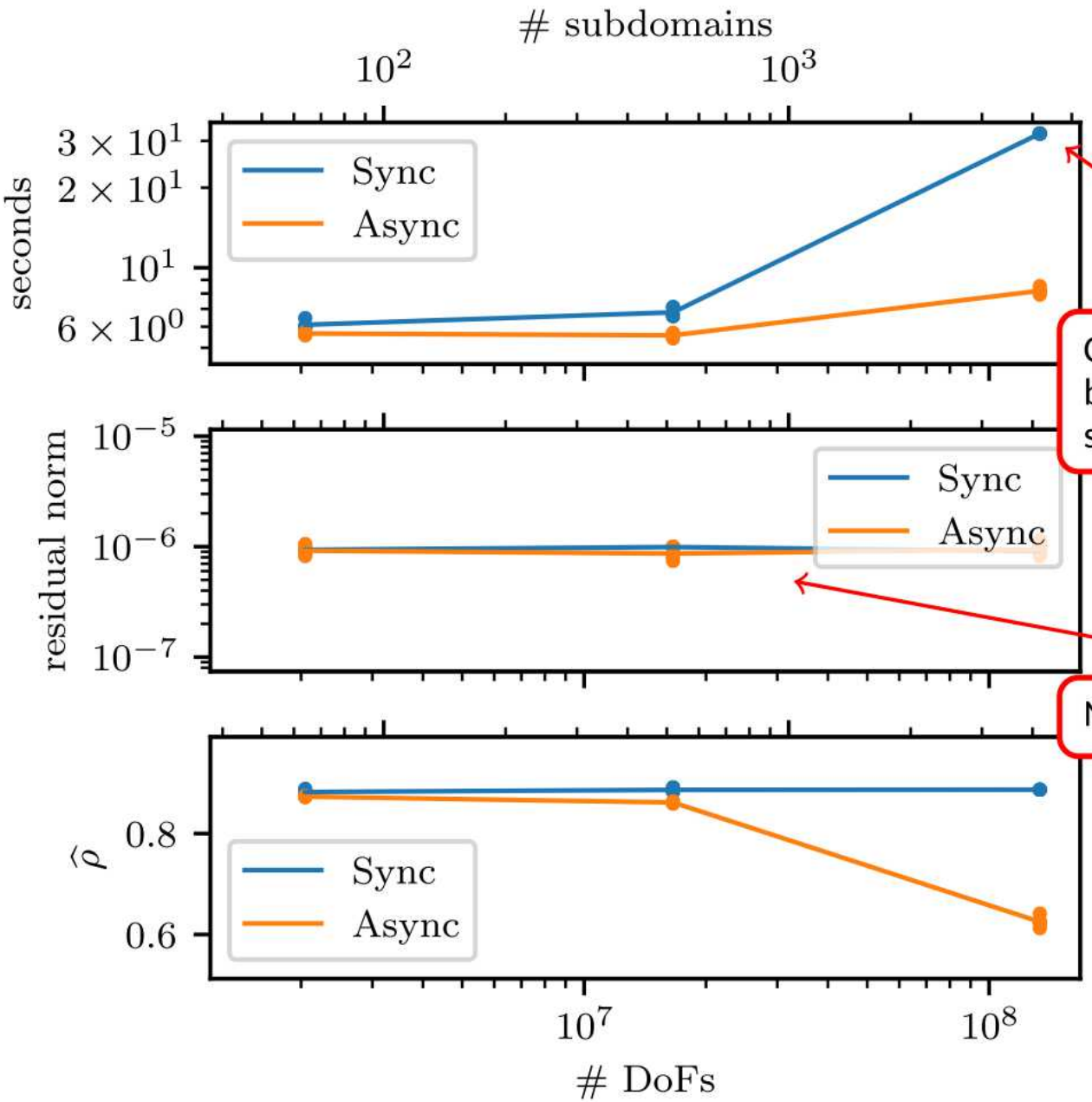
- # DoFs per subdomain: $\approx$ 40k
- # DoFs for coarse grid: $\approx$ one per subdomain
- Subdomain solves using conjugate gradients with incomplete Cholesky preconditioning, relative tolerance $1/10$
- Coarse solve using a single multigrid V-cycle.
- Coarse solve is distributed if the coarse problem is large.
- Haswell partition on Cori @ NERSC, 1 rank/core

