IODA Status and Test Results

Stephen Herbener and Steven Vahl

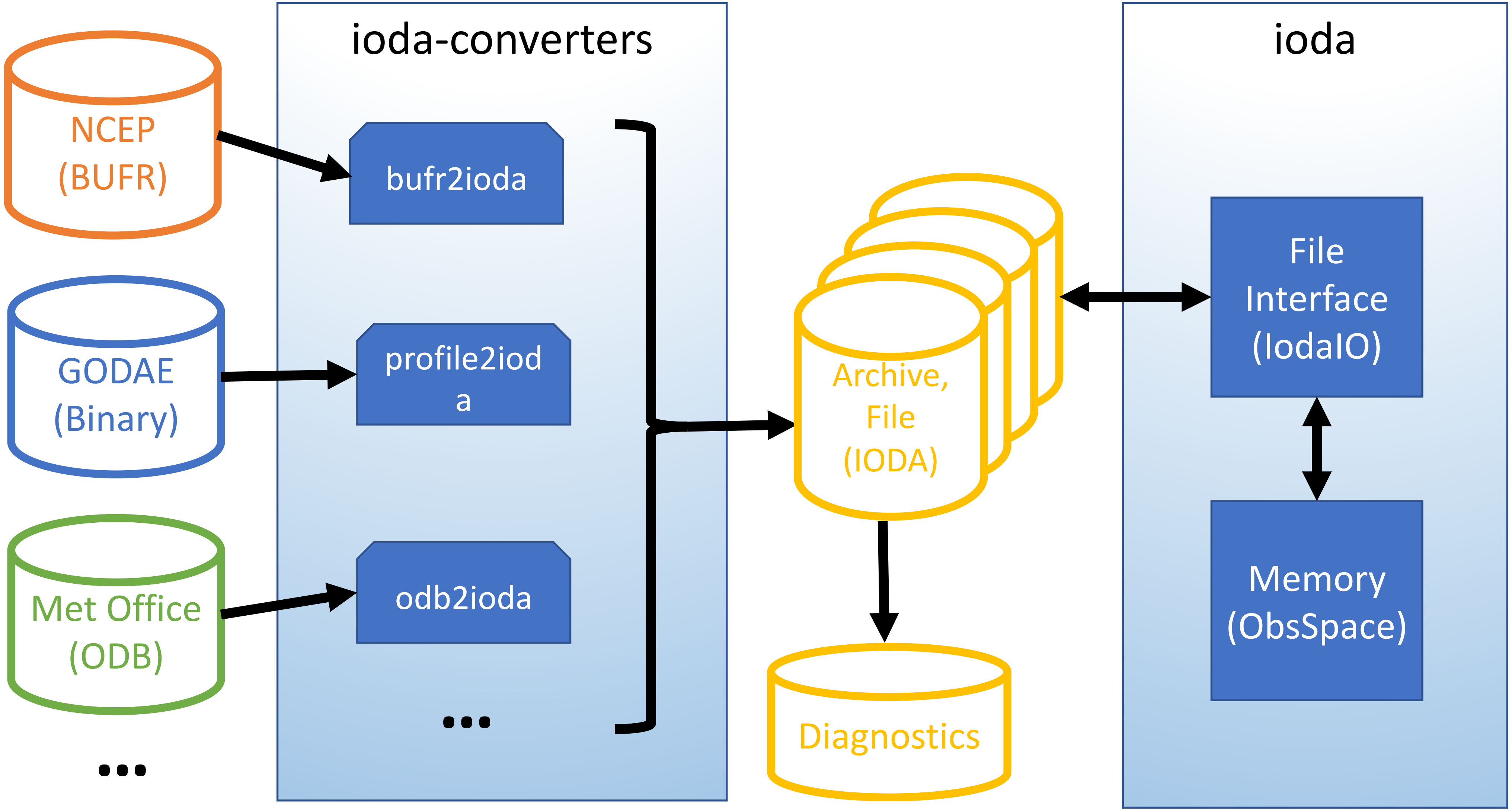
01/23/2020

Thanks to the JEDI partners and core team for contributions to the development of IODA

