A Barotropic Mode Solver for MPAS-Ocean using ForTrilinos

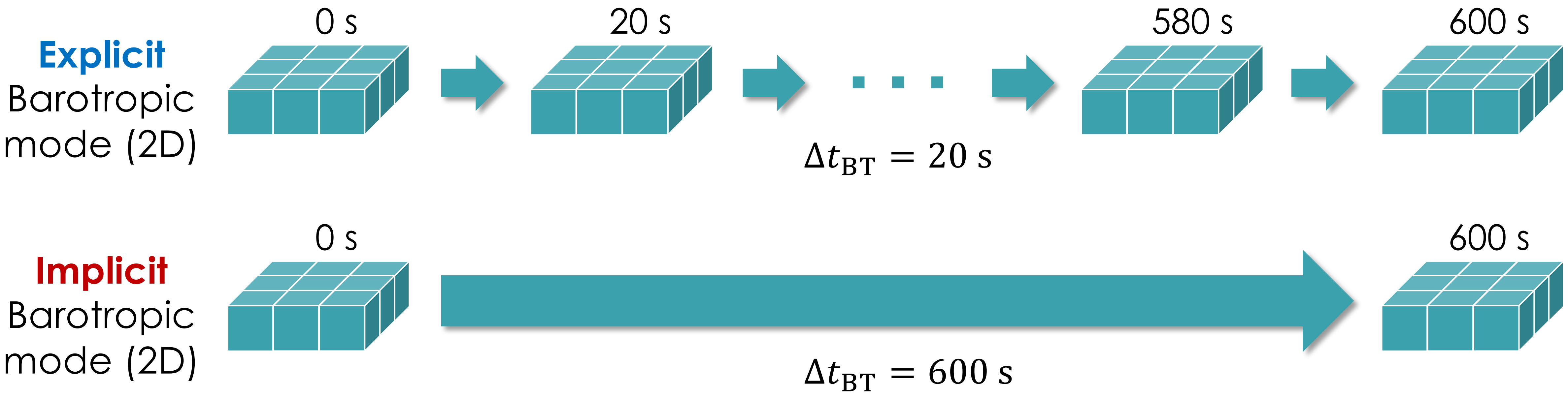
Hyun-Gyu Kang[1], Katherine J. Evans[1], Philip W. Jones[2], Mark R. Petersen[2], Andrew G. Salinger[3], Raymond S. Tuminaro[4], and Andrey Prokopenko[1]

[1]Oak Ridge National Laboratory
[2]Los Alamos National Laboratory
[3]Sandia National Laboratories (NM)
[4]Sandia National Laboratories (CA)

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

Background

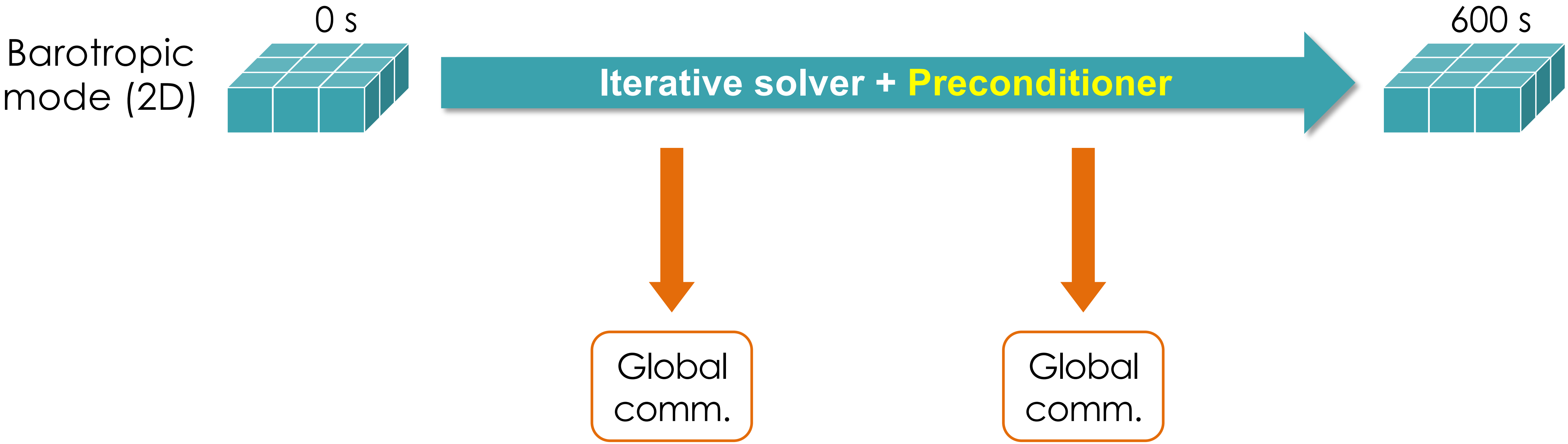
- **Split-explicit subcycling scheme** (current scheme in MPAS-O)
  - 3D baroclinic mode can be stepped with a long time step size. (≈2 m s^{-1} of wave speed)
  - 2D barotropic mode should be stepped with a short time step size. (≈200 m s^{-1} of wave speed)
- **Split semi-implicit method**
  - 2D barotropic and 3D baroclinic modes can be stepped with long time step size together, since the implicit method is **unconditionally stable**.



