JEDI performance comparison (FV3 vs. MPAS)

3D-Var, 2018-04-15_00Z, 6H window, UKMO ODB file: 216K aircraft, 30K radiosonde

FV3-GEOS

- 48K grid columns (C90)
- 72 vertical levels
- NASA Discover and UCAR Cheyenne

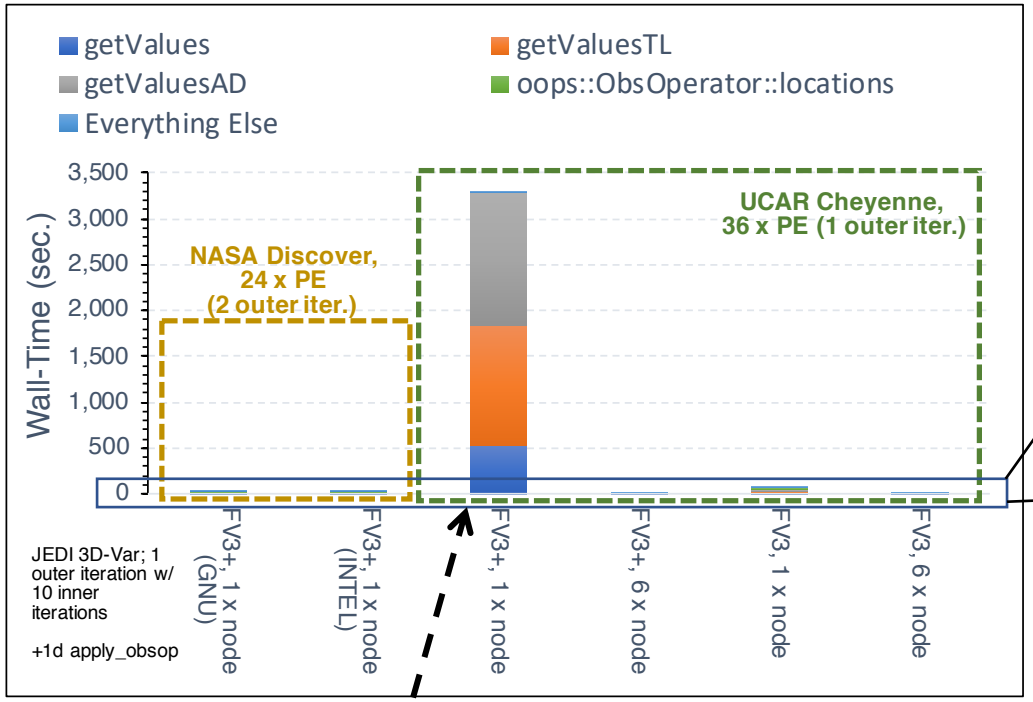
MPAS

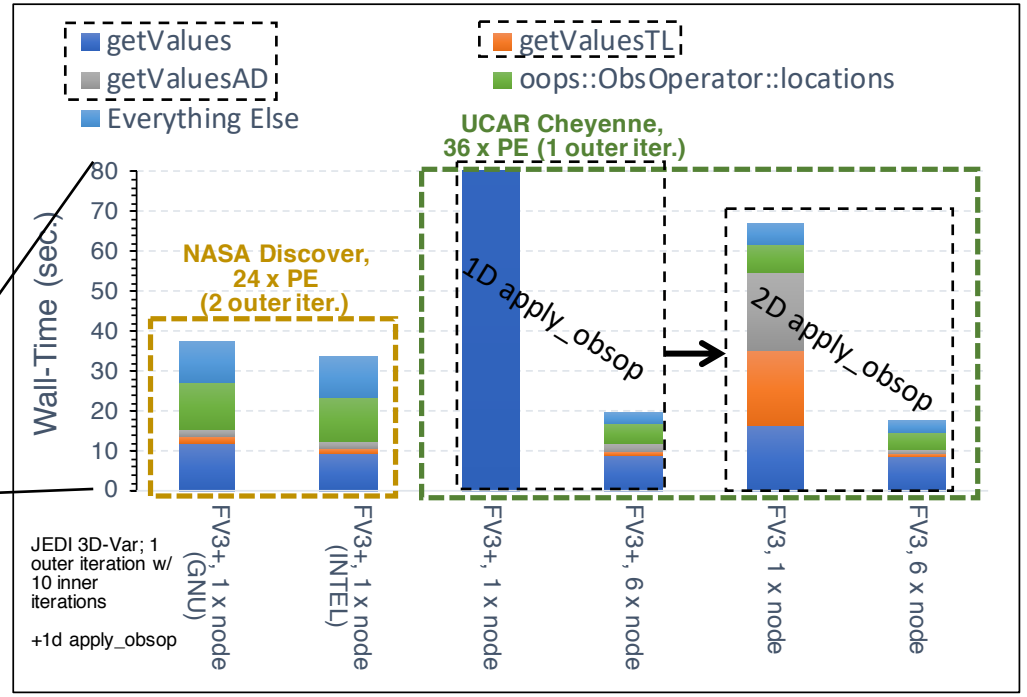
- 41K grid columns (120 km)
- 55 vertical levels
- UCAR Cheyenne only