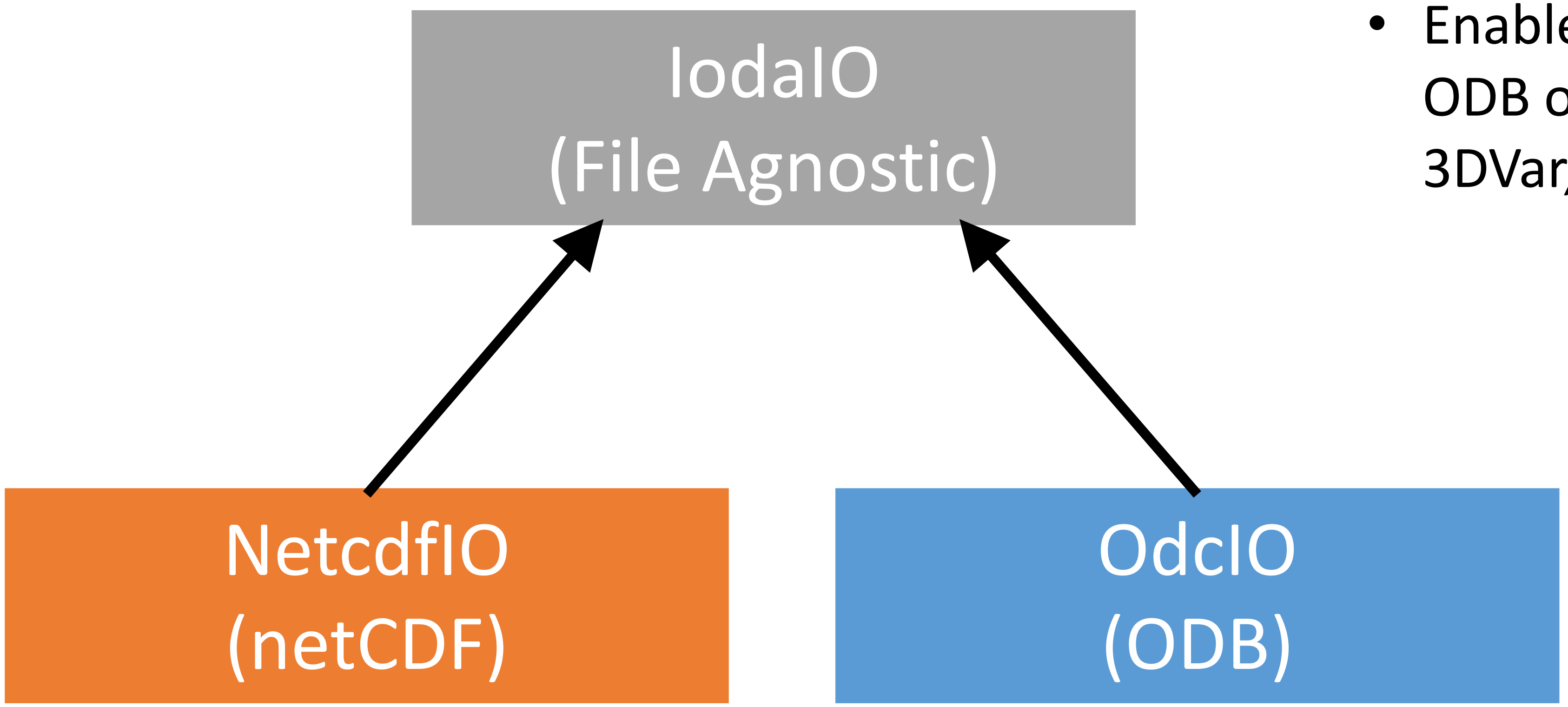
IODA Requirements

ID	Name	Description
IodaReq1	Flexible	Handle many obs types, accommodate new obs types, usable by research and operations
IodaReq2	Access to data and meta data	Efficient query/filter based on data and/or meta data values
IodaReq3	Efficient I/O	Sufficient speed and volume
IodaReq4	Efficient compression	Economical with archive storage space
IodaReq5	Portable	Run on many hardware platforms/compilers, minimize reliance on 3rd part libraries, support for multiple programming languages
IodaReq6	Security	Navy classified, EMC private
IodaReq7	Support analysis	Enable use of diagnostic tools to analyze/visualize the performance of the system
IodaReq8	Data import	Handle various raw obs file types (BUFR, ODB, netcdf, etc.), handle various data schema (NCEP prepBUFR, Met Office ODB, etc.)
IodaReq9	Ease of use	Intuitive, familiar, consistent interfaces for both developers and users
IodaReq10	Reliability	For operations it cannot break down
IodaReq11	Replicate existing functionality	Enable comparison with other DA systems (GSI, e.g.)