Background

- **Split semi-implicit method**
  - Reducing the number of iteration as small as possible using a good preconditioner

➔ More scalable and efficient semi-implicit barotropic mode solver



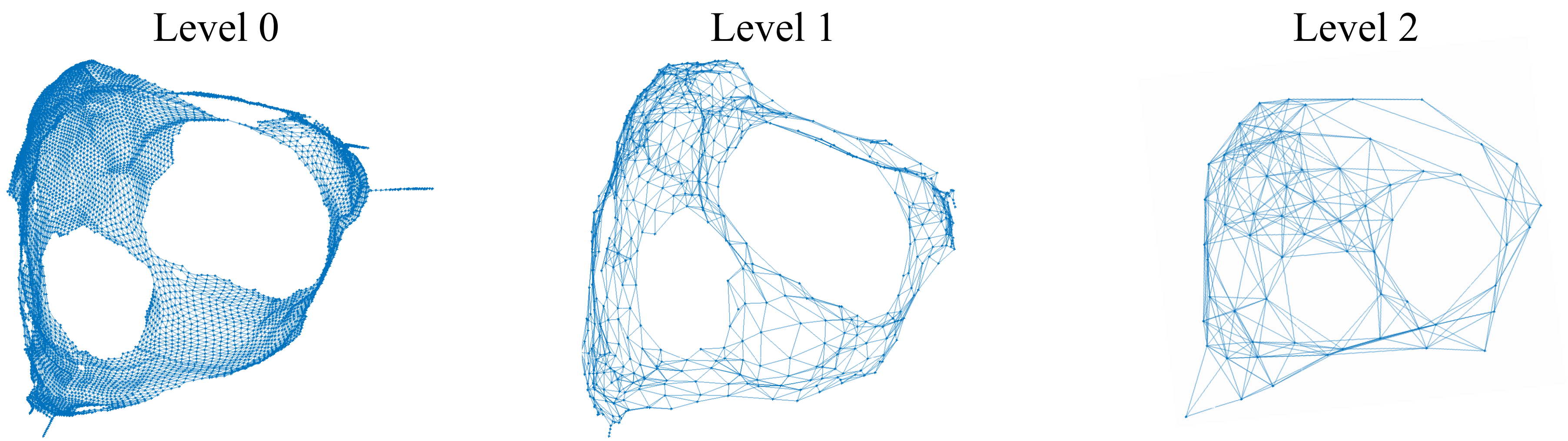
Background

- ForTrilinos
  - Object-oriented Fortran interfaces to Trilinos C++ packages
    - Modelers using modern Fortran are able to provide their codes the capability to use solvers and other capabilities on exascale machines via a straightforward infrastructure that accesses Trilinos.
  - An open-source software library
  - The new implementation using SWIG code generation
  - Developed and maintained by:
    - Seth Johnson (ORNL) and Andrey Prokopenko (ORNL) (https://fortrilinos.readthedocs.io/en/latest/)
  - A variety of choice of linear & nonlinear solvers and preconditioners
    - Users can use several combinations for a solver and a preconditioner to find the best set for their model.

- **Research purpose**
  - In the present study, we implement the semi-implicit solver for MPAS-O using ForTrilinos as a competitor of the current explicit-subcycling scheme.