Built-in possibility of load imbalance: local iterative solves

3D, RAS w/ additive coarse grid correction



3D, RAS w/ additive coarse grid correction

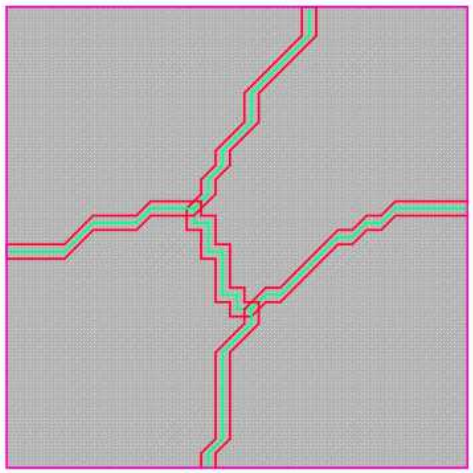


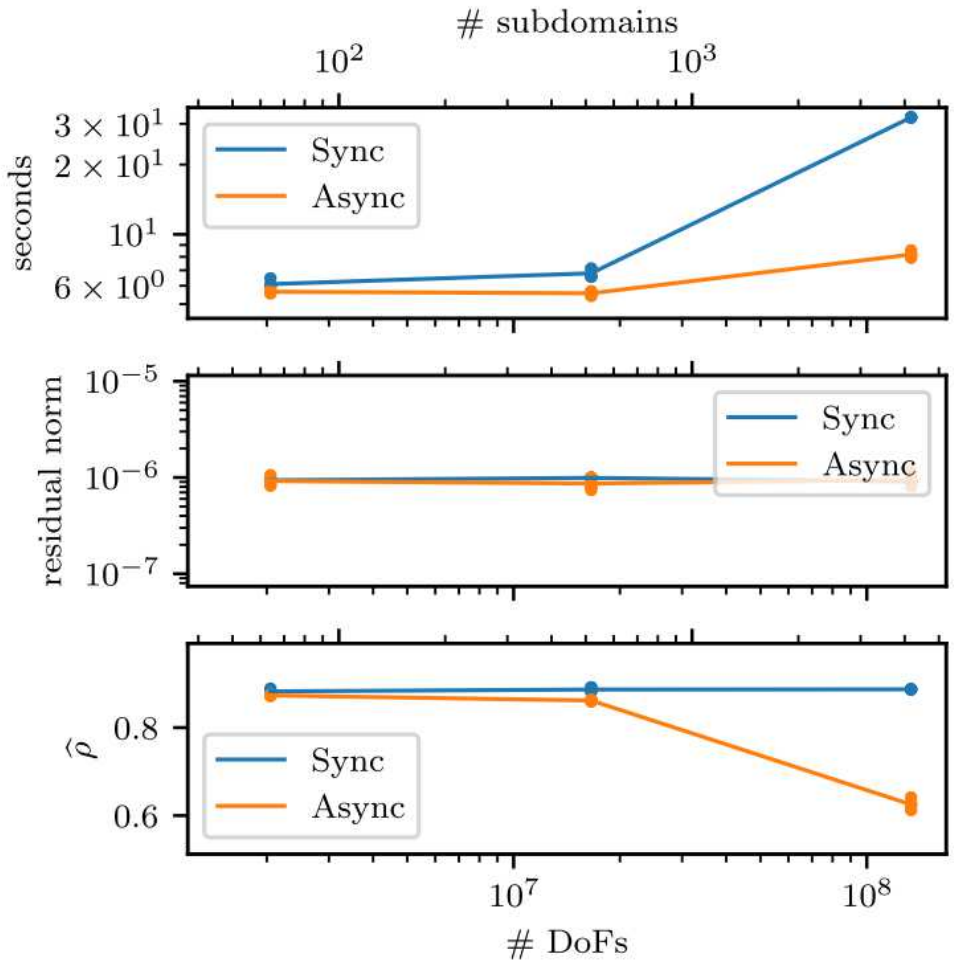
Coarse-grid $\ll$ subdomain, but large communication stencil

No more over-convergence

Conclusion and Outlook

- Asynchronous solvers can compete with their synchronous equivalents.
- Load imbalance is absorbed more easily.
- Need to pick a strategy for selection of optimization parameter α if we want to use oRAS0 (and higher order variants).
- Try asynchronous solvers on some application problems.
  (Trilinos implementation in the works)





References I


Philippe Chevalier and Frédéric Nataf. "Symmetrized method with optimized second-order conditions for the Helmholtz equation". In: *Contemporary Mathematics* 218 (1998), pp. 400–407.


Andreas Frommer and Daniel B. Szyld. "An Algebraic Convergence Theory for Restricted Additive Schwarz Methods Using Weighted Max Norms". In: *SIAM Journal on Numerical Analysis* 39 (2001), pp. 463–479.


Andreas Frommer and Daniel B. Szyld. "On asynchronous iterations". In: *Journal of Computational and Applied Mathematics* 123.1 (2000), pp. 201–216.


Martin J Gander. "Optimized Schwarz Methods". In: *SIAM Journal on Numerical Analysis* 44.2 (2006), pp. 699–731.


Frédéric Magoulès, Daniel B. Szyld, and Cédric Venet. *Asynchronous Optimized Schwarz Methods with and without Overlap*. Tech. rep. 15-06-29. Department of Mathematics, Temple University, June 2015.


Frédéric Nataf. "Absorbing boundary conditions in block Gauss–Seidel methods for convection problems". In: *Mathematical Models and Methods in Applied Sciences* 6.04 (1996), pp. 481–502.

References II

Frédéric Nataf and Francis Nier. “Convergence rate of some domain decomposition methods for overlapping and nonoverlapping subdomains”. In: *Numerische Mathematik* 75.3 (1997), pp. 357–377.