Current state of IODA

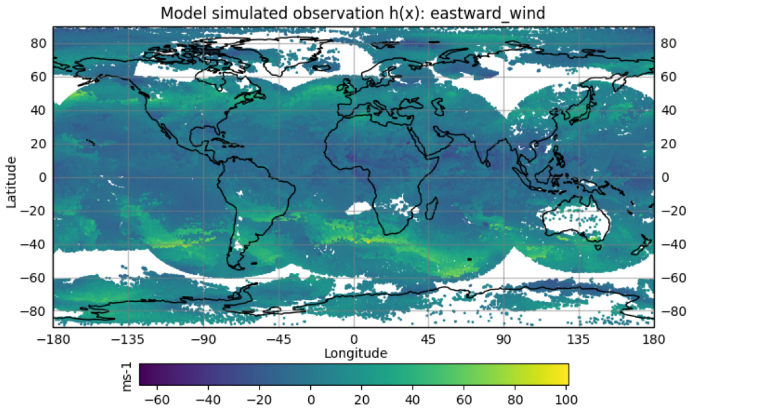


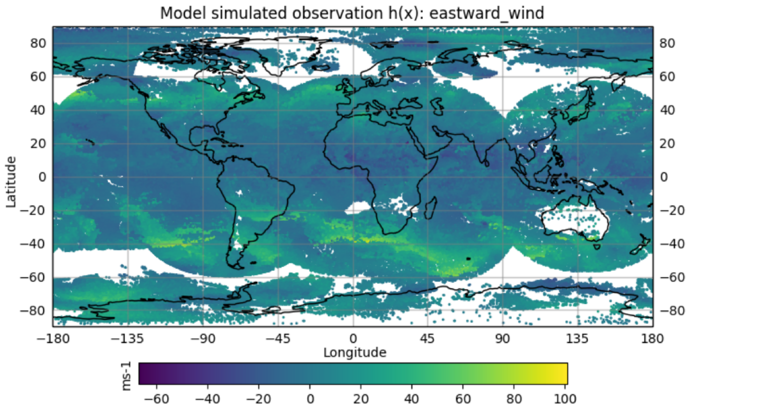
IodaIO Class Structure



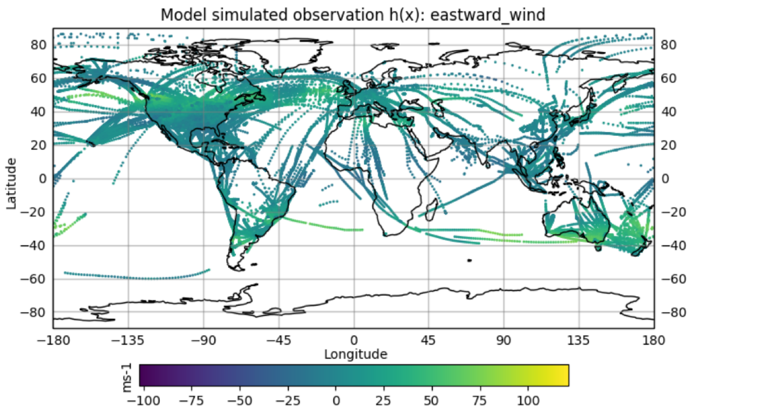
- Read, Write enabled for both netCDF and ODB
- Enables using either netCDF or ODB obs files in DA jobs (HofX, 3DVar, etc.)

