# Marine Applications

JEDI Academy - June 2019

**Travis Sluka**
Joint Center for Satellite Data Assimilation (JCSDA)

# What is SOCA?

## **S**ea-ice, **O**cean, and **C**oupled **A**ssimilation (SOCA)

Main objectives can be summarized as:

1. **Prototype for a common, flexible, ocean/ice DA**

   For use by NOAA/EMC and NASA/GMAO in coupled models and seasonal forecasting
2. **Merge ocean / atmosphere / ice DA methods**

   coupled UFOs for surface sensitive randiances

   strongly/weakly coupled DA
3. **a real-time demonstration of what JEDI is capable of**

# SOCA team

## JCSDA contributors:

- Guillaume Vernieres, Travis Sluka, Hamideh Ebrahimi
- CRTM and JEDI team

## In-kind contributors:

- Rahul Mahajan, Santha Akella, Deanna Spindler, Denise Worthen, Jong Gyun Kim, Stylianos Flampouris, Shastri Paturi, and others

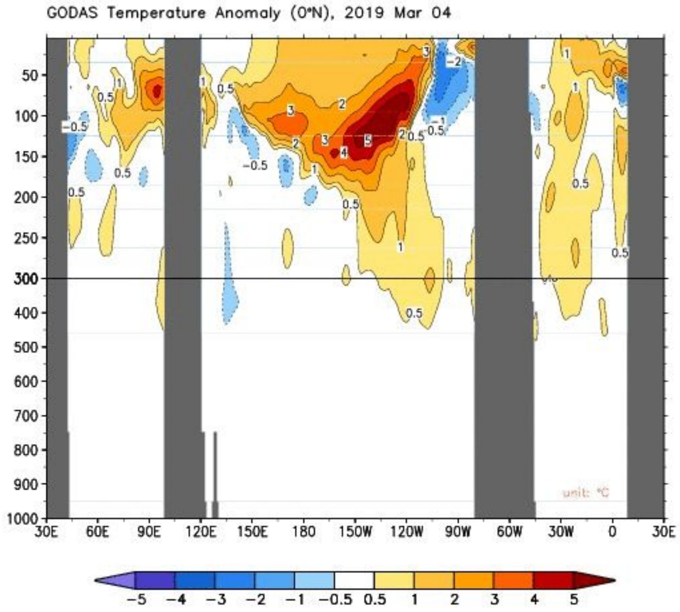
# Marine DA upgrade for NOAA/NCEP

From the point of view of a former NOAA/NCEP employee ...

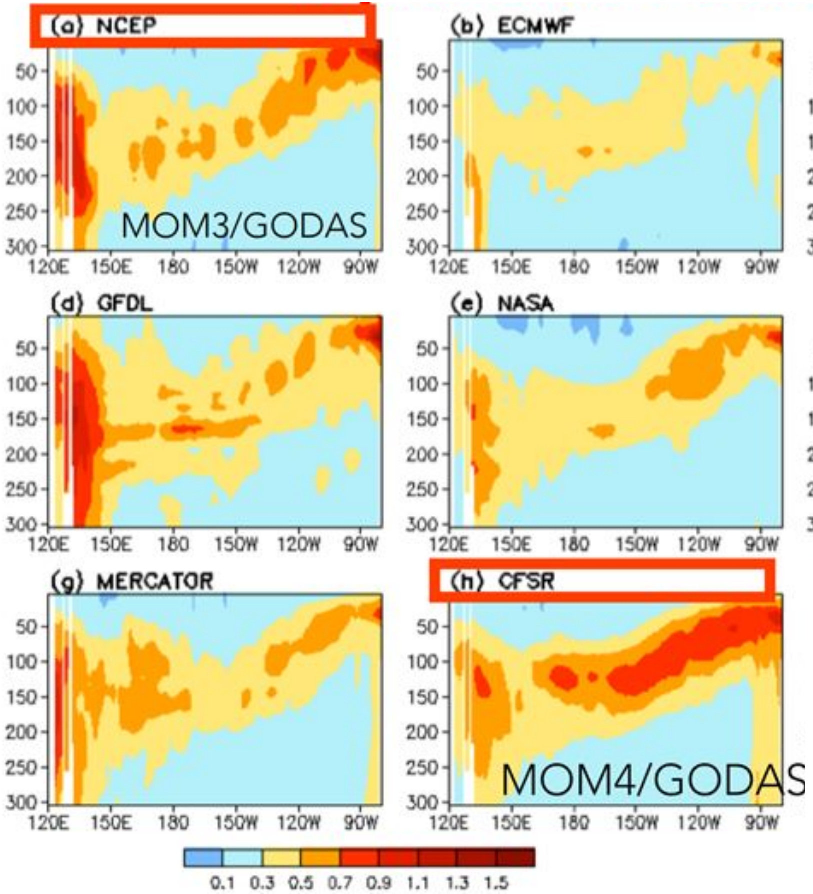
The **G**lobal **O**cean **D**ata **A**ssimilation **S**ystem (GODAS) currently operational at NOAA/EMC is **old**.

- Last significant update was **~2003**
- Limited observations (**insitu T only**)
- Simple **univariate 3DVAR**
- **Difficult to maintain**

