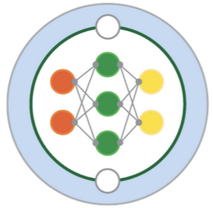
Leveraging couplers for online inference: Recent experience with NEMO

Alexis Barge — Julien Le Sommer

13th November 2024



Context: interface for hybrid physics / AI modeling

Typical use cases of hybrid modeling

Parameterization from hi-fidelity models (LES, km-scale models)

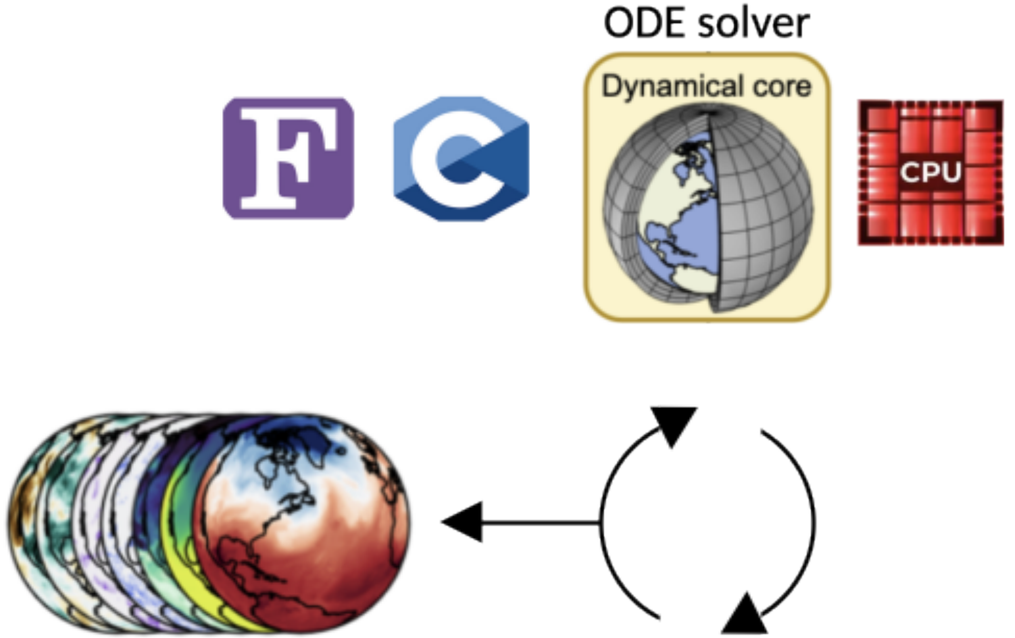
Sane et al. 2023 *Zhang et al. 2023* *Yuval et al. 2021*