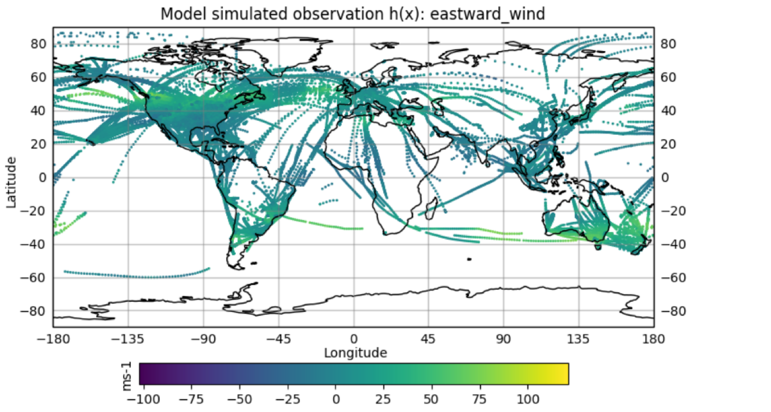
H(x) from a Sample 3DVar Run





Satwind, u-component of wind





Aircraft, u-component of wind

```
Nonlinear Jo(Satwind) = 104373, nobs = 700843, Jo/n = 0.148925, err = 2.46613e+06
Nonlinear Jo(Scatwind) = 33398.5, nobs = 430259, Jo/n = 0.0776243, err = 3.5
Nonlinear Jo(Vadwind) = 29368.5, nobs = 32858, Jo/n = 0.893801, err = 1.74811
```

netCDF obs files

```
Nonlinear Jo(Satwind) = 104373, nobs = 700843, Jo/n = 0.148925, err = 2.46613e+06
Nonlinear Jo(Scatwind) = 33398.5, nobs = 430259, Jo/n = 0.0776243, err = 3.5
Nonlinear Jo(Vadwind) = 29368.5, nobs = 32858, Jo/n = 0.893801, err = 1.74811
```

ODB obs files

- FV3-GFS, 3DVar, C192, 1865188 observations assimilated
- S4, Intel compilers, Intel MPI (impi), 864 MPI Tasks

IODA Status

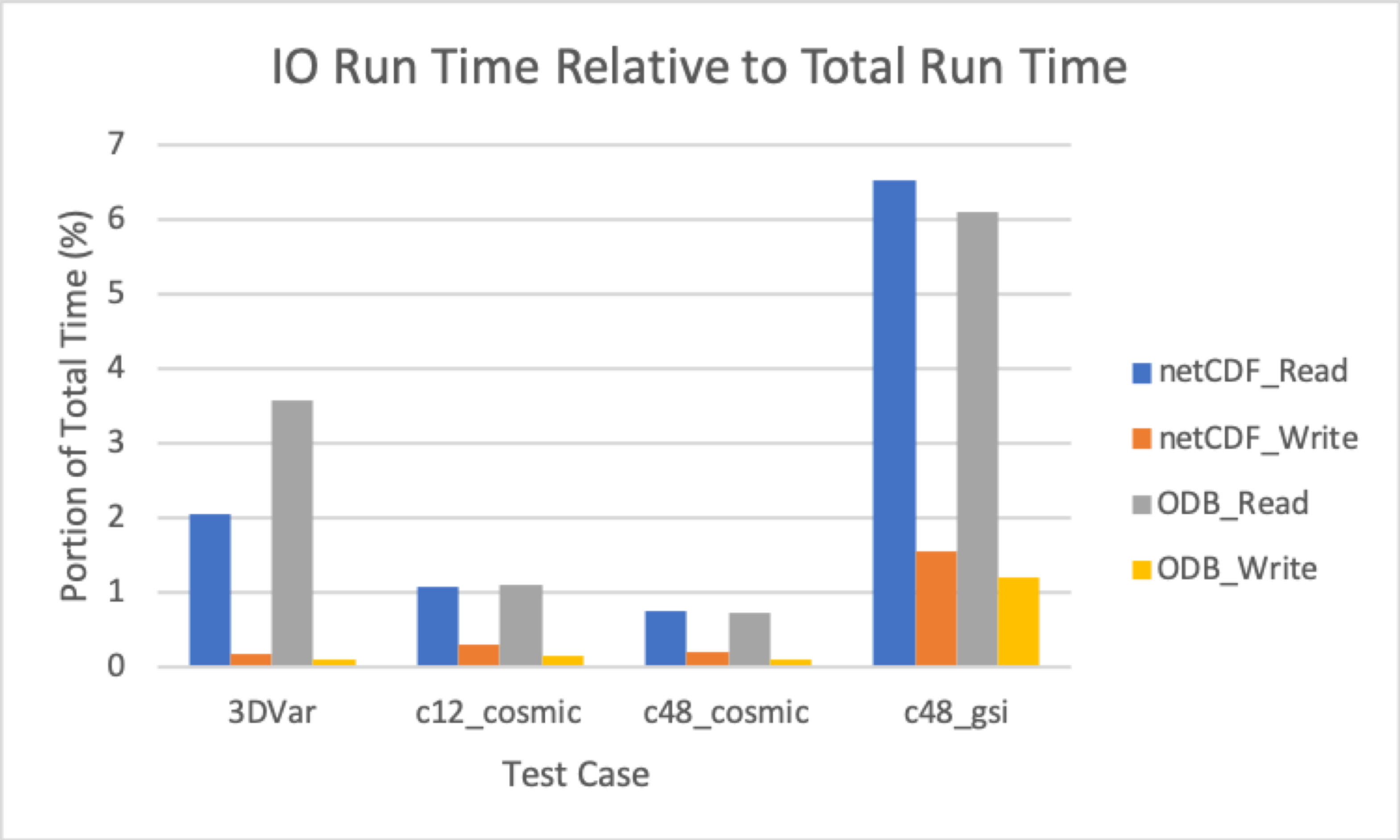
Gray: Not heavily tested/measured yet
Red: Focus for today's discussion