NCEP operational GODAS



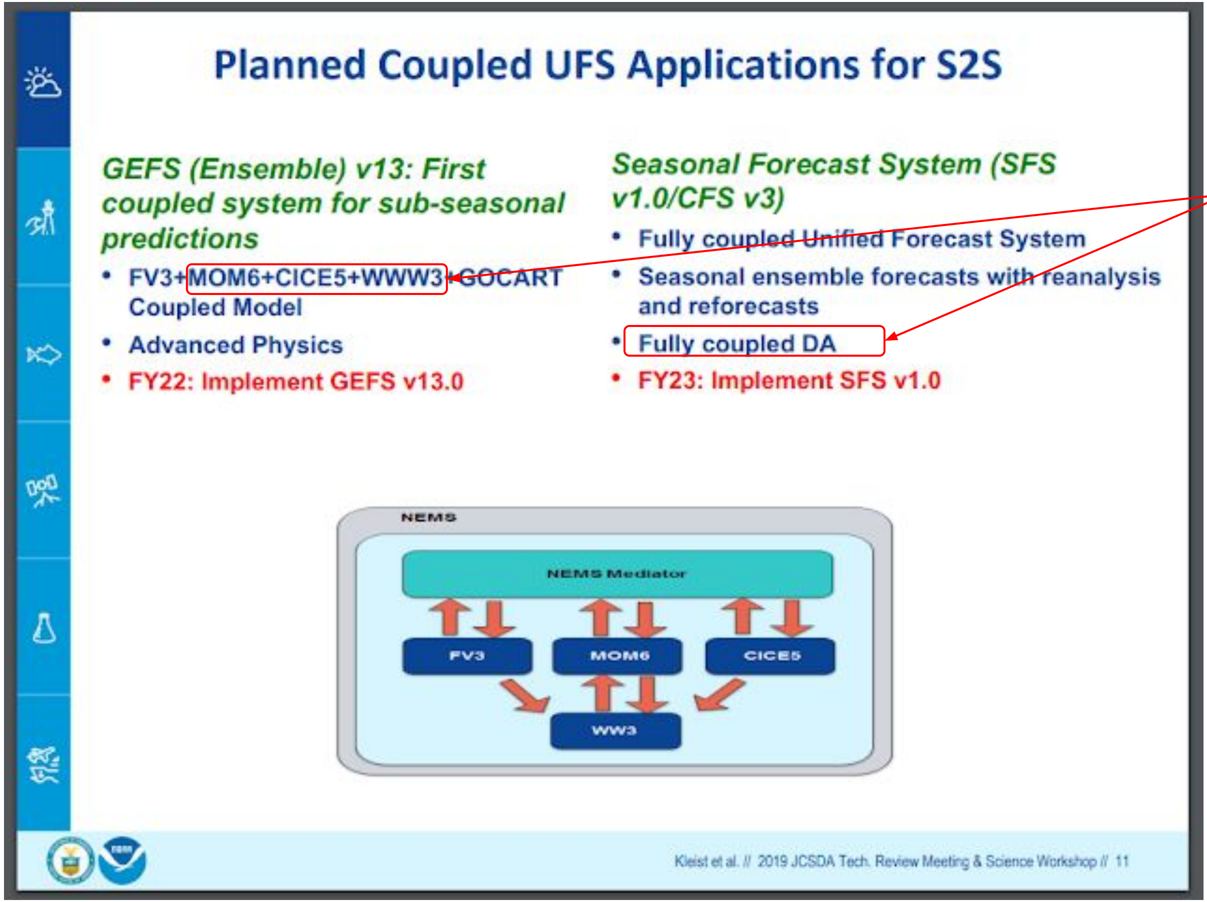
# Marine DA upgrade for NOAA/NCEP




RMSD of anomaly correlation vs ensemble mean of various operational products, along Pacific EQ

**NOAA's GODAS often performs poorly**

# Marine DA upgrade for NOAA/NCEP

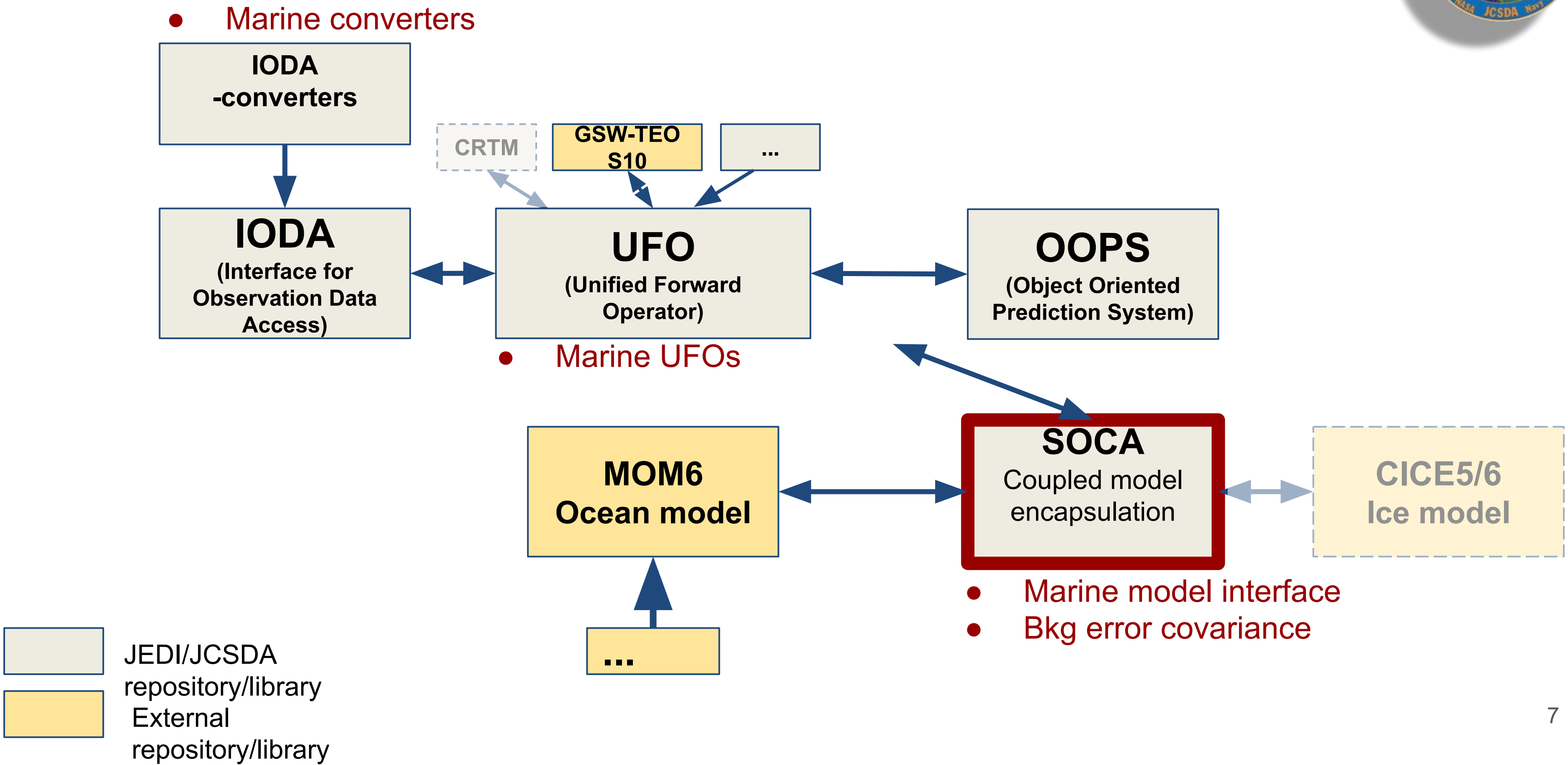



Plans at NOAA/NCEP relying on marine JEDI development

Initial marine JEDI prototype expected to be delivered end of this year.