Model error correction from reanalysis or DA increments

Gregory et al. 2024 *Chapman and Berner, 2023*

Acceleration of code components with neural emulators

Hogan and Bozzo, 2018 *Chantry et al. 2021*



Existing solutions

Implement or convert NN in Fortran

neural-fortran *Fortran-Keras Bridge* *FNN* *inference-engine*

Call Python scripts from Fortran with Python bindings

call_py_fort *Forpy*

Leverage the existing C/C++ bindings of specific ML libraries

infero *FTorch* *TF-lib* *TorchClim*

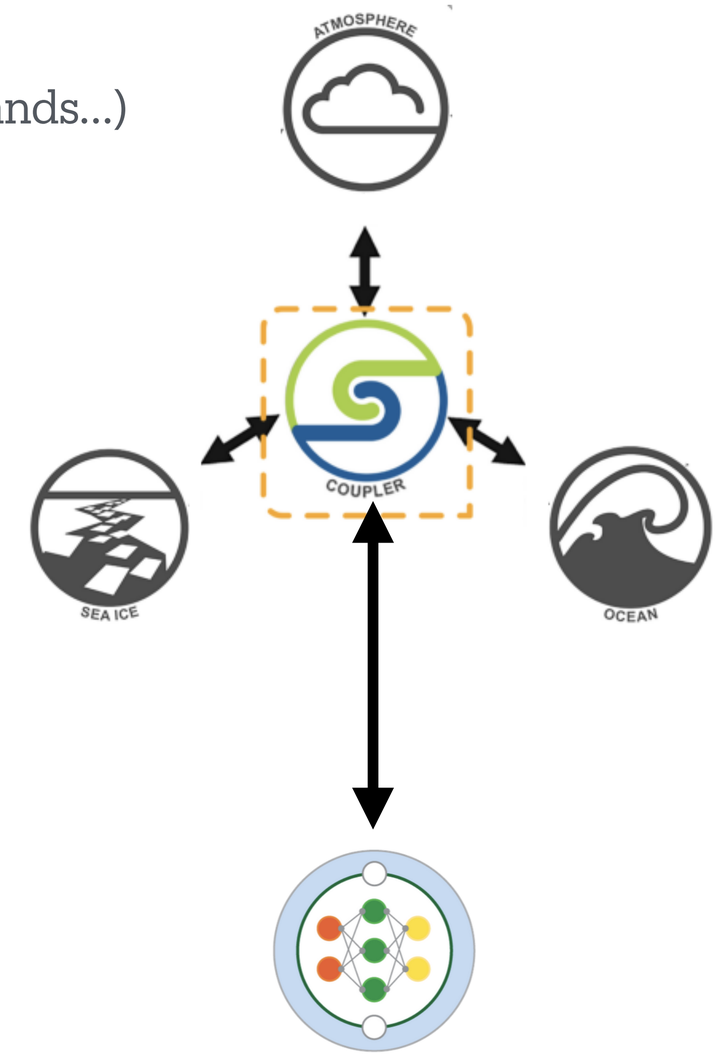
Leverage high-level couplers Fortran and Python APIs

SmartSim *PhyDLL* *Eophis (OASIS)*

Couplers

Earth-System Modeling (ESMs)

Models composed of different sub-models (ocean, atmosphere, sea-ice, waves, lands...)
Tie all together in one single software
Performed by couplers



Hybrid modeling for ESMs

Assemble ML components as standard sub-model

Couplers are already in codes

Low development to adapt existing interfaces

OASIS3-MCT : 5 of the 7 European ESMs in CMIP6
YAC : ICON
CMEPS : NCAR CESM — NOAA UFS
FMS : NOAA GFDL
CPL7/MCT : E3SM
C-COUPLER2 : Chinese institutions models

EOPHIS: a library for deploying ML models through OASIS

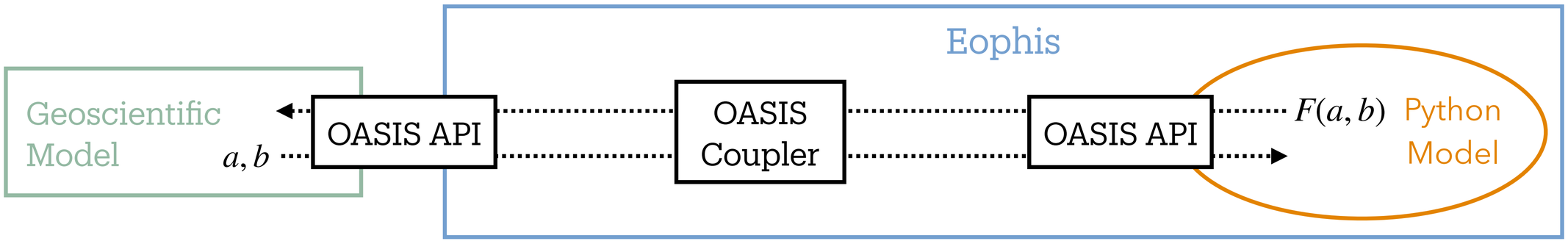
OASIS3-MCT5 https://oasis.cerfacs.fr/en

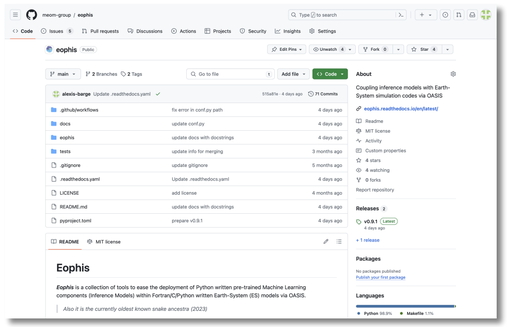
Coupling library between different codes
Interpolate and exchange 2D/3D fields
Python, C/C++ and Fortran API

