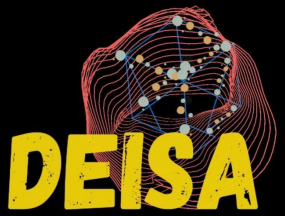


Dask Enabled In Situ Analytics

•••

Coupling MPI Simulations With Dask

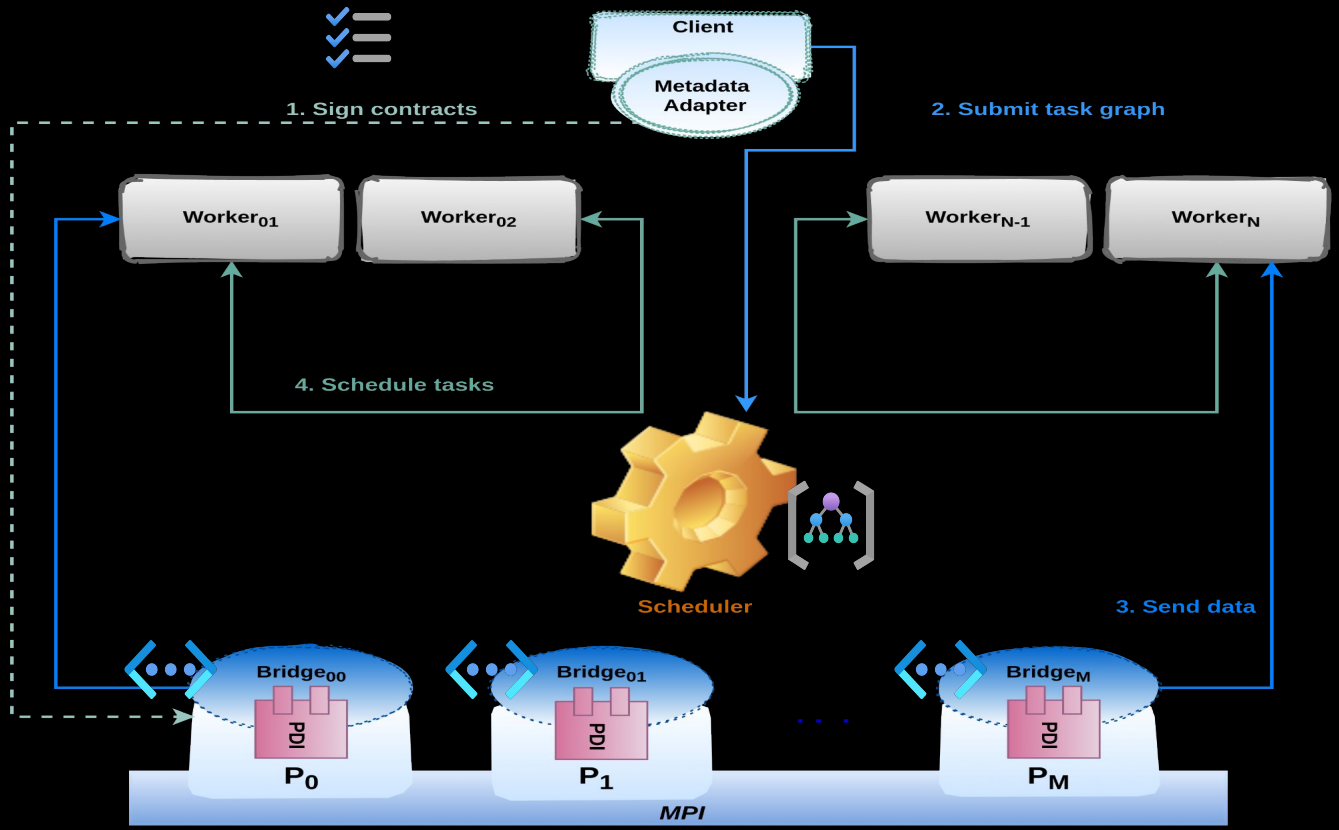
Gueroudji Amal (Radix-io)

DEISA Architecture

Deisa virtual arrays

Contracts

External tasks integration



DEISA [Deisa Plugin Configurations]

```
plugins:
    mpi: # get MPI rand and size
    deisa:
        scheduler_info: scheduler.json
        init_on: init
        time_step: $step
        deisa_arrays: # Deisa Virtual arrays
            G_temp: # Field name
                type: array
                subtype: double
                size:
                    -'$cfg.maxTimeStep'
                    -'$cfg.loc[0] * ($rank % $cfg.proc[0])'
                    -'$cfg.loc[1] * ($rank / $cfg.proc[0])'
                subsize: # Chunk size
                    -1
                    -'$cfg.loc[0]'
                    -'$cfg.loc[1]'
                start:  # Chunk start
                    -$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
```