Implementation of a ¼ degree ocean/ice DA system

# SOCA

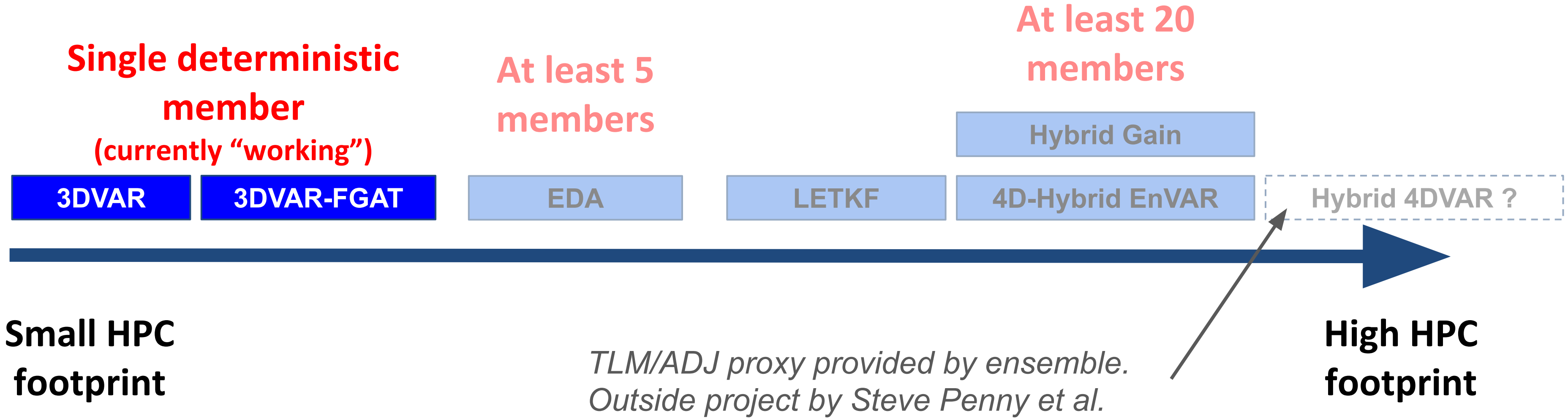


Marine converters
IODA
-converters
CRTM
GSW-TEO
S10
...
IODA
(Interface for Observation Data Access)
UFO
(Unified Forward Operator)
OOPS
(Object Oriented Prediction System)
Marine UFOs
MOM6
Ocean model
SOCA
Coupled model encapsulation
CICE5/6
Ice model
Marine model interface
Bkg error covariance
...
JEDI/JCSDA repository/library
External repository/library

# Marine DA - planned methods

Implementing a number of DA methods, giving end-user many choices

- available in observation and state space solver
- No 4DVAR (unless someone wants to write me a TLM/ADJ for MOM6)!



# SOCA - changing DA methods

To go from 3DVAR to 3DVAR-FGAT:

```
model:
 name: SOCA
 tstep: PT1H
 advance_mom6: 0
 variables: [cicen, hicen, socn, tocn, ssh, hocn]
```

→

```
model:
 name: SOCA
 tstep: PT1H
 advance_mom6: 1
 variables: [cicen, hicen, socn, tocn, ssh, hocn]
```

To go from state space to observation space solver:

```
minimizer:
 algorithm: DRPCG
```

→

```
minimizer:
 algorithm: RPCG
```

# SOCA - Background Error

B-matrix for the ocean is modelled with a combination of

- **BUMP** (horizontal correlations)
- **Variable transforms** (balance operators, multivariate aspect)
- **Other Parameterizations** (vertical correlation, error variance)

$$\boldsymbol{K}\boldsymbol{D}\boldsymbol{C_v}^{\frac{1}{2}}\boldsymbol{C_h}^{\frac{1}{2}}\boldsymbol{C_h}^{\frac{T}{2}}\boldsymbol{C_v}^{\frac{T}{2}}\boldsymbol{D}\boldsymbol{K}^T$$

Currently tightly part of the SOCA repository, but plan to generalize more to allow greater mixing and matching of different marine B matrix methods

# SOCA - Background Error

$$\boldsymbol{K} D C_v^{\frac{1}{2}} C_h^{\frac{1}{2}} C_h^{\frac{T}{2}} C_v^{\frac{T}{2}} D \boldsymbol{K}^T$$

## Variable transformations

They look more complicated than they really are…

**temperature**, **salinity**, **sea surface height**, (and eventually velocity) are transformed into control variables that are uncorrelated

(balanced and unbalanced parts of S, SSH, U, V)

$$\boldsymbol{K} = \begin{bmatrix} I & 0 & 0 & 0 \\ K_{ST} & I & 0 & 0 \\ K_{\eta T} & K_{\eta S} & I & 0 \\ K_{cT} & 0 & 0 & I \end{bmatrix}$$

**Weaver et al, 2006**

$$\delta S_B = \frac{\partial S}{\partial T}\delta T$$

*Trocoli and Haines, 1999*

$$\delta \eta_B = -\int_{Bottom}^{0} \frac{\delta \rho(T,S,z)}{\rho_0} dz$$

*Cooper and Haines, 1999*

$$\delta c_B = \frac{\partial c}{\partial T}\delta T$$

# SOCA - Background Error

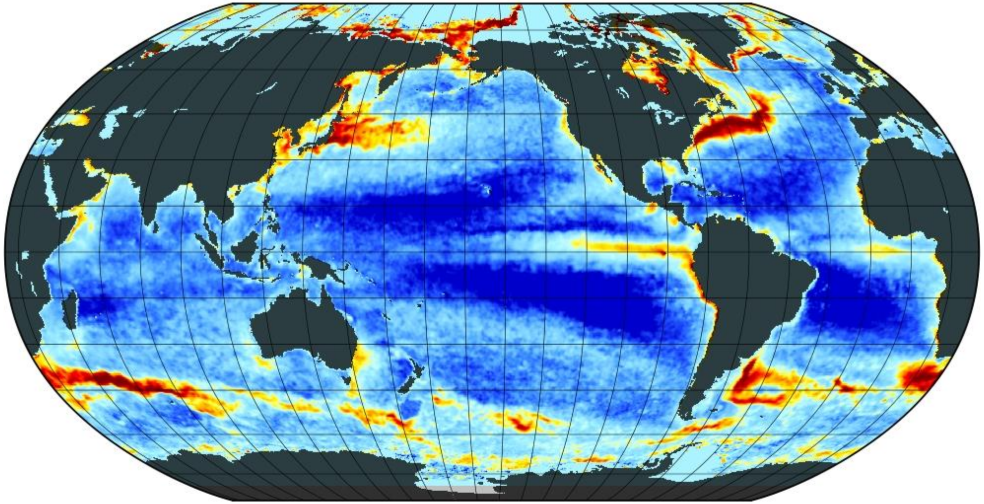
$$K\mathbf{D}C_v^{\frac{1}{2}}C_h^{\frac{1}{2}}C_h^{\frac{T}{2}}C_v^{\frac{T}{2}}\mathbf{D}K^T$$

## Background error variance

**Temperature** - function of vertical temperature gradient, modulated by a precomputed horizontally varying surface field

**Salinity** - none, below the mixed layer

**SSH** - none along EQ, 0.1m in extra tropics



Imposed minimum temperature background error at surface

Due to the previous variable transforms, temperature is the key variable here

# SOCA - Background Error

$$K D \boldsymbol{C_v}^{\frac{1}{2}} C_h^{\frac{1}{2}} C_h^{\frac{T}{2}} \boldsymbol{C_v}^{\frac{T}{2}} D K^T$$

**Vertical convolution**

Should be handled by BUMP…

But for now we parameterize based on the mixed layer depth given by the model, and model level thicknesses