Requirement	IODA Status
Flexible	Running DA tasks with ~30 obs types (10 conv, 20 radiance)
Access to data and meta data	Providing support for ~20 filter operations (constructed from generic filter ops in UFO), plus support for obs operators for the ~30 obs types
Efficient I/O	Testing larger DA runs using millions of observations
Efficient compression	Testing the construction of obs data archive
Portable	GNU, Intel and Clang compilers; OpenMPI, MPICH2, IMPI MPI implementations Running on supercomputers (Hera, Discover, Cheyenne, S4), AWS EC2 instances, and on laptops (mac, linux)
Security	Installation/execution can be done behind firewalls
Support analysis	Have capability to write netcdf and ODB formats giving access to a variety of diagnostic tools
Data import	Currently handle Netcdf, ODB, BUFR, Marine binary profile file types, from various sources including GSI, NCEP, Met Office, GODAS and GODAE
Ease of use	Simple interface (get, put) for direct access by obs operators, and access via obs vectors
Reliability	Testing larger DA runs using millions of observations
Replicate existing functionality	GSI conversion path includes extraction of H(x) and filtering results from GSI runs for comparison with JEDI results, all obs assimilated by GSI have been converted for use in JEDI

IODA tests

- Comparing ODB and netCDF file formats
- Created a matching set of ODB obs files from the existing set of netCDF obs files
- Collected stats from various test cases
  - H(x) runs
  - FV3-GFS 3Dvar runs
  - ObsSpace constructor/destructor runs (Ioda IO Read/Write)
- ObsSpace tests
  - Started with IASI obs data (616 obs per location)
  - Read/Write 1000 locations (616,000 obs), 2000 locations, 4000 locations, etc. up to 256000 locations (157 million obs)
  - Repeated each case 10 times and averaged the constructor (read) and destructor (write) times reported by OOPS