Cheyenne: GNU 7.3.0; OPENMPI 3.0.1
Discover: GNU 7.3.0; OPENMPI 3.0.0
All builds use the same compile flags:

- Fortran: -O3 -funroll-all-loops -finline-functions
- C++: -O3

FV3





FV3::develop is very slow on single cheyenne node owing to getValues/TL/AD

We expect getValues to be the major cost in 3D-Var:

- Initialize horiz. interp. weights
- Communicate round-robin distributed model values (MPI)
- Apply horiz. interp. weights

Wall-times strongly scale between 1 and 6 nodes, even though #PE's constant and not memory limited

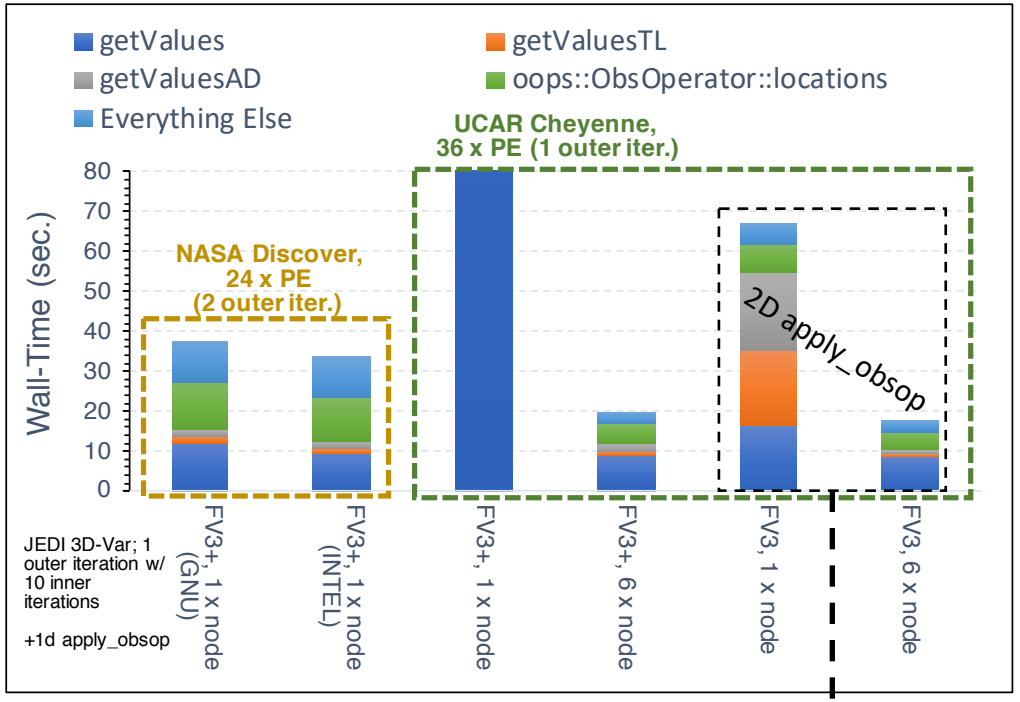
Result not shown: wall-times on Discover do not change on 6 nodes