# SOCA - Background Error

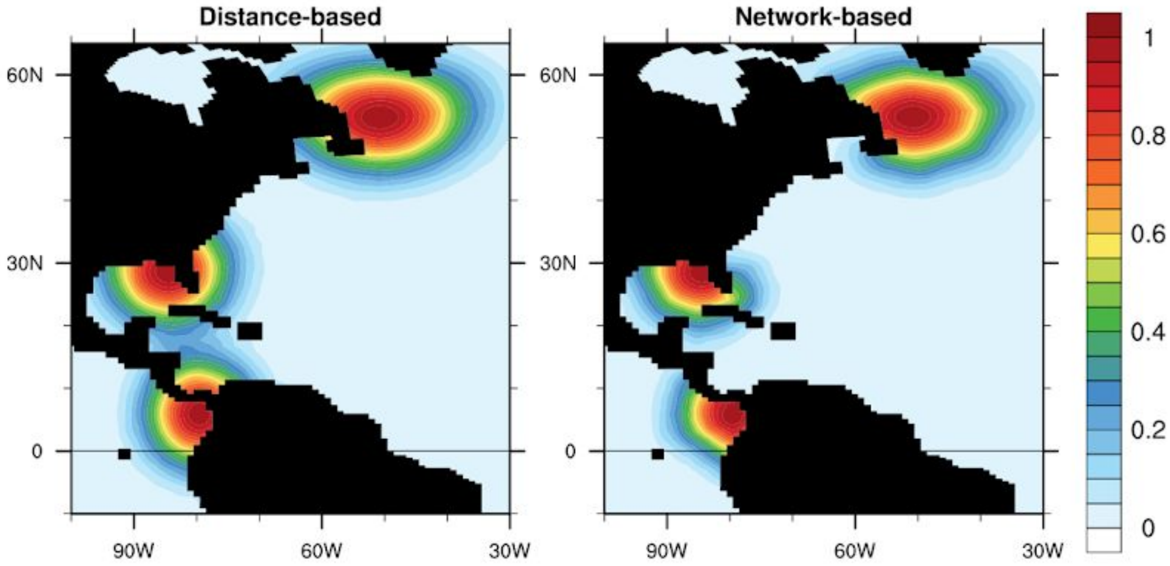
$$K D C_v^{\frac{1}{2}} \boldsymbol{C_h}^{\frac{1}{2}} \boldsymbol{C_h}^{\frac{T}{2}} C_v^{\frac{T}{2}} D K^T$$

## Horizontal convolution

Tricky given that pesky land in the way!

Often times done with **diffusion operators.** But this can be slow.

BUMP can handle **land masks** and provides a good proxy for what a diffusion operator would do



# SOCA - Change of Variables

soca/src/Transforms/instantiateBalanceOpFactory.h

```
namespace soca {

void instantiateBalanceOpFactory() {
 static oops::LinearVariableChangeMaker<soca::Traits,
              oops::LinearVariableChange<soca::Traits, soca::VertConv> >
              makerBalanceOpVertConvSOCA_("VertConvSOCA");

 static oops::LinearVariableChangeMaker<soca::Traits,
              oops::LinearVariableChange<soca::Traits, soca::BkgErr> >
              makerBalanceOpBkgErrSOCA_("BkgErrSOCA");

 static oops::LinearVariableChangeMaker<soca::Traits,
              oops::LinearVariableChange<soca::Traits, soca::BkgErrGodas> >
              makerBalanceOpBkgErrGODAS_("BkgErrGODAS");

 static oops::LinearVariableChangeMaker<soca::Traits,
              oops::LinearVariableChange<soca::Traits, soca::BkgErrFilt> >
              makerBalanceOpBkgErrFILT_("BkgErrFILT");

 static oops::LinearVariableChangeMaker<soca::Traits,
              oops::LinearVariableChange<soca::Traits, soca::Balance> >
              makerBalanceOpBalanceSOCA_("BalanceSOCA");
}}
```

The previous components of the background error covariance (other than BUMP) are contained in separate "**LinearVariableChange**" classes, and added to a common **factory**

# SOCA - Bkg Err Configuration

$C_h$

```yaml
Covariance:
     covariance: SocaError
     strategy: specific_univariate
     load_nicas: 1
     lsqrt: 1

     variable_changes:
     - varchange: BkgErrGODAS
       t_min: 0.1
       t_max: 2.0
       t_dz:  20.0
       t_efold: 500.0
       s_min: 0.0
       s_max: 0.25
       ssh_min: 0.0   # value at EQ
       ssh_max: 0.1   # value in Extratropics
       ssh_phi_ex: 20 # lat of transition from extratropics

     - varchange: VertConvSOCA
       Lz_min: 2.0
       Lz_mld: 1
       Lz_mld_max: 500.0
       scale_layer_thick: 1.5

     - varchange: BalanceSOCA
       dsdtmax: 0.1
       dsdzmin: 3.0e-6
       dtdzmin: 1.0e-6
       nlayers: 2
```

$D$

$C_v$

$K$

… these are then instantiated if specified on the .yaml configuration file.

Multiple ways of representing the various components of the background error covariance can be implemented.

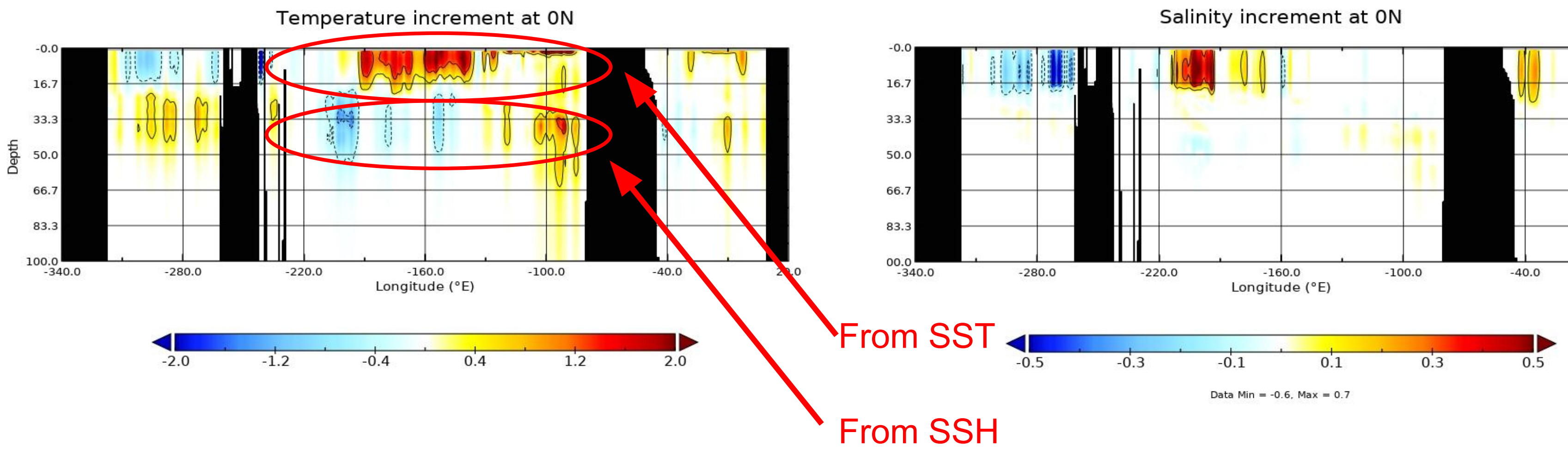
The components can then be mixed and matched as desired at run-time.