Widely deployed in European models

Eophis

Deploy OASIS API in Python scripts
Manage connexions between exchanges and ML models
Configure coupling environment
~ 1.5k lines





https://github.com/meom-group/eophis

MIT Licensed , Open Source

Realizations

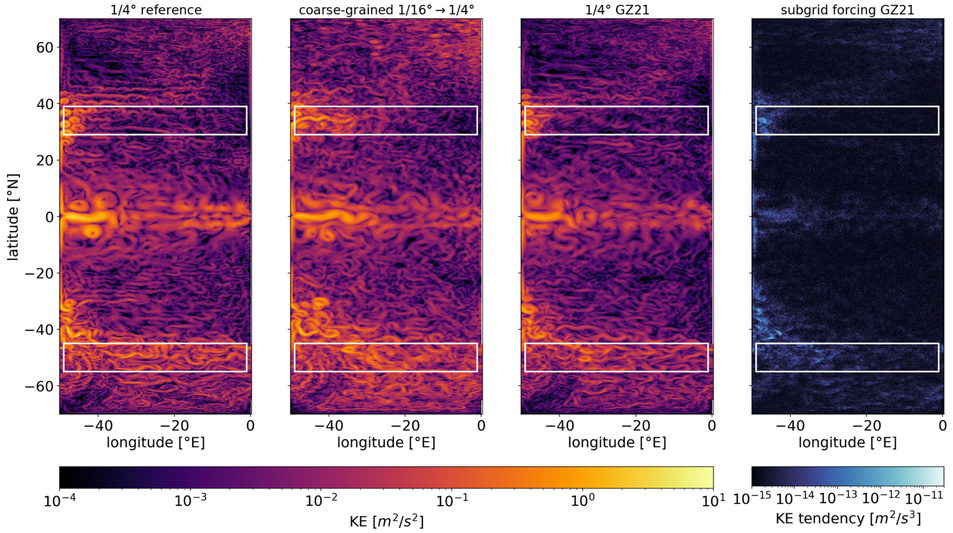
NEMO-DINO.GZ21

Idealized config

Stochastic subgrid forcing

Infer stochastic elements from 2D velocities

Guillaumin and Zanna, 2021



More with David's presentation

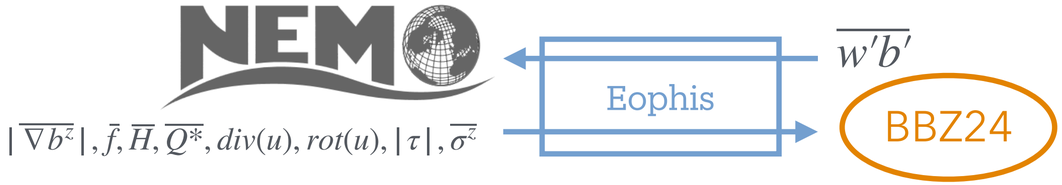
eORCA025-MLE.BBZ24

