Dask External Tasks
for HPC/ML In Transit Workflows

Amal Gueroudji
Julien Bigot
Bruno Raffin
Robert Ross
HP-C/DA Workflows

**Machine Learning** is everywhere now!

- Data analytics more and more equates to ML-based HPDA
- Streaming & *task-based* tools, often in *Python*, such as *Dask*

**Numerical simulation** is not dead!

- We still need to produce the data
- Fortran is dying? Long live *C++* & *MPI* (+X?) for *parallel* processing

Two worlds, two languages, two sets of tools that need coupling!
Dask distributed?

A Python distributed runtime

Scheduler/workers (+client) model to run work (each on its own process/node)

A task-based model to describe work

Many APIs ported on top of dask

- Numpy => distributed Arrays
- SciPy
- Scikit-learn
- Pandas => distributed Dataframes
- ...
Dask for post hoc analytics

File-system IO performance is an issue
In situ analytics

Simulation…disk…analytics legacy workflow

- Hit the disk **performance bottleneck**!

**Solution**: in situ analytics use the network instead

- Run simulation & analytics concurrently
- Often dedicate some MPI ranks for analytics
  - MPI Communicator system is perfect for that
  - Eg: one per node
  - Or in transit with dedicated nodes

But… MPI is not well suited for HPDA

Can we do better? => **Deisa**!

---

The Deisa1 approach for analytics

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

DEISA Bridge

DEISA Bridge

DEISA Bridge

PDI

PDI

PDI

P_0

P_1

P_M

Worker #1

Worker #2

... Worker #N

Dask scheduler

DEISA Metadata adapter

Analytics client

Can be large, some might not even be used

Contention on the scheduler

1. data send

2. metadata send

3. metadata fetch

4. tasks execution

3. task-graph submission

1. data send

2. metadata send

3. metadata fetch

4. tasks execution

DEISA Bridge

DEISA Bridge

DEISA Bridge

Worker #1

Worker #2

... Worker #N

Dask scheduler

DEISA Metadata adapter

Analytics client

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used

Contention on the scheduler

loop must match the simu

Can be large, some might not even be used
Introducing now: Deisa3 with External tasks
What is an external task

- Represents data that will be produced later
  - By an external tool that is not under Dask scheduler control
- Introduce a new state in tasks state machine: *external*
- Support a new client call & scheduler RPC to create external tasks
  - The user chooses a UID (key)
  - Adding new parameters to Dask `Future.__init__` with default value for compatibility
  - Sets the task state to *external*
- Support a new client call & worker RPC to provide external tasks data
  - The user lists UIDs & provide data values
  - Adding new parameters to Dask `scatter`
  - Transition the task state from *external* to *memory*
Evaluation

Irene supercomputer skylake partition @ CEA TGCC, France

- 1,653 nodes, each 2 Intel Skylake CPUs × 24-cores @ 2.7 GHz, 180 GB memory
- 100Gb/s EDR Pruned fat tree InfiniBand network
- Lustre parallel file system (300GB/s)

Simple 2D Heat PDE solver mini-app

Principal Component Analysis based data analytics (Imported from a production code requirement)

- Unsupervised tool to reduce dimensionality
- Available in Scikit-learn & Dask-ML (based on SVD)
Incremental PCA

PCA needs all data in memory, single task

- Incremental PCA works on data minibatches
  - constant memory complexity
  - multiple tasks
- Dask-ML offers an IPCA
  - different API than PCA
- Implemented a new version of IPCA
  - API compatible with Dask-ML PCA

```python
from dask_ml.decomposition import InSituIncrementalPCA
from dask_interface import Deisa

# Initialize the Deisa
Deisa = Deisa(scheduler_info, config_file)
client = Deisa.get_client()

# Get data descriptor as a list of Deisa arrays object
arrays = Deisa.get_deisa_arrays()

# Filter data
gt = arrays['global_t'][...]
arrays.validate_contract()

ipca=InSituIncrementalPCA(n_components=2, copy=False,
  svd_solver='randomized')

ipca = ipca.fit(gt, ['t', 'X', 'Y'], ['X'], ['Y'])

# Submit the task graph to the scheduler
explained_variance, singular_values = client.persist([pca.explained_variance_, pca.singular_values_])

# ...
```
Weak Scalability

[Diagram showing weak scaling performance for 128 MB per process with bars for different processes (4, 16, 32, 64) and workers (2, 4, 8, 16, 32). The y-axis represents duration in seconds, and the x-axis represents processes or workers.]

- Simulation
- Post Hoc Write
- DEISA1 Communication
- DEISA3 Communication

- Post hoc IPCA
- Post hoc New IPCA
- DEISA1 IPCA
- DEISA3 New IPCA

(reading data + Analytics) x3
(waiting data + Analytics) x7 x1.8 x2.5 x3
Strong scalability (in hour.core)
Variability over iterations and processes

Communication time

- by MPI rank
- averaged over iterations
- std dev in red

High node allocation impact

- Pruned fat tree

DEISA1 has high noise

- Scheduler contention
- Wait for laggars
- Impact perf

DEISA1
lots of metadata

DEISA3
less metadata
Deisa3 based on external tasks

Metadata sent from simulation to dask ahead of time

● A single task-graph constructed encompassing all time-steps
  ○ Requires the addition of the “external tasks” concept to dask
● Time is a dimension like any other
  ○ More expressivity (e.g. one graph for time derivative)
● Reduced metadata transfer
  ○ Less contention on the scheduler
● Contracts
  ○ Detect data actually required by the graph, do not transfer useless data
  ○ Better performance
To conclude

A work to support in-transit HPC/ML workflows: MPI + Dask = Deisa

● Added external tasks in Dask
  ○ Makes Deisa3 possible: single-graph, contracts, less contention
  ○ A concept useful beyond Deisa!

● Implemented a new IPCA in Dask-ML

● Evaluated an in situ Heat2D / PCA workflow
  ○ Outperforms plain Dask by a high margin
  ○ Solves performance issues of Deisa1

Now working to

● Bring external tasks to Dask main branch & make them available to the world
● Move to a workflow with production simulation & more complex ML-analytics
Dask for post hoc analytics

File-system IO performance is an issue

```python
import dask.array as da
from dask_ml.decomposition import IncrementalPCA
import yaml, json
import h5py

# Connect to Dask
sched = json.load(open('sched.json'))
client = dask.distributed.Client(sched['address'])

# load the simulation configuration
simu = yaml.load(open('simulation.yml'))

# Build a lazy array descriptor from HDF5
gtemp = h5py.File('data.hdf5', 'r')['gtemp']
gtemp = da.from_array(gtemp, chunks=(1, 4096, 4096))

for step in range(0, simu['timesteps']):
    pca = IncrementalPCA(n_components=2, copy=True, svd_solver='randomized')
    pca.fit(gtemp[step,:,:])
    print(pca.explained_variance_)
```