...keeping with the OOP mentality of JEDI

# SOCA - surface obs impact

In the ocean, vast majority of observations are of the surface (SST, SSS, SSH)

The balance operators, and MLD based vertical convolution are crucial for impacting the deeper ocean.



# Marine Observations

We are able to ingest a fairly complete set of ocean observations

Note that for the satellite observations, these are all **retrievals**.

Direct radiance assimilation using CRTM is planned for latter.

## NCEP's GODAS

- **Insitu T**
  (and that's it)

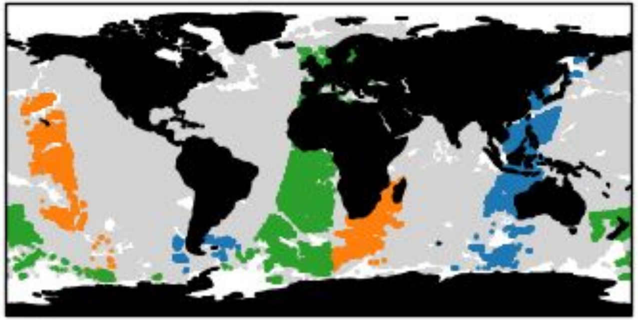
## SOCA

- **SST retrievals**
  - **VIIRS** (Suomi NPP, NOAA-20)
  - **ABI** (GOES-16)
  - **AHI** (Himawari 8)
  - **AVHRR** (MetopA, MetopB, MetopC, NOAA-19)
  - **MODIS** (Aqua, Terra)
- **Altimetry - absolute dynamic topography**
  - NESDIS RADS database
    (cryosat, Jason 2/3, Sentinnel, SARAL, …)
- **Sea surface Salinity retreivals**
  - SMAP / SMOS
- **Insitu T/S**
  - FNMOC / GMAO / World Ocean Database
- **Ice fraction**

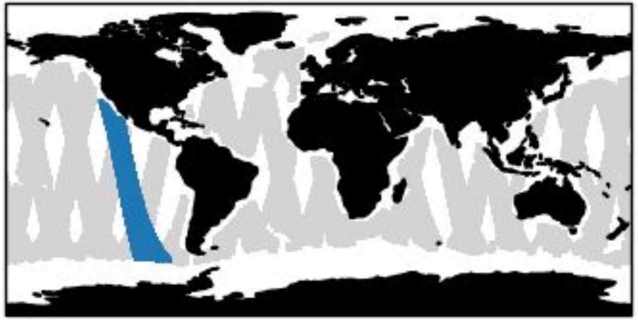
# Marine Observations

1 day of observations
( 2018-04-15 )

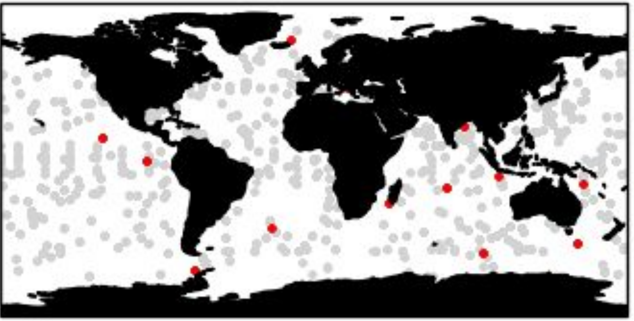
**sea surface temperature (IR)**
AVHRR (metopa, noaa19)
VIIRS (suomi-npp)



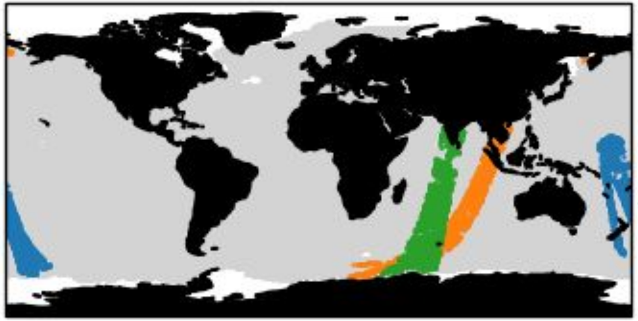
**sea surface salinity**
SMAP



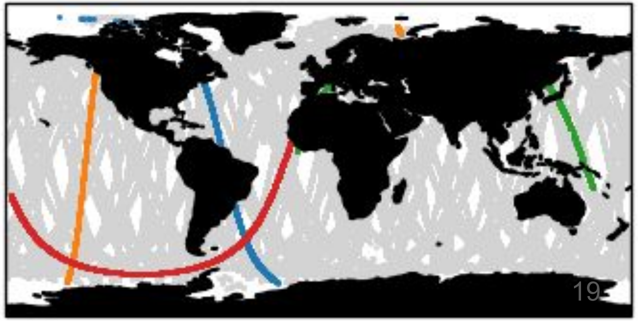
**Insitu T/S**



**sea surface temperature (MW)**
GMI, AMSR2, WindSat



**Altimetry**
Jason-2, Jason-3, Sentinel-3a,
Cryosat-2, SARAL



# Marine UFOs

Marine observation operators in UFO:

- altimetry (absolute dynamic topography)
- insitu temperature (insitu / potential temperature conversion)
- sea ice fraction
- sea ice thickness
- sea surface temperature
- sea surface salinity

} A simple instance of `ufo::ObsIdentity`

- coolskin SST
- GMI radiance with CRTM
- SMAP radiance with CRTM

} on our todo list

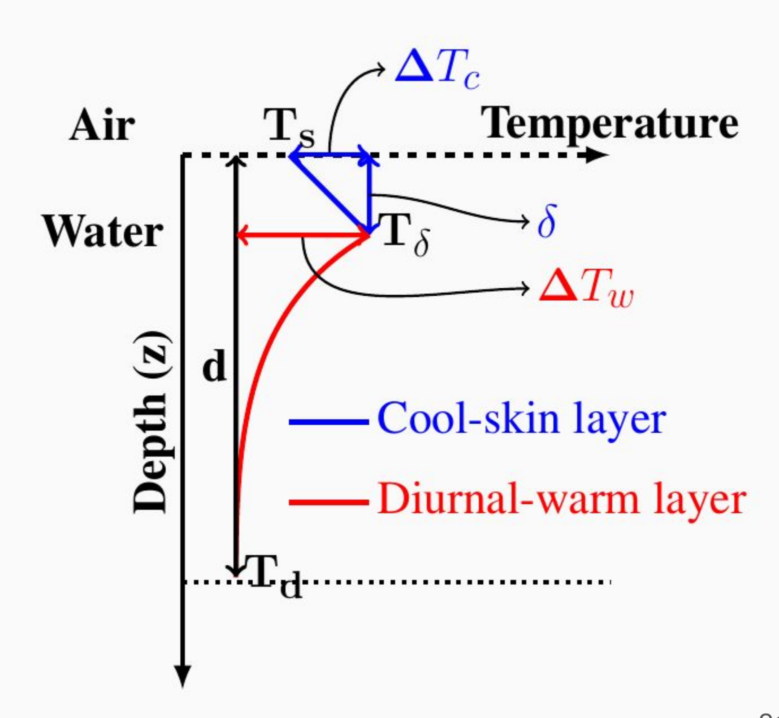
# Marine UFO - cool skin

Perhaps our most complex marine UFO so far, uses surface ocean **and** atmospheric fields:

- `sea_surface_temperature`
- `net_downwelling_shortwave_radiation`
- `upward_latent_heat_flux_in_air`
- `upward_sensible_heat_flux_in_air`
- `net_downwelling_longwave_radiation`
- `friction_velocity_over_water`

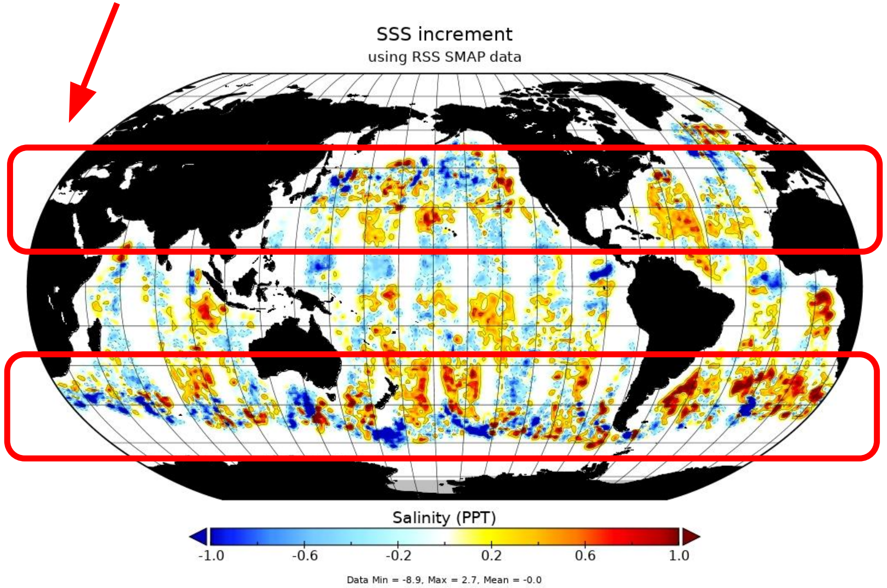
3DVAR then produces an increment for the atmospheric fields

(ignored for now, but useful in coupled DA?



# QC methods

SMAP salinity assimilation showed large / noisy increments where SST is too cold




```yaml
- ObsSpace:
    name: SeaSurfaceSalinity
    ObsDataOut: {obsfile: ./Data/sss.out.nc}
    ObsDataIn:  {obsfile: ./Data/sss.nc}
    simulate:
      variables: [sea_surface_salinity]
  ObsOperator:
    name: SeaSurfaceSalinity
  Covariance:
    covariance: diagonal
  ObsFilters:
  - Filter: Domain Check
    Where:
    - variable: sea_area_fraction@GeoVaLs
      minvalue: 0.5
  - Filter: Domain Check
    where:
    - variable: sea_surface_temperature@GeoVaLs
      minvalue: 15
```

# QC methods

SMAP salinity assimilation showed large / noisy increments where SST is too cold.

QC filters already in place to filter out SSS observations based on background SST. No code needed!

```yaml
- ObsSpace:
    name: SeaSurfaceSalinity
    ObsDataOut: {obsfile: ./Data/sss.out.nc}
    ObsDataIn:  {obsfile: ./Data/sss.nc}
    simulate:
      variables: [sea_surface_salinity]
  ObsOperator:
    name: SeaSurfaceSalinity
  Covariance:
    covariance: diagonal
  ObsFilters:
  - Filter: Domain Check
    Where:
    - variable: sea_area_fraction@GeoVaLs
      minvalue: 0.5
  - Filter: Domain Check
    where:
    - variable: sea_surface_temperature@GeoVaLs
      minvalue: 15
```

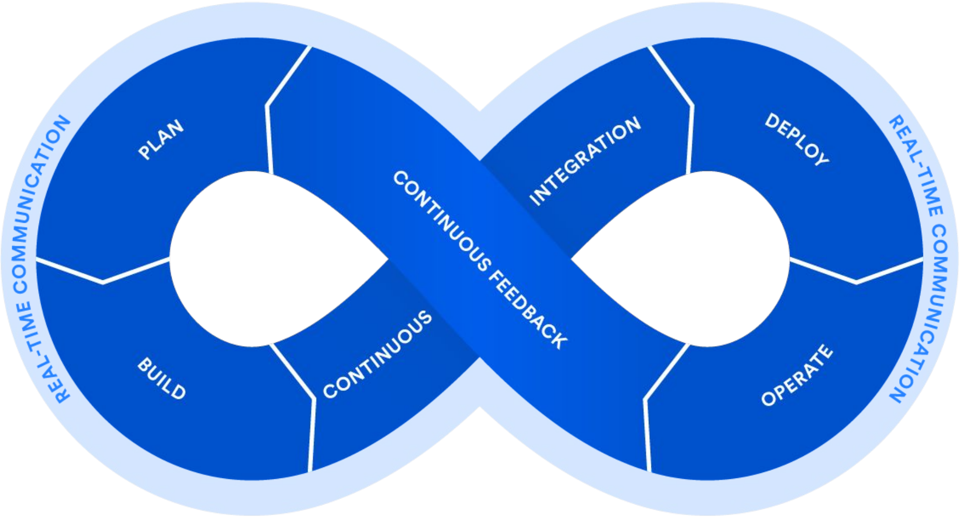
# Realtime marine DA

**“a real-time demonstration of what JEDI is capable of”**

Goal is to have a 1° to ¼° ocean/ice model running in “real-time”, using the latest stable codebase.