IodaIO Execution Time

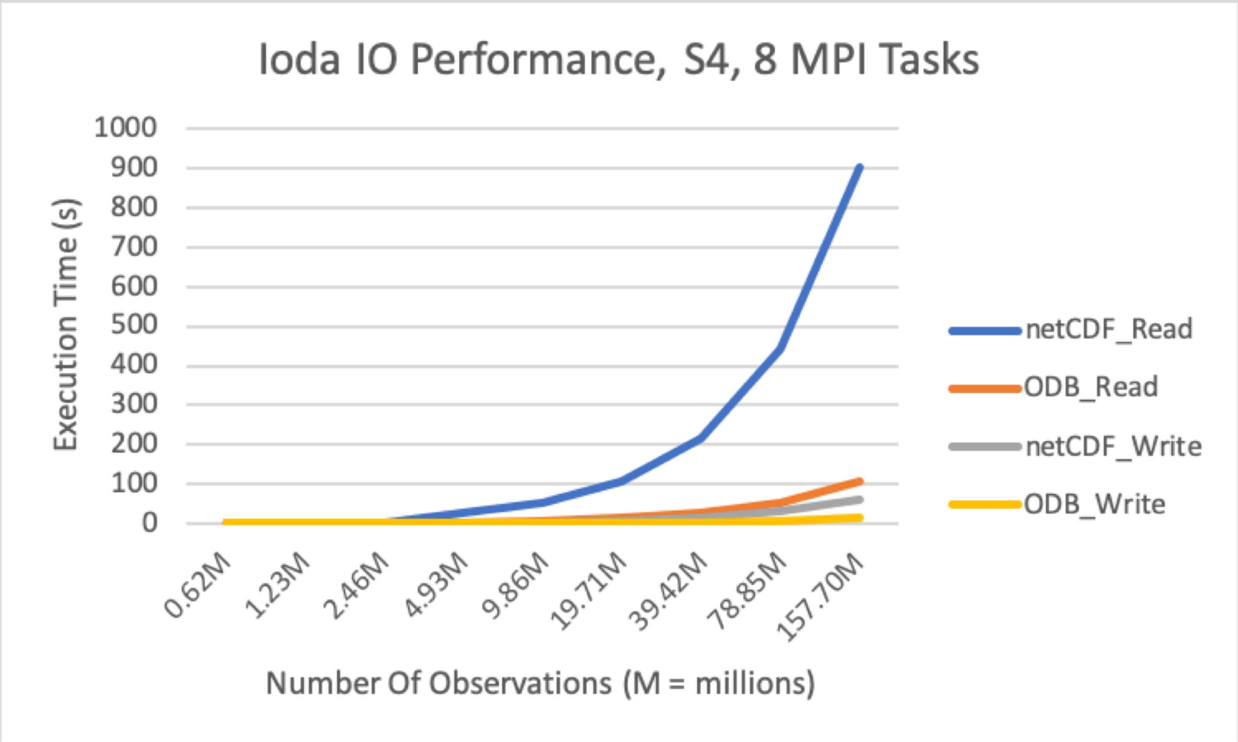


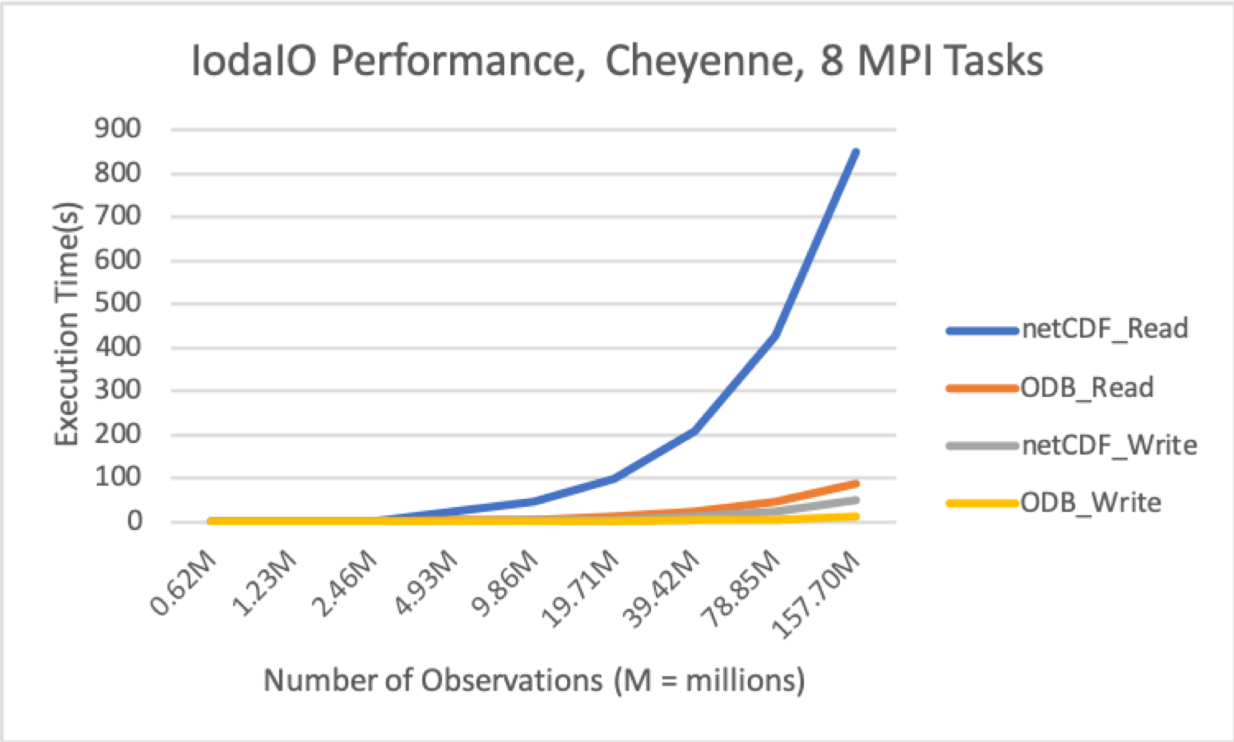
Test Cases

- 3DVar
  - FV3-GFS, C192, 1865188 obs
  - Averages from 2 runs using netCDF and 2 runs using ODB.
- c12_cosmic
  - H(x)
  - Desktop, c12, cosmic observation data,
  - Averages from 4 runs using netCDF and 4 runs using ODB
- c48_cosmic
  - Same as c12_cosmic, except c48 resolution
- c48_gsi
  - Same as c48_cosmic, except obs collected from GSI ncdiag output

- Read and write percentages are comparable between netCDF and ODB
- Opportunities exist to improve these numbers
  - Eg, Each MPI task only reads the portion of the file that it uses