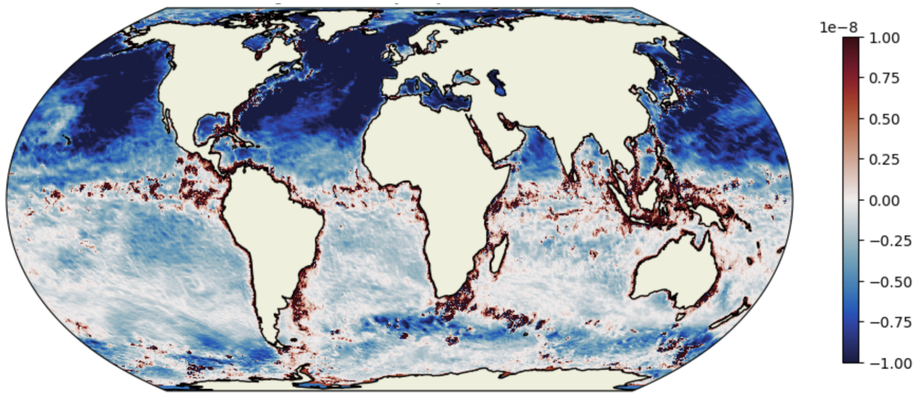
Realistic global 1/4° NEMO config

MLE parameterization

Infer 2D vertical buoyancy fluxes

Bodner, Balwada and Zanna, 2024



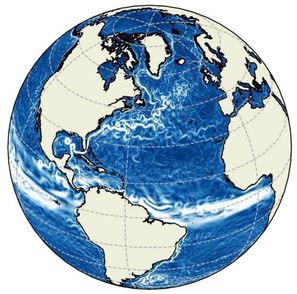


Snapshot of outsourced vertical buoyancy flux (W/m2)

More with Marcella's presentation

Flexible load balancing

eORCA025-MLE.BBZ24



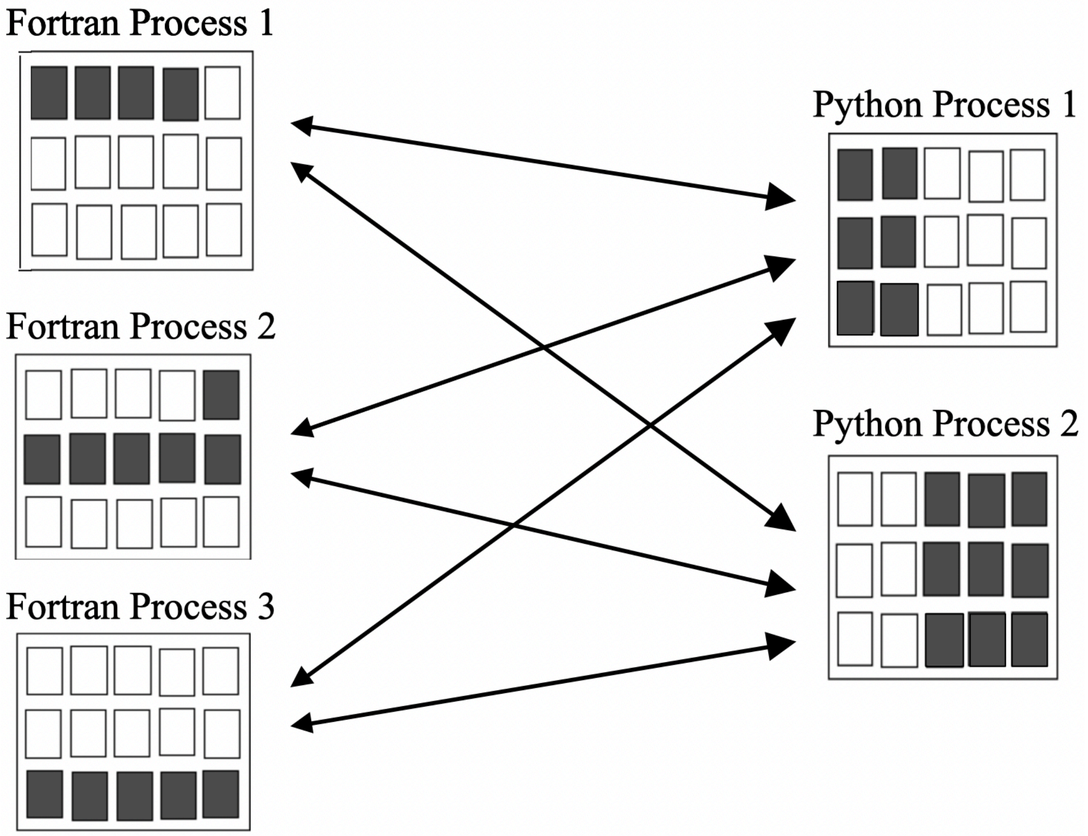
999 NEMO processes (CPUs)
Vary Python processes

Python communications on CPUs
Inference on GPUs if available
1 GPU per group of 40 Python CPUs