Initialization

Deisa Virtual Arrays

Map real data to virtual arrays

Single-graph Incremental Time Derivative

```
# 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(futures)
```

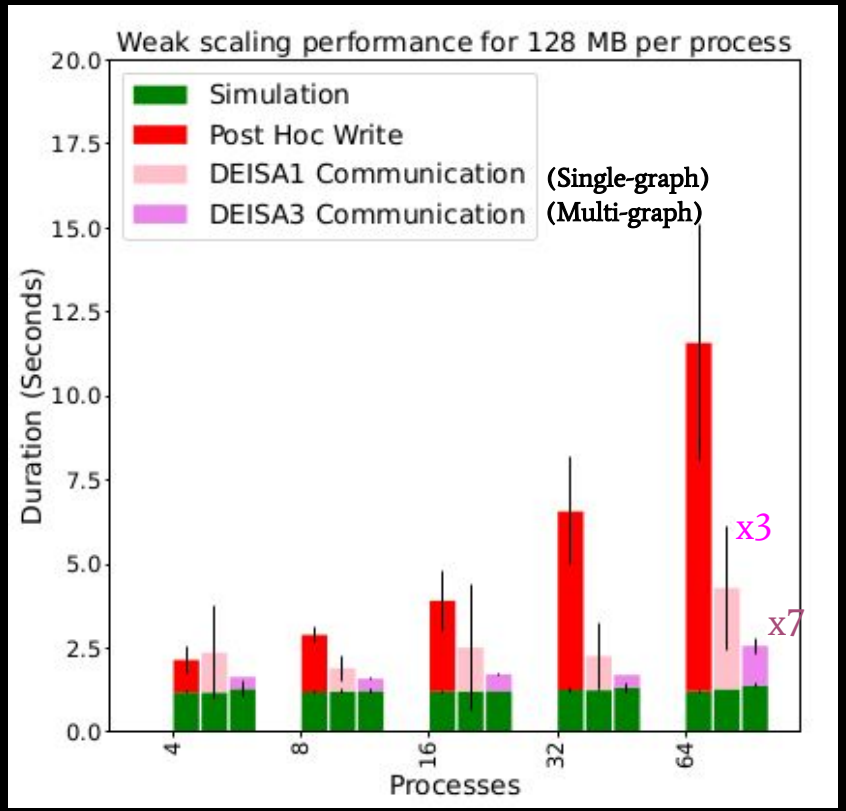
```
# Deisa in situ analytics
import dask.array as da
import yaml, json
import Deisa
Deisa = Deisa('scheduler.json', 'config.yml')
client = Deisa.get_client()
# Get Deisa Virtual Arrays
arrays = Deisa.get_deisa_arrays()
# Select data through contract
F = arrays["G_temp"][...]
dFdt = 2. / 3. * (F[3: -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

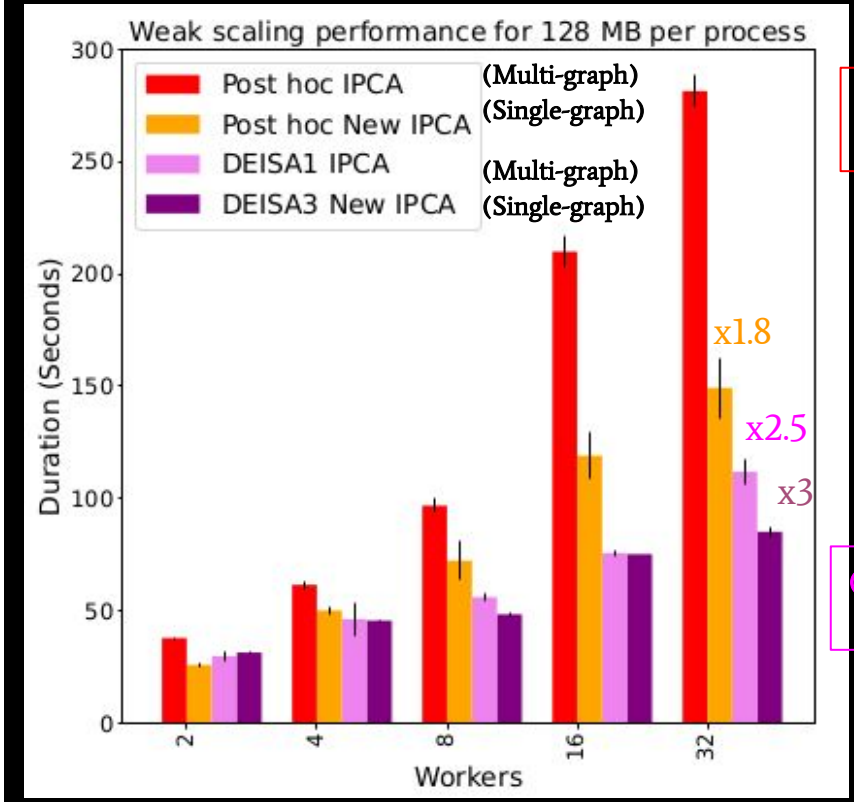
Incremental Dimensionality Reduction PCA