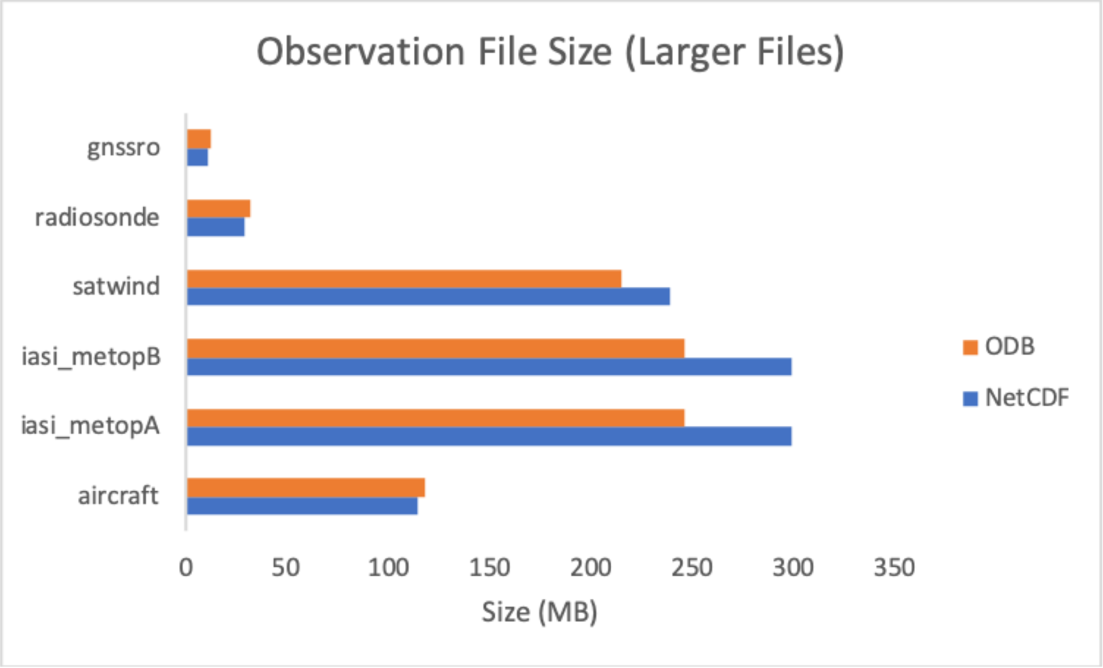
Ioda IO Timing

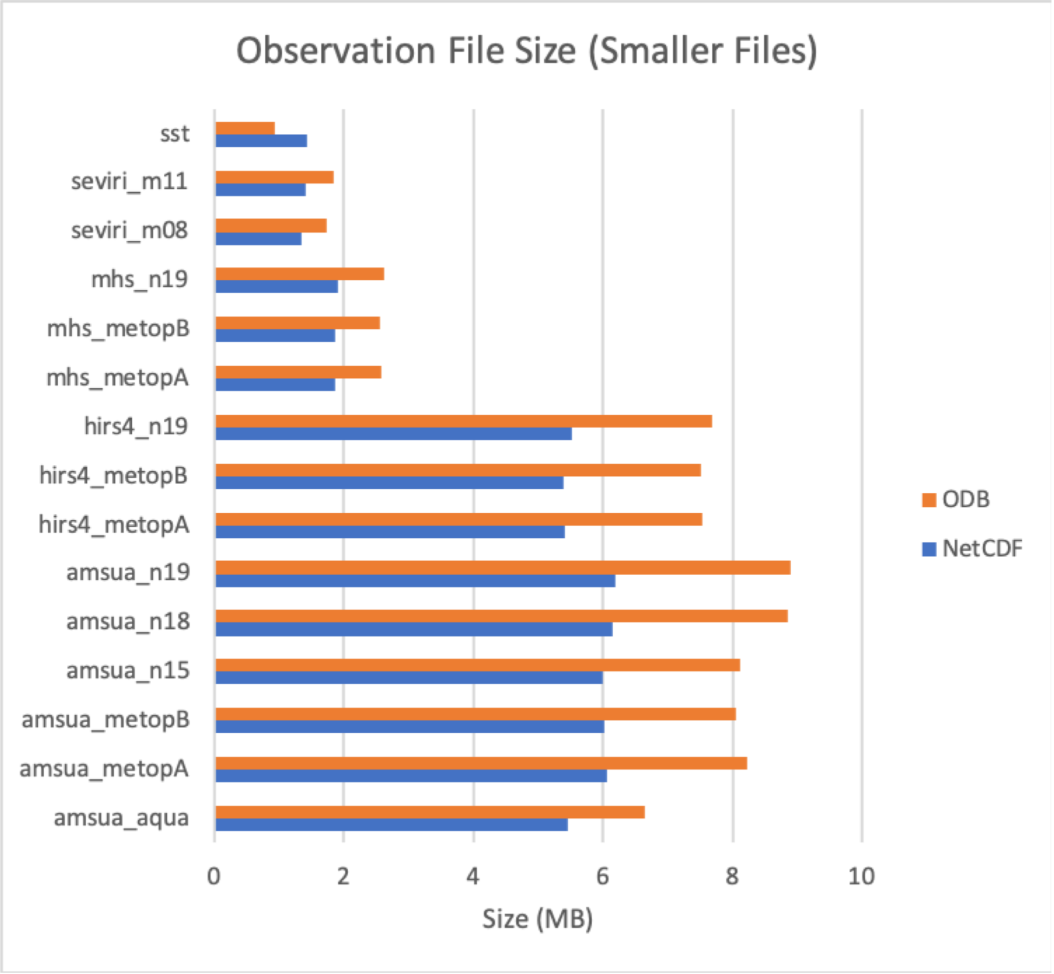




- IASI observation data
- ODB tends to be faster for both read and write (but netCDF_Write times are comparable)
- netCDF_Read can likely be tuned to go faster

IODA File Size





- All file pairs contain same number of observations
- For the larger files, ODB tends to be smaller than netCDF
- For the smaller files, netCDF tends to be smaller than ODB

Summary

- Work needs to be done (and opportunities exist) to speed up Ioda IO
- At this point netCDF and ODB seem comparable in speed and size
  - ODB has a slight edge in the particular test cases that we have

Backup

Observation Types

Conventional Observation Types	Radiance Observation Types	
Aircraft	AMSU-A	SEVERI
Radiosonde	ABI	SNDR
Radar	AHI	Satwind
Sfc	AIRS	Scatwind
SfcShip	ATMS	SMAP
VAD wind	CRIS	Marine SST
Wind profiler	GNSSRO	
Marine insitu temp	HIRS-4	
Marine profilier	IASI	
Marine SST	MHS	
	SBUV-2	
	VIIRS AOD	
	Cryosat	
	ICEC	

Generic filter building blocks

Generic filter
Bounds check
Background check
Blacklisting
Thinning
Gaussian thinning
Domain check
Difference check

IODA Converters

Obs category	Language	Input file type	Obs types
ODB	Python	Met office ODB2	Aircraft, Radiosonde, AMSU-A
GNSSRO	Fortran	BUFR	GNSSRO
GSI	Python	Ncdiag	Conventional, Radiance (all types handled by GSI)
Marine	Python	Netcdf, BUFR, Binary	ArgoClim2, SST, Profile, Ship, Trak, ADT, In-situ, SSS
NCEP	Python, Fortran	BUFR, prepBUFR	Aircraft, Radiosonde, AMSU-A