
# Dask-Enabled External Tasks For In Transit Analytics

Amal Gueroudji[1,2], Julien Bigot [1], Bruno Raffin[2]
amal.gueroudji@cea.fr, julien.bigot@cea.fr, bruno.raffin@inria.fr

[1] Maison de la Simulation, UVSQ, CNRS, CEA, Universite Paris-Saclay
[2] Inria, CNRS, Grenoble INP, LIG, Univ. Grenoble Alpes

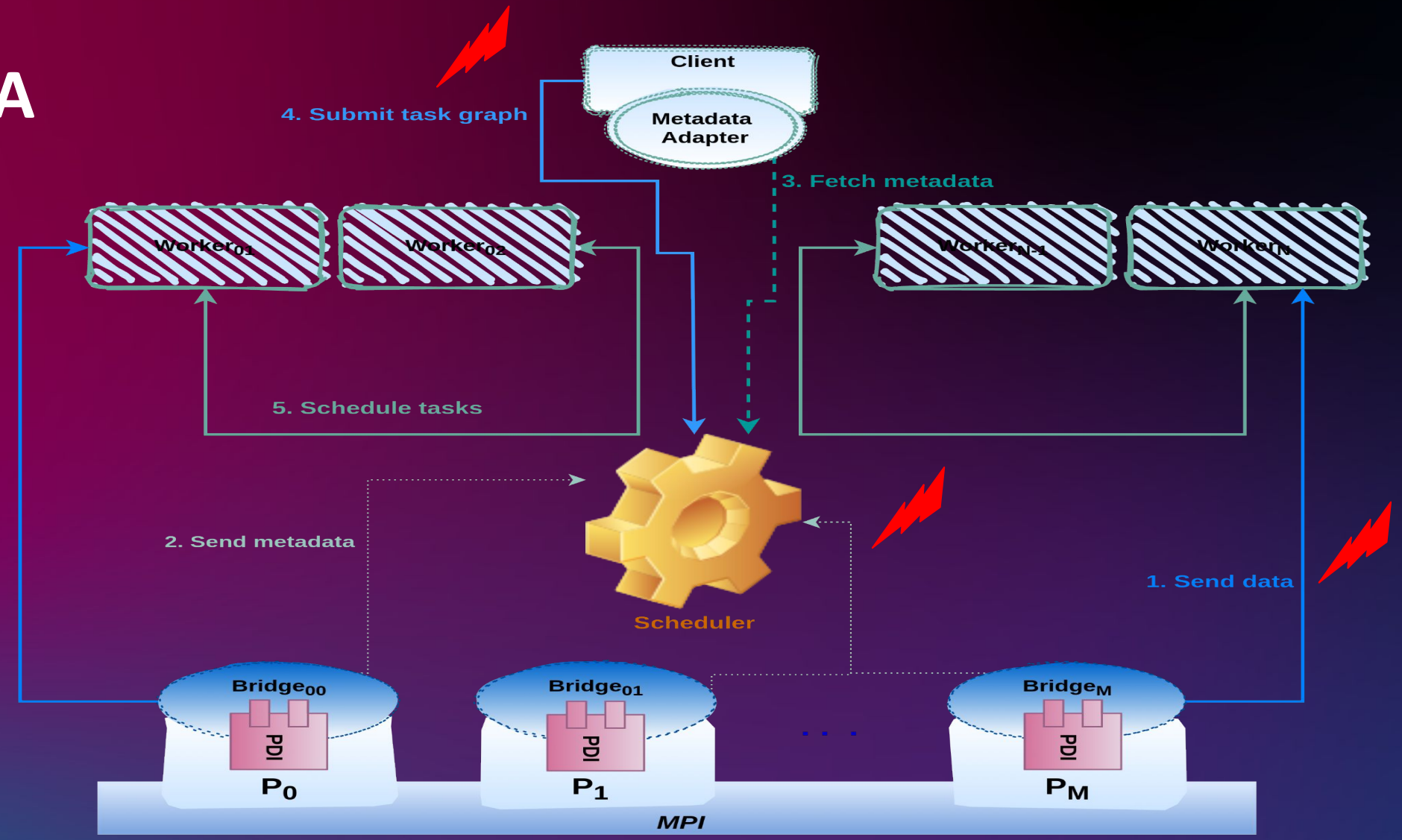
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 800945 — NUMERICS — H2020-MSCA-COFUND-2017