```
# 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"][...]
ipca=InSituIncrementalPCA(n_components=2)
ipca = ipca.fit(gt, ["t", "X", "Y"], ["X"], ["Y"])
explained_variance ,singular_values  =
  client.persist([ pca.explained_variance_ ,
  pca.singular_values_])
 client.compute(futures)
```

```
# 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"][...]
ipca=InSituIncrementalPCA(n_components=2)
ipca = ipca.fit(gt, ["t", "X", "Y"], ["X"], ["Y"])
explained_variance ,singular_values  =
  client.persist([ pca.explained_variance_ ,
  pca.singular_values_])
 arrays.validate_contract()
 client.compute(futures)
```

DEISA vs Post hoc Weak Scalability

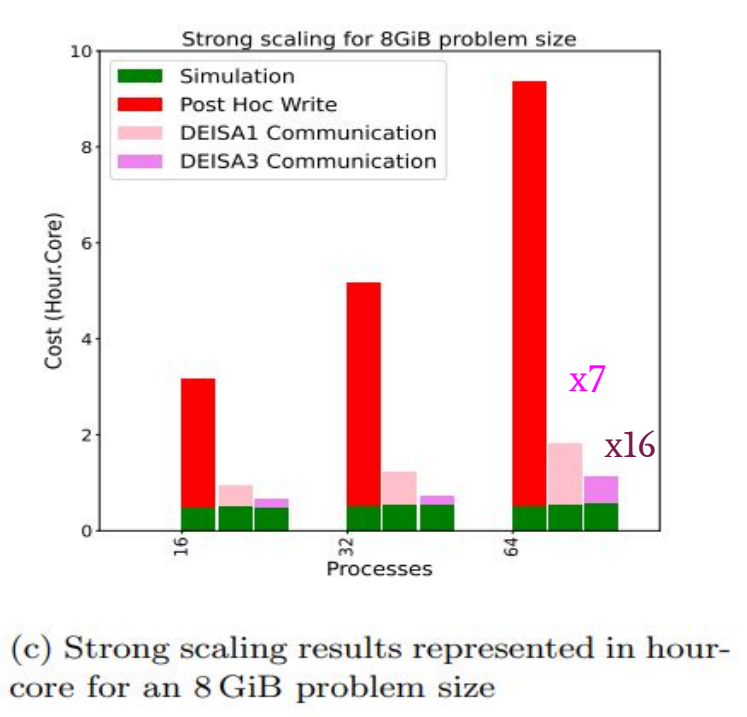




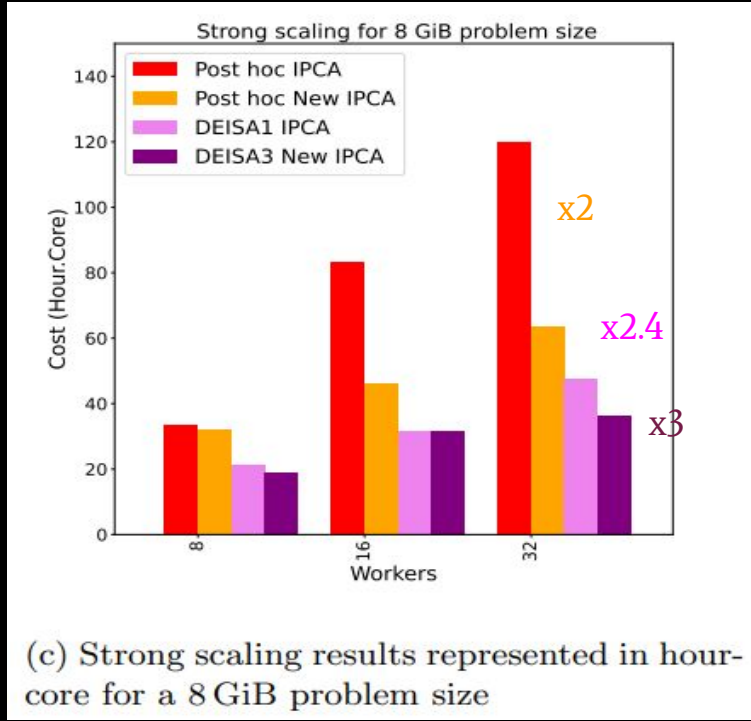
(reading data + Analytics)

(waiting data + Analytics)

DEISA vs Post hoc efficiency in hour.core



(c) Strong scaling results represented in hour-core for an 8 GiB problem size



(c) Strong scaling results represented in hour-core for a 8 GiB problem size