Design of a semi-implicit barotropic mode solver

- **Semi-implicit formulation**
  - Crank-Nicolson (CN) method
    - Second-order in time, unconditionally stable
    - Backward time stepping for the 1st time step only to damp spurious oscillations from CN method
- **Iterative solver**
  - Preconditioned s-step CG provided by Belos linear solver framework in ForTrilinos
    - More scalable algorithm compared to classic CG
    - Solver tolerance: 10^{-8} → carefully chosen by using energy conservation error
- **Preconditioner**
  - Algebraic multigrid preconditioner provided by MueLu multigrid solver framework



ForTrilinos interface in MPAS-O

- **Call-tree structure of the ForTrilinos in MPAS-O**

MPAS-O code

ForTrilinos interface

mpas_ocn_time_integration_si.F

subroutine **ocn_time_integration_si**

call **ocn_time_integration_btrmode**

mpas_ocn_fortrilinos_imp_mod.F

subroutine **ocn_time_integration_imp_btrmode**

Setup a coefficient matrix ($\mathbf{A}$)

1) InsertGlobalValues → Initial only
2) sumIntoGlobalValues → Each time step

Read Parameterlist → Initial only

Initialize a handle

Setup the problem

Calculate initial residual

Solve the problem ($\mathbf{Ax} = \mathbf{b}$)

Check the solution

Get solution ($\mathbf{x}$)