# Context

- Post hoc workflows suffer from IO bottleneck
- In situ workflows are complicated to setup

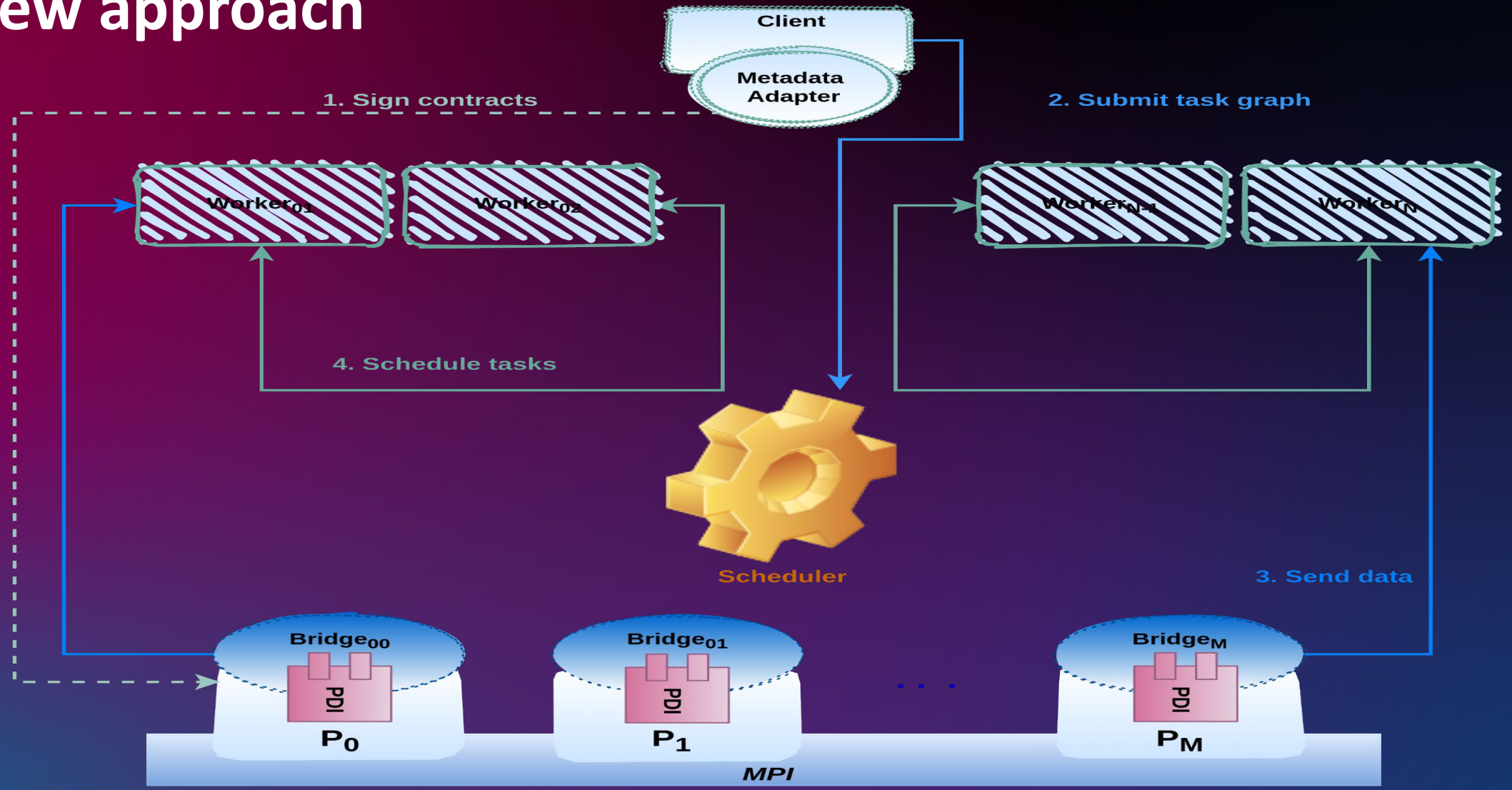
- Bring together post hoc easiness and in situ performance
- Couple MPI simulations with Dask distributed analytics

# DEISA




A. Gueroudji, J. Bigot and B. Raffin, "DEISA: Dask-Enabled In Situ Analytics," 2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics (HiPC), 2021, pp. 11-20, doi: 10.1109/HiPC53243.2021.00015.