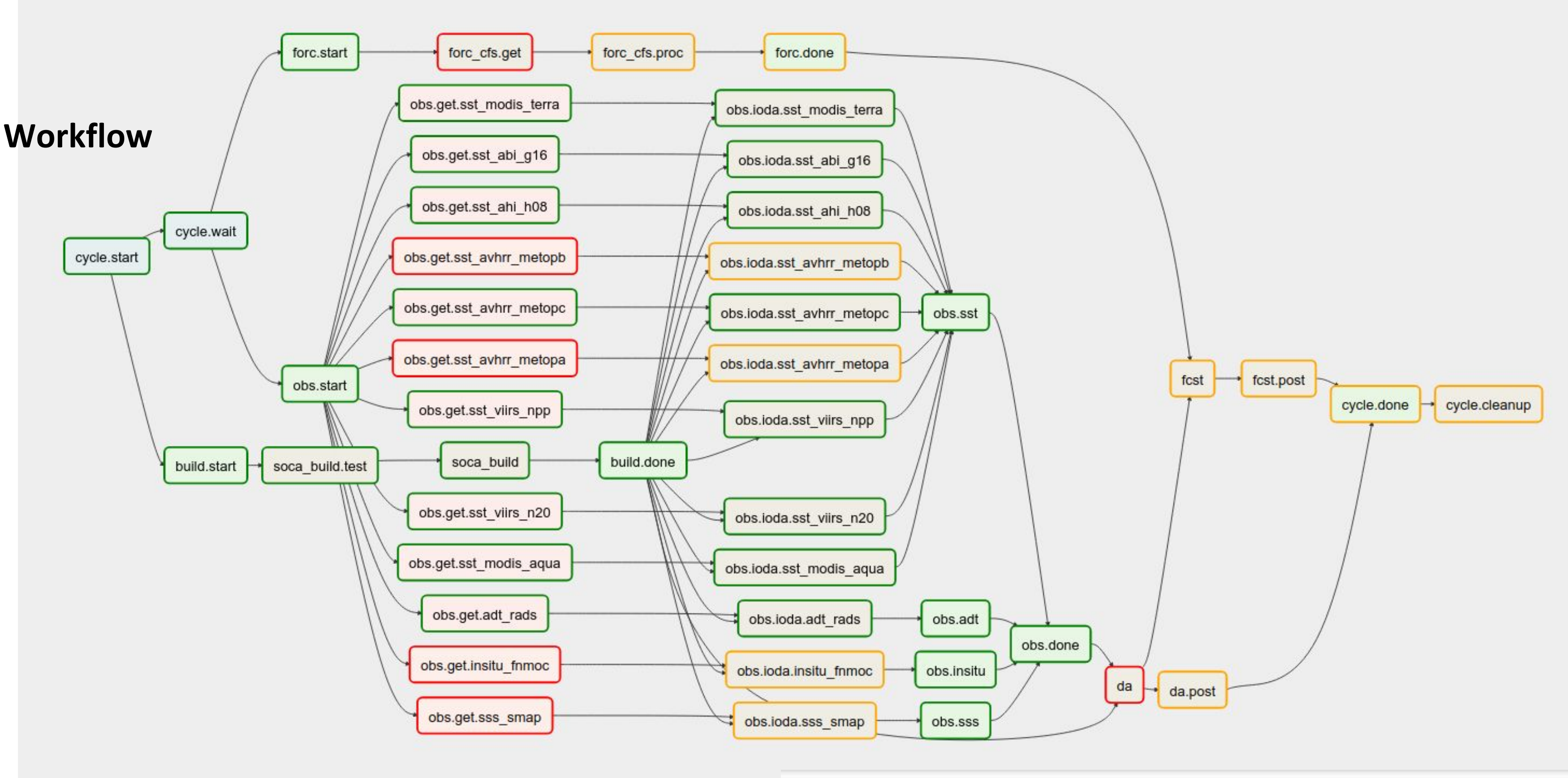
Currently running on a local server, but will be transitioned to **Amazon Cloud** and **Travis-CI**

DevOps - Continuous Delivery



# Realtime marine DA - workflow

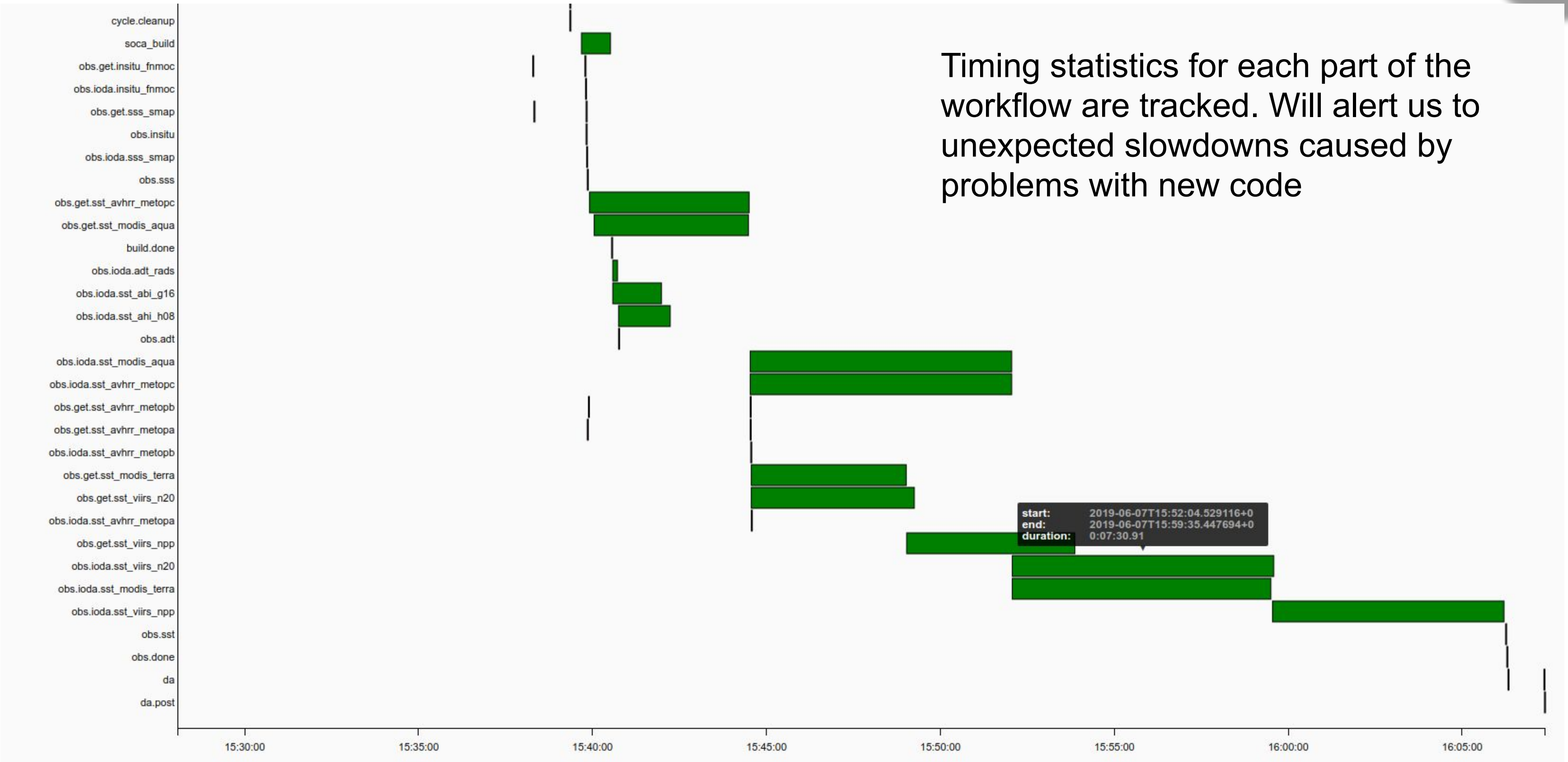
**Airflow Workflow**




BashOperator | CycleWait | DummyOperator | PythonOperator | TimeDeltaSensor

success | running | failed | skipped | up_for_reschedule | up_for_retry | queued | no_status