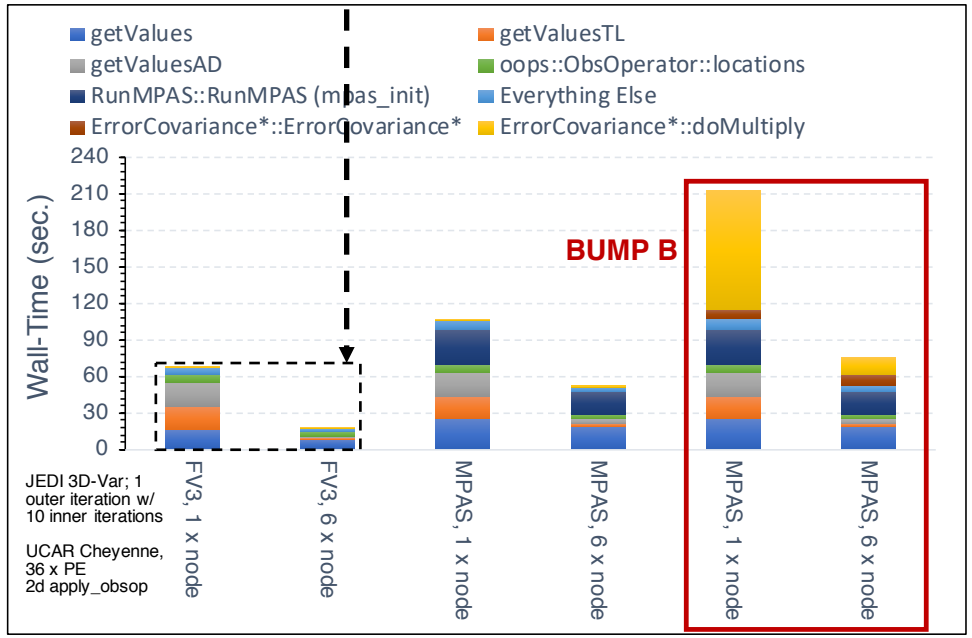
2D apply_obsop refactors getValues/TL/AD model interfaces to BUMP:

  - 1D: Loop over single-level 1d arrays, communicate + apply weights
  - 2D: simultaneously communicate 2d array with all model levels
- LARGE impact on 1 cheyenne node, SMALL impact on six nodes
- 1 node still 4x slower per outer iter. on Cheyenne than on Discover

- 2D apply_obsop used in both FV3 and MPAS
- getValues/TL/AD wall-times scale with number of nodes (BUMP+MPI)
- MPAS is slower than FV3 due to mpas_init and getValues
  - initialization of interpolation?
  - Currently reading full MPAS restart file
- BUMP covariance multiply also scales with # nodes; similar MPI comms

UCAR Cheyenne,
36 x PE
FV3 & MPAS





Remaining questions/clues

- Warning message on cheyenne
  - mca_base_component_repository_open: unable to open mca_mtl_psm: libpsm_infinipath.so.1: cannot open shared object file: No such file or directory (ignored)
  - UCAR CISL helpdesk says this should not cause a problem, but is it?

- Why does apply_obsop wall-time scale with number of nodes on cheyenne? Same for ErrorCovarianceBUMP::doMultiply? Communications bottleneck? Why not on Discover?

- Why does 2D communication speed up getValues/TL/AD on cheyenne? Is this a clue to the underlying problem on cheyenne? Infiniband bandwidth?

- Performance is consistent on Discover across compiler/MPI libraries. Do other libraries behave differently on Cheyenne? Intel vs. GNU? MPT vs. OpenMPI vs. iMPI?