# Our new approach



# Our contributions

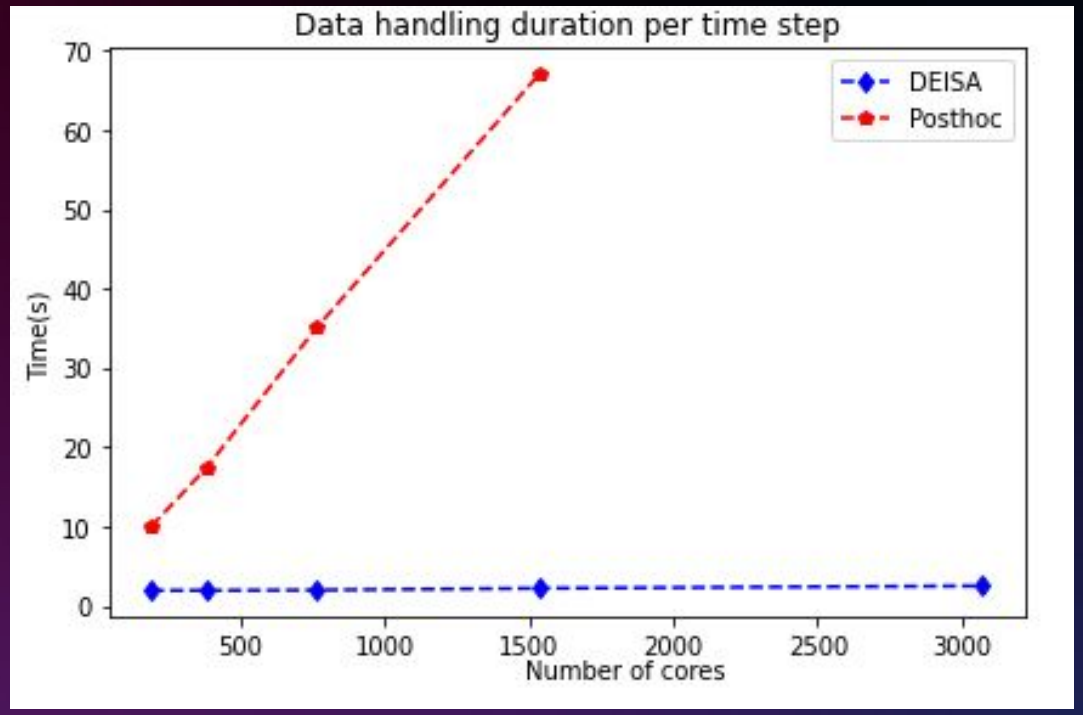
- Ahead of time task-submission
  - Overlapping tasks from different timesteps
  - Reducing the load of the scheduler
  - Scaling better with less latency
- Automatic management of task dependency over all the dimensions
- Send only needed data by automatically making selections in the spatio-temporal distribution
  - Better performance by avoiding unnecessary communications

# Preliminary results

- IRENE supercomputer @ TGCC, France,
- Nodes:
  - 2x24-cores Intel Skylake@2.7GHz
  - 180GB RAM
- InfiniBand network (100GB/s),
- Lustre PFS: 70GB/s transfer rate
- Mini App 2D heat solver

| Parameter | Value |
|---|---|
| MPI nodes / Dask worker node | 4 |
| MPI process / MPI node | 2 |
| Dask worker / Dask worker node | 8 |
| Thread / Dask worker | 6 |
| MPI process / Dask worker | 1 |
| Data size / MPI process | 1 GiB |
| Data size / MPI node | 2 GiB |
| Mean data size / Dask worker node | 8 GiB |
| Used protocol | UCX |
| Analytics | Norm2 of a temporal derivative |

TABLE I: Fixed parameters used in the evaluation




- Almost perfect week scaling for Deisa
- With exactly the same posthoc algorithm