# Realtime marine DA - workflow




Timing statistics for each part of the workflow are tracked. Will alert us to unexpected slowdowns caused by problems with new code

# SOCA - ctests

```
Test project /home/tsluka/work/jedi-soca/soca.build/soca
 1/27 Test  #1: soca_mains_coding_norms .........   Passed    0.11 sec
 2/27 Test  #2: soca_src_coding_norms ...........   Passed    0.55 sec
 3/27 Test  #3: test_soca_forecast_identity ......   Passed    0.52 sec
 4/27 Test  #4: test_soca_forecast_mom6 .........   Passed    1.28 sec
 5/27 Test  #5: test_soca_socaerror_init ........   Passed    1.02 sec
 6/27 Test  #6: test_soca_enspert ...............   Passed    1.88 sec
 7/27 Test  #7: test_soca_geometry ..............   Passed    0.43 sec
 8/27 Test  #8: test_soca_state .................   Passed    0.37 sec
 9/27 Test  #9: test_soca_modelaux ..............   Passed    0.35 sec
10/27 Test #10: test_soca_model .................   Passed    1.29 sec
11/27 Test #11: test_soca_increment .............   Passed    0.56 sec
12/27 Test #12: test_soca_errorcovariance .......   Passed    0.48 sec
13/27 Test #13: test_soca_linearmodel ...........   Passed    0.67 sec
14/27 Test #14: test_soca_balance ...............   Passed    0.50 sec
15/27 Test #15: test_soca_bkgerrfilt ............   Passed    0.38 sec
16/27 Test #16: test_soca_bkgerrsoca ............   Passed    0.47 sec
17/27 Test #17: test_soca_bkgerrgodas ...........   Passed    1.83 sec
18/27 Test #18: test_soca_vertconv ..............   Passed    0.42 sec
19/27 Test #19: test_soca_ensvariance ...........   Passed    0.48 sec
20/27 Test #20: test_soca_dirac_soca_cov ........   Passed    1.01 sec
21/27 Test #21: test_soca_hofx3d ................   Passed    0.67 sec
22/27 Test #22: test_soca_hofx ..................   Passed    1.92 sec
23/27 Test #23: test_soca_enshofx ...............   Passed    3.75 sec
24/27 Test #24: test_soca_3dvarsoca .............   Passed    2.13 sec
25/27 Test #25: test_soca_3dvargodas ............   Passed    2.53 sec
26/27 Test #26: test_soca_checkpointmodel .......   Passed    0.51 sec
27/27 Test #27: test_soca_3dvarfgat .............   Passed    8.68 sec

100% tests passed, 0 tests failed out of 27
Total Test time (real) =  34.86 sec
```

As part of the workflow, the latest **develop** branch for every used repository on github is tested every night

If all tests pass, these branches are marked as a **stable nightly release**, and used for real time cycles.

Obviously the quality and speed of SOCA tests are very important!

# SOCA - ctests

```
23: < Test      : CostJo    : Nonlinear Jo(SeaSurfaceSalinity) = 66.2894, nobs = 51, Jo/n = 1.29979, err = 1
23: > Test      : CostJo    : Nonlinear Jo(SeaSurfaceSalinity) = 0 --- No Observations
23: < Test      : CostFunction: Nonlinear J = 29783.1
23: > Test      : CostFunction: Nonlinear J = 29716.8
23: --------------------------------------------------------------------------
1/1 Test #23: test_soca_3dvargodas .............***Failed    4.61 sec
```

Here, a bug was introduced to the filtering of the sea surface salinity observations.

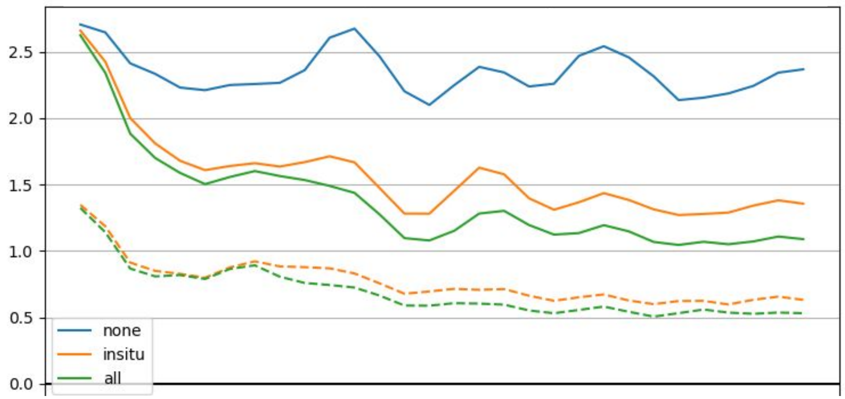
The ctests fail because answers have changed. And the log files help point to the cause of the change of answers

# SOCA cycles

After April 2019 code sprint, we had all the pieces in place to do long 3DVAR cycles showing performance with **no obs**, **only insitu**, and **insitu+satellite** obs.

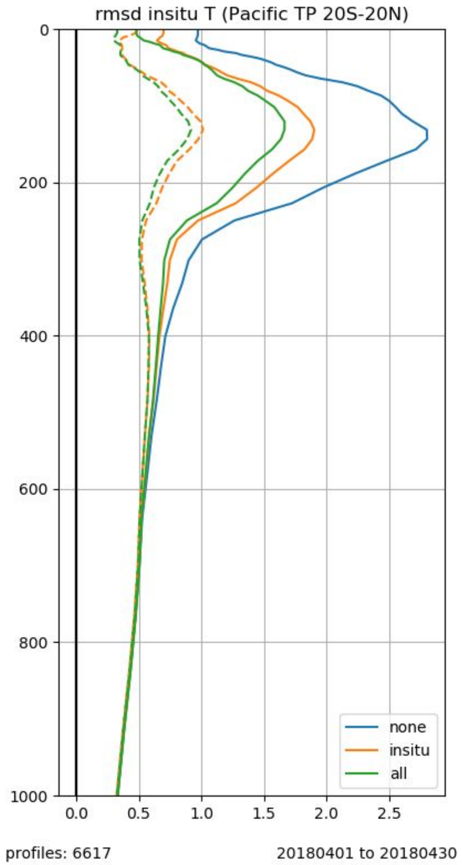
Not much science, but shows that the *completely untuned* system is starting to work.

Insitu T O-F  RMSD in Pacific EQ




April 1, 2018

April 30, 2018



# SOCA - near term goals

- Prototype ocean/ice system working at ¼ degree for use at NOAA/EMC NASA/GMAO
- "real-time" continuous deployment demonstration, running on cloud

**Code sprints**

- Marine IODA/UFO improvement (April 2019)
- Marine model interfaces (WaveWatch III, CICE5/6)
- Multi-domain UFO

longer term goals

- 1/12 degree MOM6 configuration (RTOFS at NOAA/NCEP?)
- coupled DA - (Coupled atm/ocn H(x), weakly/strongly coupled DA )