Halos created during communications

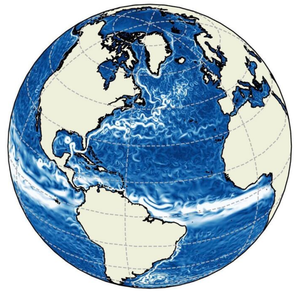
Adjustment of resources

Optimal process number for model and inference
Isolate GPU access to Python processes
Flexibility (fast, smooth, cheap…)



Flexible load balancing

eORCA025-MLE.BBZ24



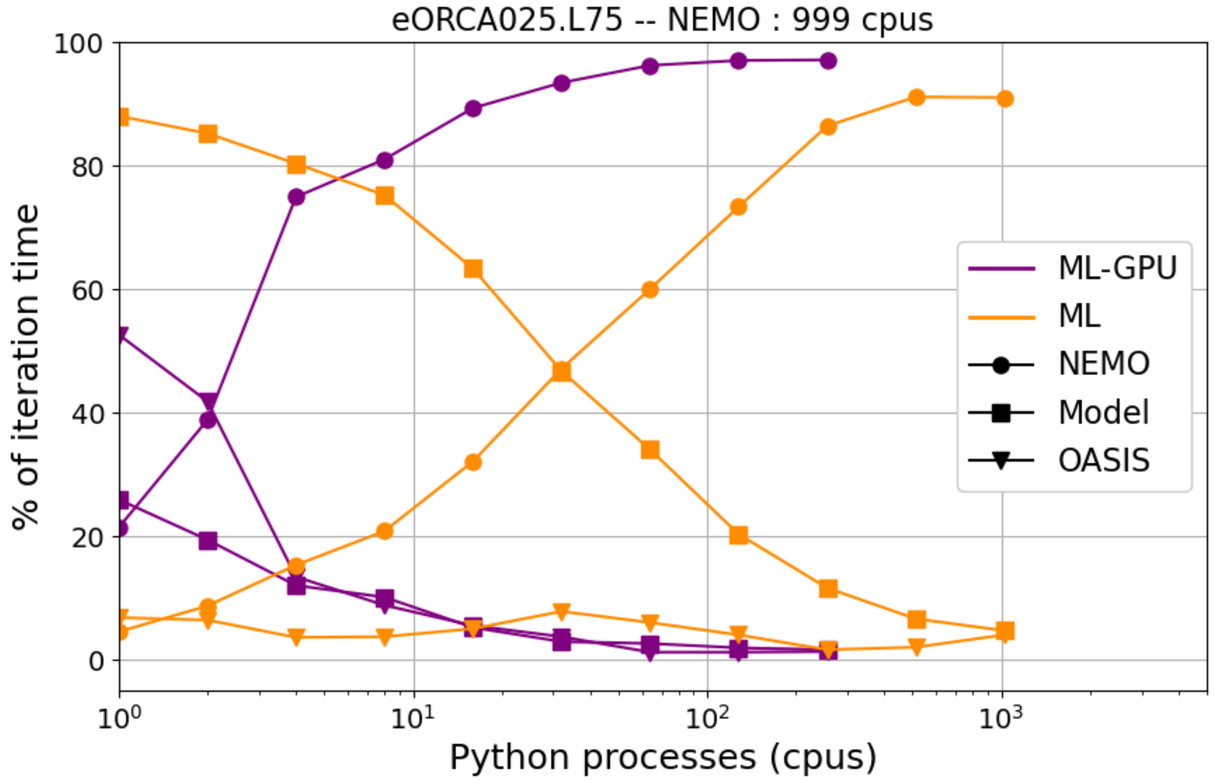
999 NEMO processes (CPUs)
Vary Python processes

Python communications on CPUs
Inference on GPUs if available
1 GPU per group of 40 Python CPUs