# To read more

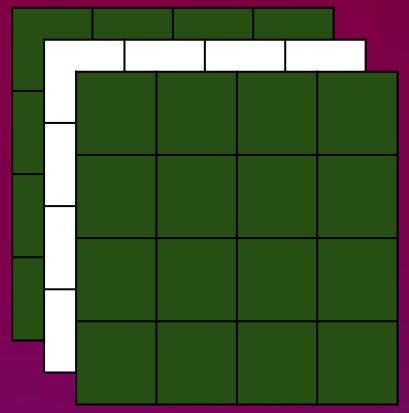

- **DEISA paper:** A. Gueroudji, J. Bigot and B. Raffin, "DEISA: Dask-Enabled In Situ Analytics," 2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics (HiPC), 2021, pp. 11-20, doi: 10.1109/HiPC53243.2021.00015.
- **PDI: https://pdi.dev/**
- **Deisa pypi: https://pypi.org/project/deisa/**
- **DEISA github: https://github.com/GueroudjiAmal/deisa**

Amal Gueroudji, Julien Bigot , Bruno Raffin
amal.gueroudji@cea.fr, julien.bigot@cea.fr, bruno.raffin@inria.fr

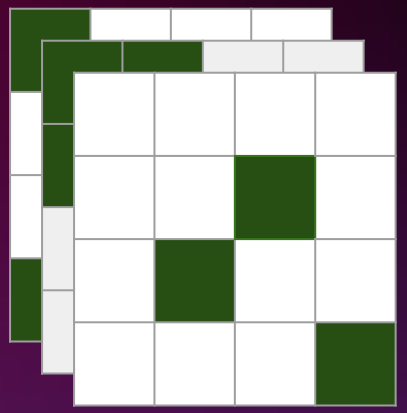
# Our contributions

- Improve data model by introducing deisa virtual arrays
- Introduce contract to reduce data communications
- Introduce external tasks in Dask distributed integrate transparently simulation data in Dask task graphs

# Our contributions



***Before***



***Now[contracts]***

- Send all generated data by time step

- Send only needed data by automatically making selections in the spatio-temporal distribution
  - Better performance by avoiding unnecessary communications

# Conclusions

**In this talk:**

- Improvements in Deisa design for both performance and productivity

**Next steps:**

- Further experiments and comparison to the old version of Deisa
- Production use case Gysela

# Implementation[Configuration]

```
plugins :
    mpi : # get MPI rand and size
    deisa :
        scheduler_info : scheduler . json
        init_on : init
        time_step : $step
         deisa_arrays : # Deisa Virtual arrays
            G_temp : # Field name
                type : array
                subtype : double
                 size :[ timedim,  ' $cfg . loc [0] * ( $rank % $cfg . proc [0]) ',  ' $cfg . loc [1] * ( $rank / $cfg . proc [0]) ']
                 subsize : [ 1, ' $cfg . loc [0] ', ' $cfg . loc [1] '] # chunk size
                 start : $step, ' $cfg . loc [0] * ( $rank % $cfg . proc [0]) ',  ' $cfg . loc [1] * ( $rank / $cfg . proc [0]) ' ]
                + timedim : 0 # a tag for the time dimension
        map_in : # deisa array mapping
            temp : G_temp
```

# Implementation[Analytics]

```
# Post hoc analytics
 import dask.array as da
 import yaml, json
 from distributed import Client
 import h5py
 client = Client('scheduler.json')
 # Read Dask array from
 file = h5py . File ( ' data . hdf5 ' , mode = 'r ')
 # Select data
 F = file["G_temp"][...]
 dFdt = 2. / 3. * (F[3: - 1] - F[1: - 3] - (F[4:]- F[:- 4]) / 8.)
 Norm = da.linalg.norm(dFdt, axis=(1,2))
 futures = client.persist(Norm)
 client.compute()
```

```
# Deisa in situ analytics
 import dask.array as da
 import yaml, json
 import Deisa
 Deisa = Deisa('scheduler.json', 'config.yml')
 client = Deisa.get_client()
 # Get Deisa Virtual Arrays
 arrays = Deisa.get_deisa_arrays()
 # Select data through contract
 F = arrays["G_temp"][...]
 dFdt = 2. / 3. * (F[3: - 1] - F[1: - 3] - (F[4:]- F[:- 4]) / 8.)
 Norm = da.linalg.norm(dFdt, axis=(1,2))
 futures = client.persist(Norm))
 arrays.validate_contract()
 client.compute()
```