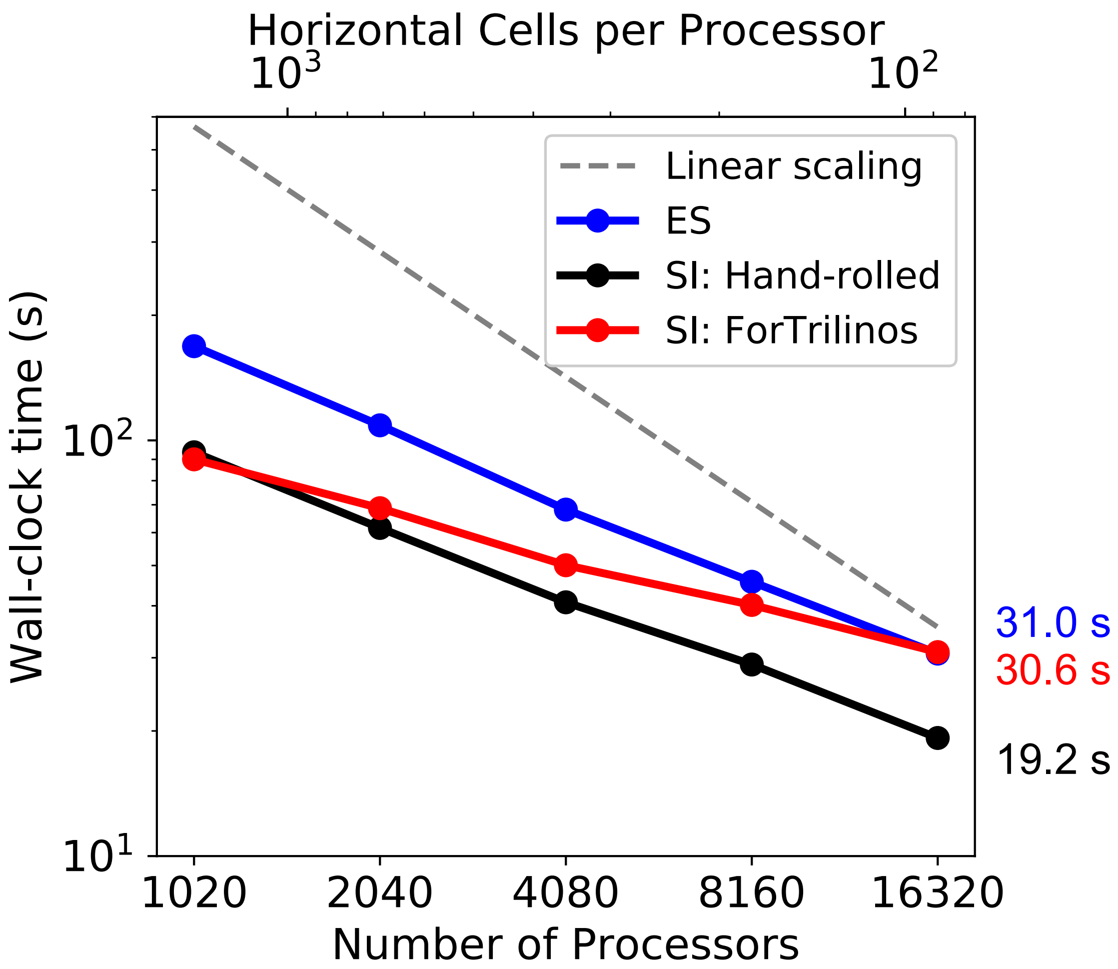
Clean up

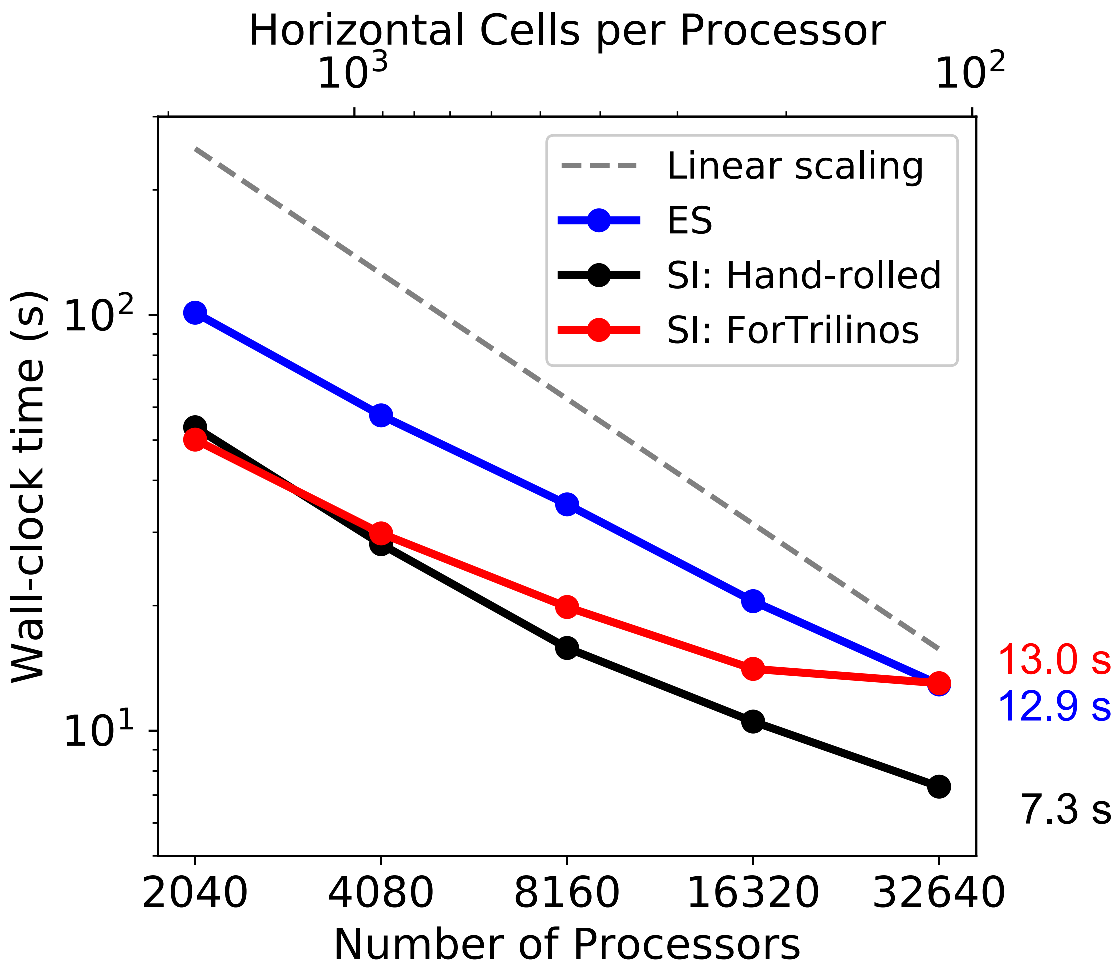
```
! Step 0: create a handle
solver_handle = TrilinosSolver() !; FORTRILINOS_CHECK_IERR()

! Step 1: initialize a handle
call solver_handle%init(comm) !; FORTRILINOS_CHECK_IERR()

! Step 2: setup the problem
call solver_handle%setup_matrix(A) !; FORTRILINOS_CHECK_IERR()

! Step 3: setup the solver
call solver_handle%setup_solver(plist) !; FORTRILINOS_CHECK_IERR()
```

```
call solver_handle%solve(B, X) !; FORTRILINOS_CHECK_IERR()
```