Halos created during communications

Adjustment of resources

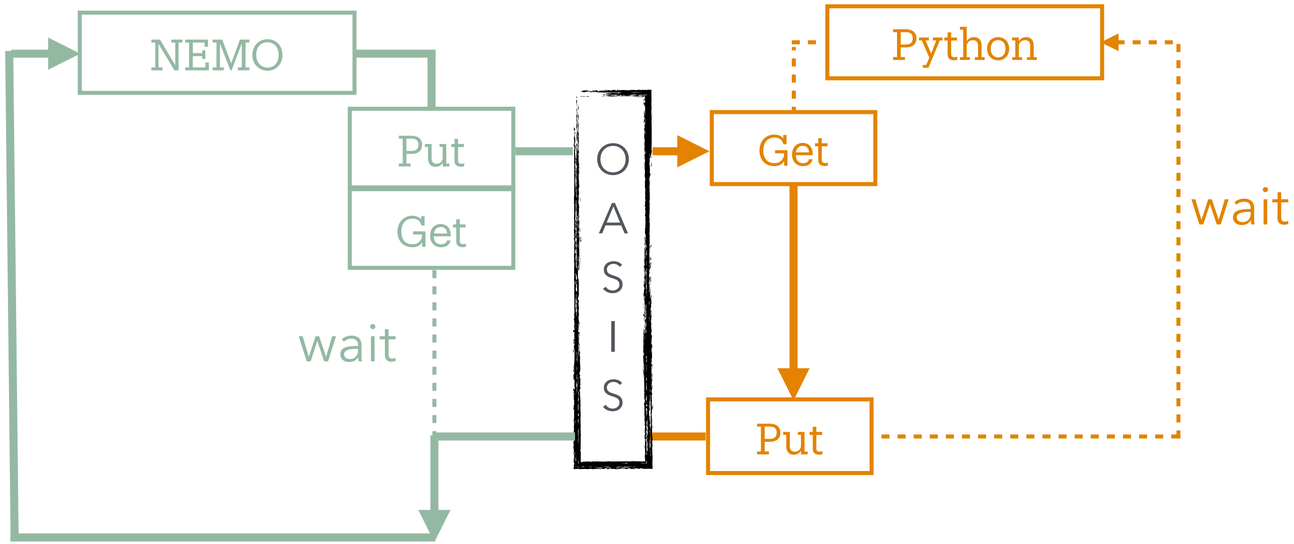
Optimal process number for model and inference
Isolate GPU access to Python processes
Flexibility (fast, smooth, cheap…)



Asynchronous inference

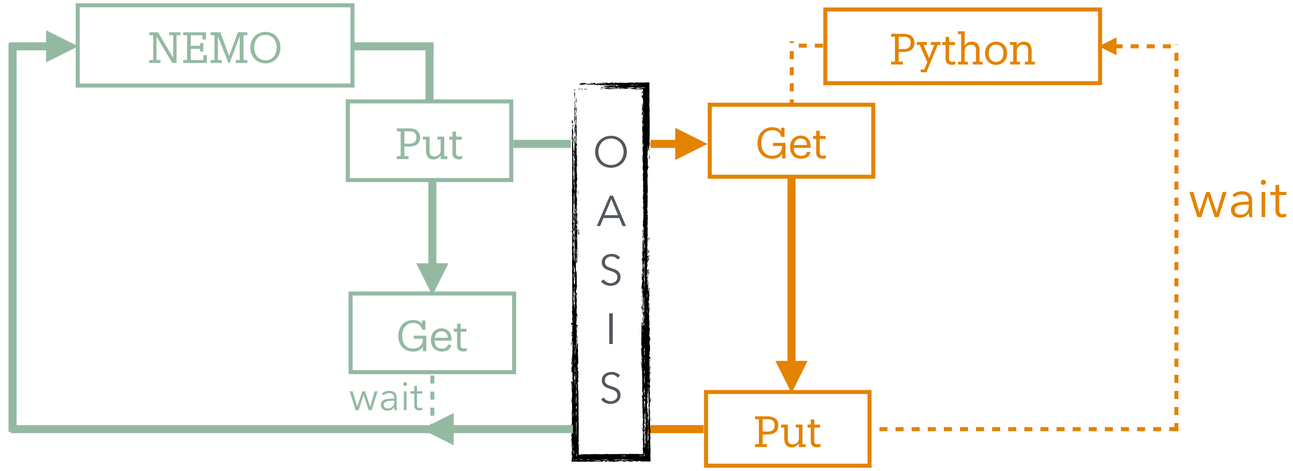
Synchronous (-S)

Send inputs
Wait for Python returns
Continue computation



Asynchronous (-A)

Send inputs as soon as possible
Continue computation as far as possible
Wait for Python returns



Asynchronous inference

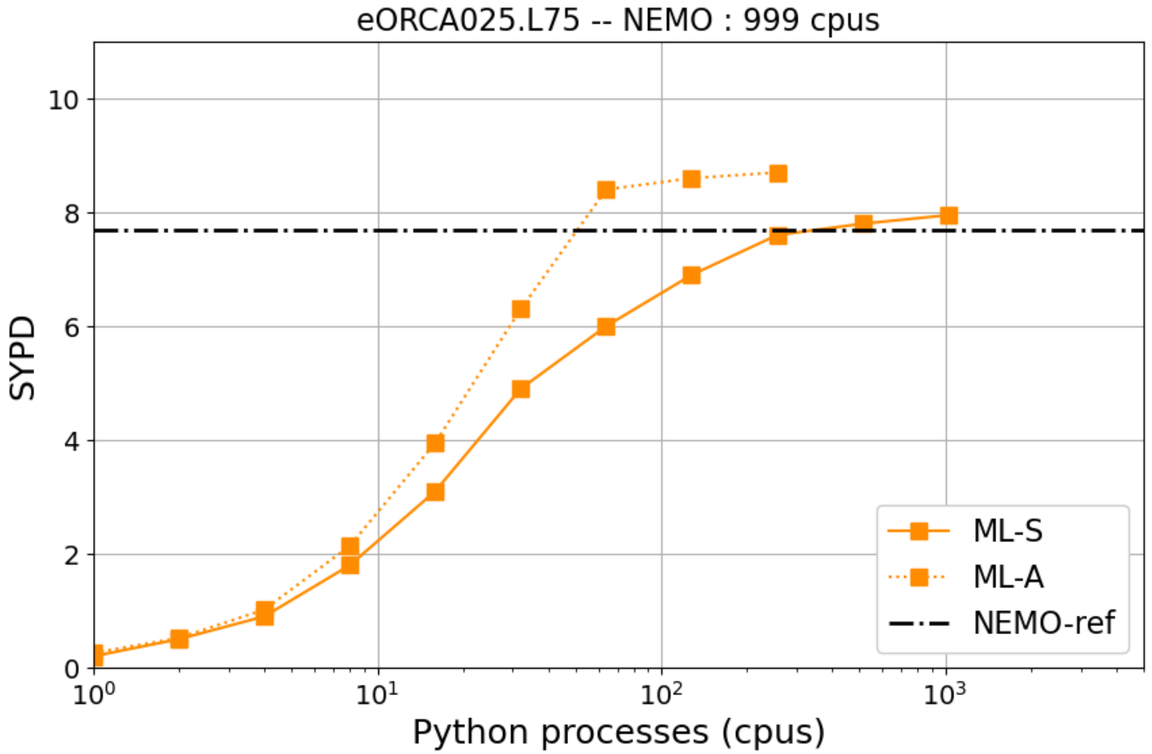
Synchronous (-S)

Send inputs

Wait for Python returns

Continue computation

Irreducible overhead for a given SYPD



Asynchronous (-A)