Results

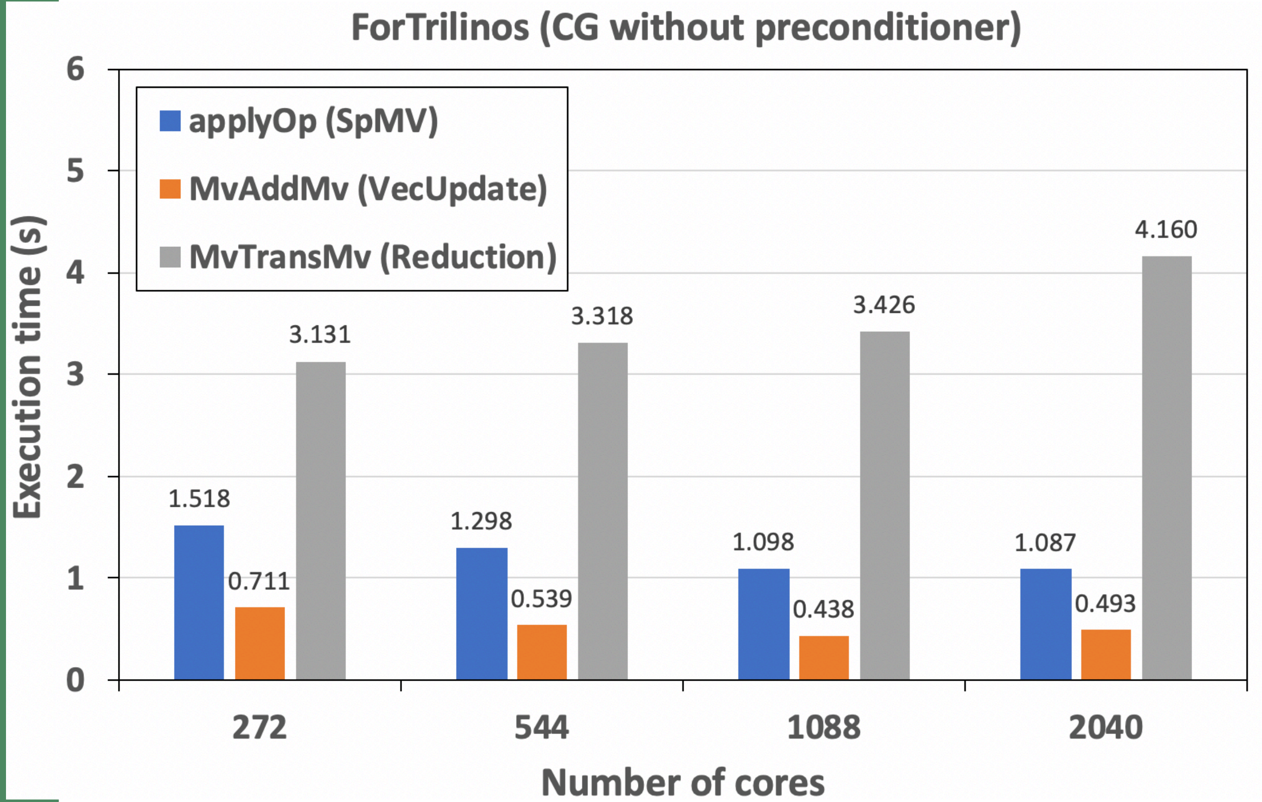
- **Strong scaling results**
  - RRS30to10 test case (1,444,561 cells; one day integration)
  - RRS18to6 test case (3,693,225 cells; six hours integration)