Send inputs as soon as possible

Continue computation as far as possible

Wait for Python returns

Large reduction in resources overhead

Optimal resources overhead	Synchronous	Asynchronous
ML	20-30%	4-6%

Asynchronous inference

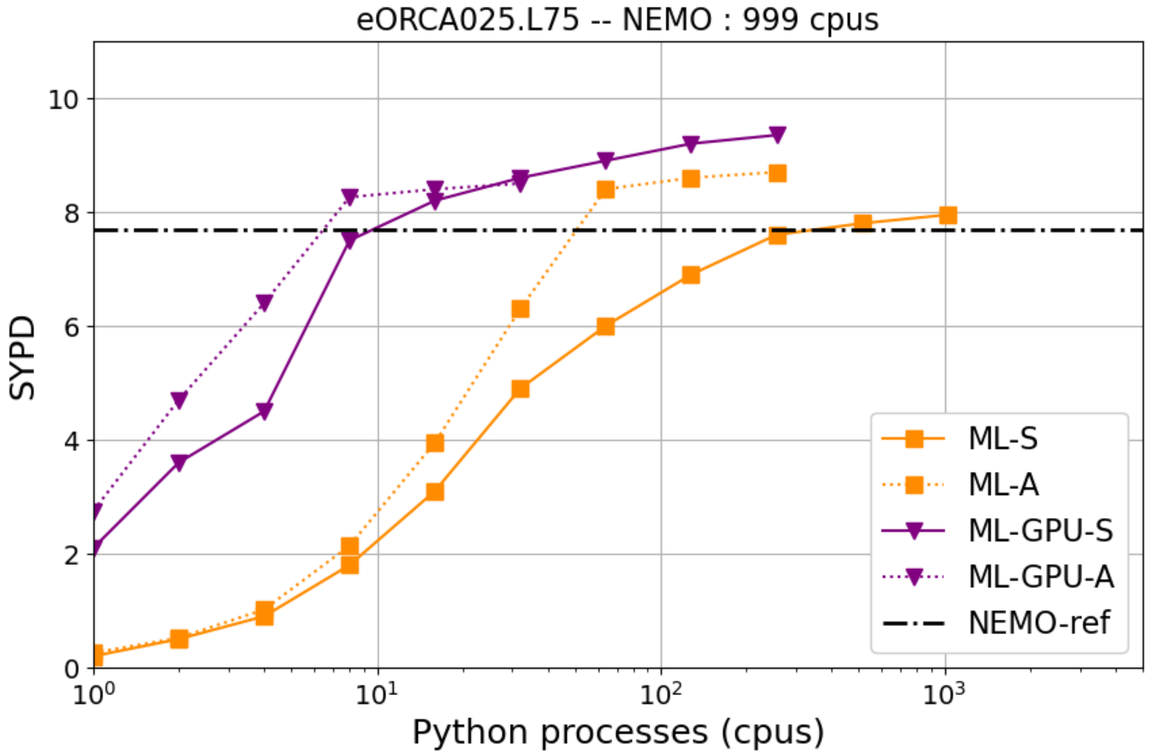
Synchronous (-S)

Send inputs

Wait for Python returns

Continue computation

Irreducible overhead for a given SYPD



1 GPU per group of 40 Python CPUs

/!\ Machine dependent

Asynchronous (-A)

Send inputs as soon as possible

Continue computation as far as possible

Wait for Python returns

Large reduction in resources overhead

Optimal resources overhead	Synchronous	Asynchronous
ML	20-30%	4-6%
ML-GPU	1-3%	< 1%

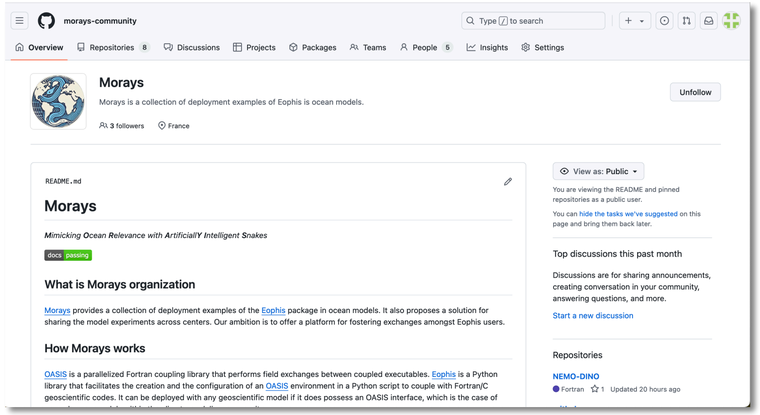
Wrap-up

Benefit of leveraging couplers

Flexible resources allocation
Ability to run parallel inference
Asynchronicity
(Flexible prototyping of ML components)
(Inference separated from model)

Perspectives

Evaluation of performances for 3D coupling



Find more on GitHub Morays-Community

Use cases of ML closures in ocean models (NEMO, CROCO) with Eophis
Examples deployment and tutorials

https://github.com/morays-community