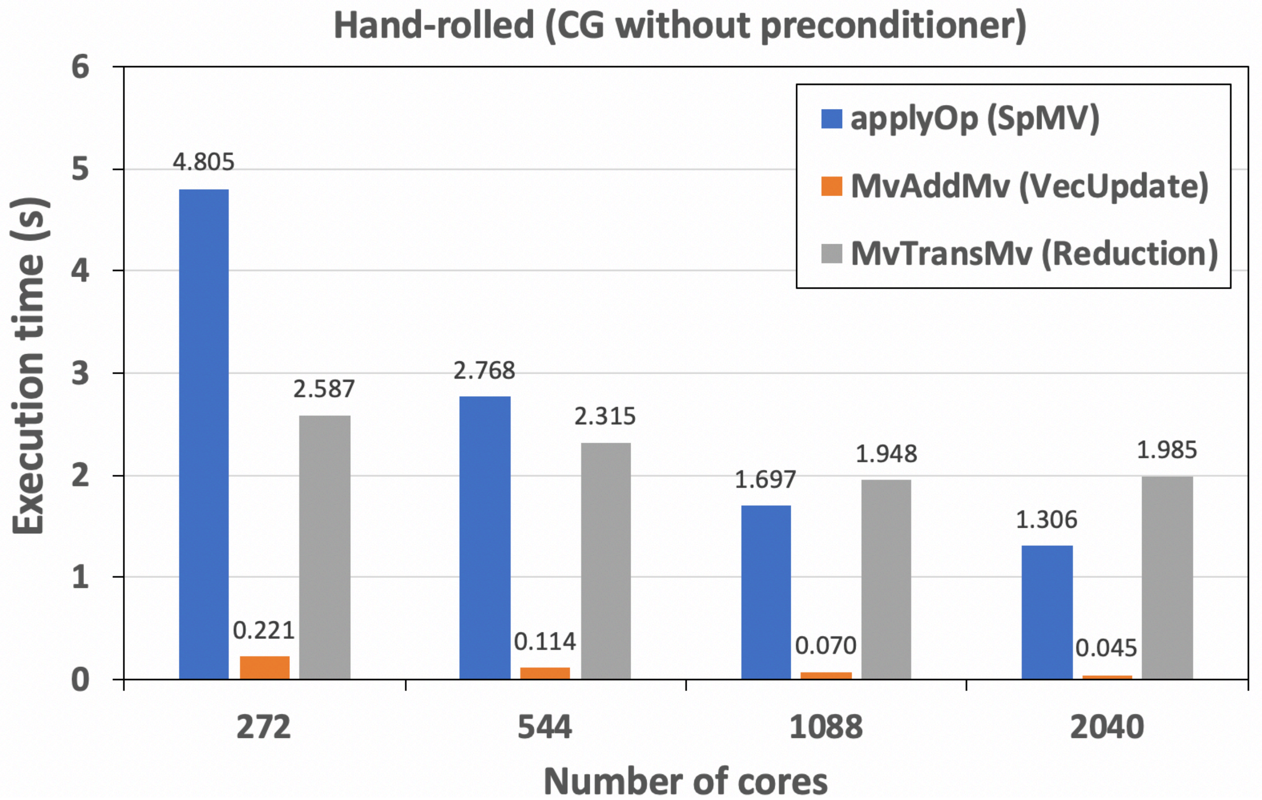




Results

- **Bottlenecks in ForTrilinos solver (timer comparison: ForTrilinos vs Hand-rolled solver)**
  - Test case: EC60to30 (total 235,160 cells) for one-day integration
  - MPI_BARRIER is used to synchronize MPI processors before entering the barotropic mode solver.
  - Global reduction is a main bottleneck for both solvers.
  - For the ForTrilinos solver, global reduction and vector updating processes are slower than we expected.





Summary and discussion

- **A new semi-implicit barotropic mode solver has been implemented in MPAS-O using ForTrilinos.**
  - As a competitor to an existing explicit-subcycling scheme, the semi-implicit barotropic mode solver has been implemented in MPAS-O.
  - ForTrilinos which is a Fortran interface to Trilinos C++ math library is introduced for an implicit solver.
  - After careful investigations, we choose the s-step CG linear solver and the algebraic multigrid preconditioner as a linear solver and a preconditioner, respectively.
  - The ForTrilinos semi-implicit solver scales well and faster than the existing scheme until 200 cells per core, while it is slower than the hand-rolled semi-implicit solver.
  - Thanks to the hand-rolled solver, some bottlenecks in the ForTrilinos solver are identified, which will be investigated more thoroughly.
    - Function call overheads in Trilinos?
    - Additional global communications in somewhere?