Data Science with DASK

2nd MarDATA Block Course
“Advanced Scientific Programming”

November 19th, 2020

Martin Claus, Katharina Höflich, Willi Rath

MARDATA HELMHOLTZ SCHOOL FOR MARINE DATA SCIENCE

GEOMAR Helmholtz Centre for Ocean Research Kiel

Agenda for today

- Overview of Dask's fundamental concepts
- First hands-on session on Dask basics
- Review of the key takeaways
- Excursion: Dask distributed deployments / tools
- Second hands-on session on Dask for machine learning
- Question / Answer Session

Let's also schedule breaks?

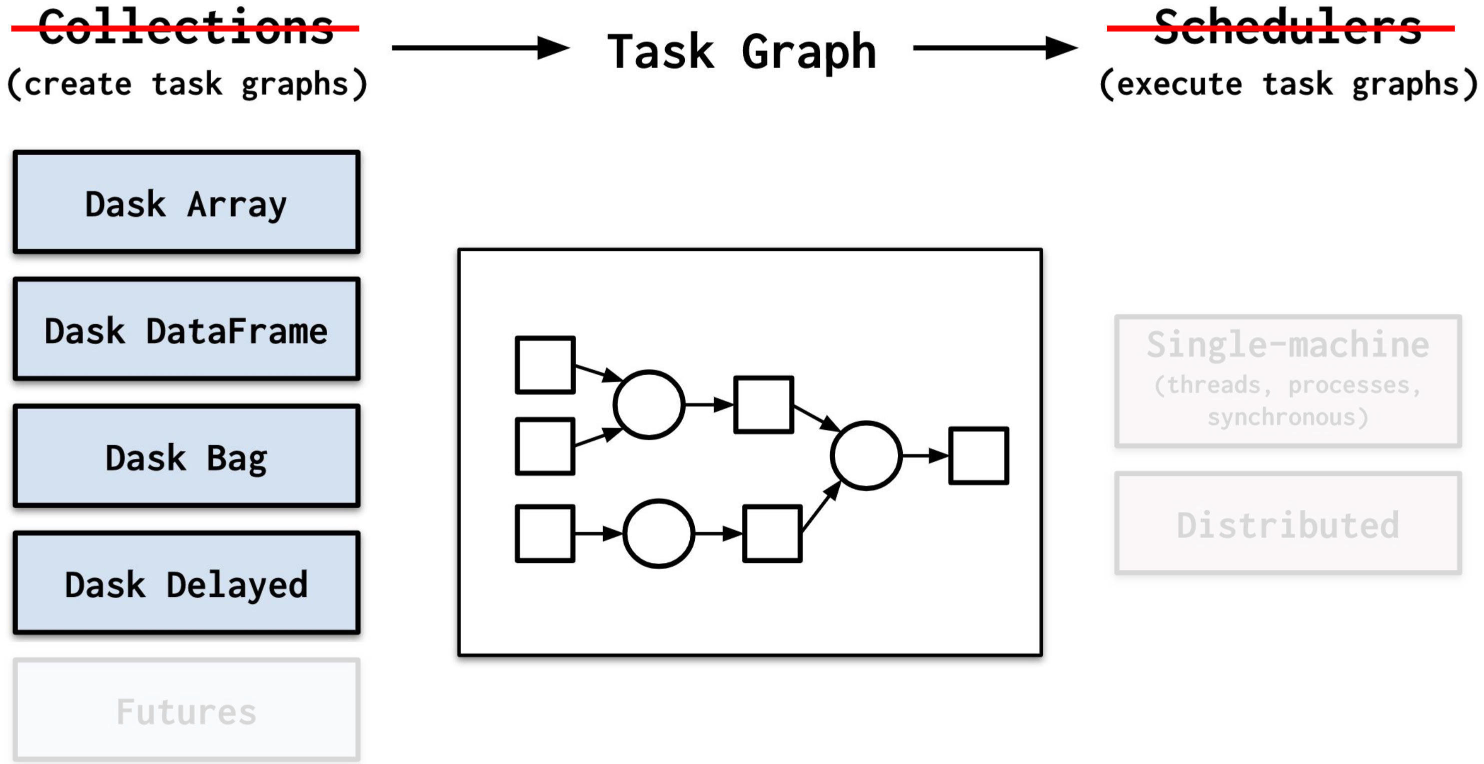
Aim of this course

We want you to understand the following:

- Dask is all about *task-graphs*.
- Dask provides various ways of *building task graphs*
  some of which can *replace existing toolboxes* like Numpy or Pandas.
- Dask provides various ways of executing task-graphs
  in *parallel* on a single machine or on *distributed clusters*.

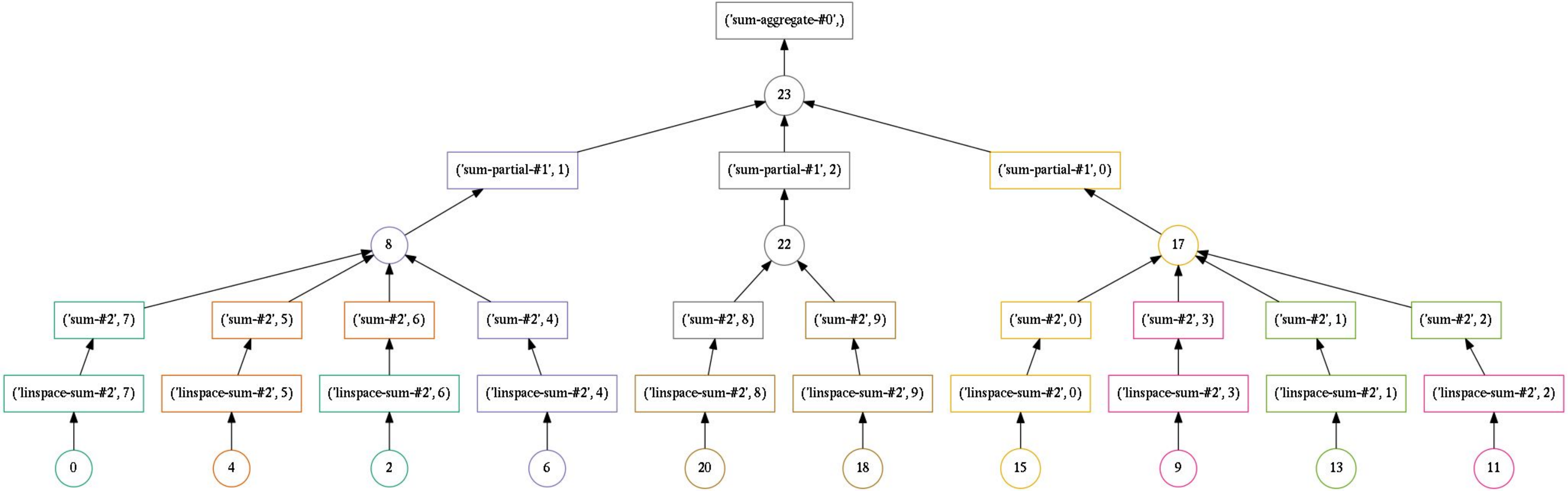
And: Know when *<u>NOT to use</u>* Dask!

Dask base concepts



from: https://docs.dask.org/en/latest/

Task graphs encode the flow of information



Your turn: *Can you find workflows / calculations you already know that can be described as a directed acyclic graph?*

Hands on

What is Dask *[30 minutes?]*

- Lecture-style overview of the core concept of Dask.
- Simple examples for explicitly building a task graph.

Creating task graphs *[60 minutes?]*

- Using Dask bag.
- Dask Dataframes.
- Dask Arrays.

Different ways of executing task graphs *[45 minutes?]*

- Get to know a simple application.
- Run the workflow on a single machine.
- Scale _out_ to multiple machines.

[Lunch ~ 12:30 - 13:00]

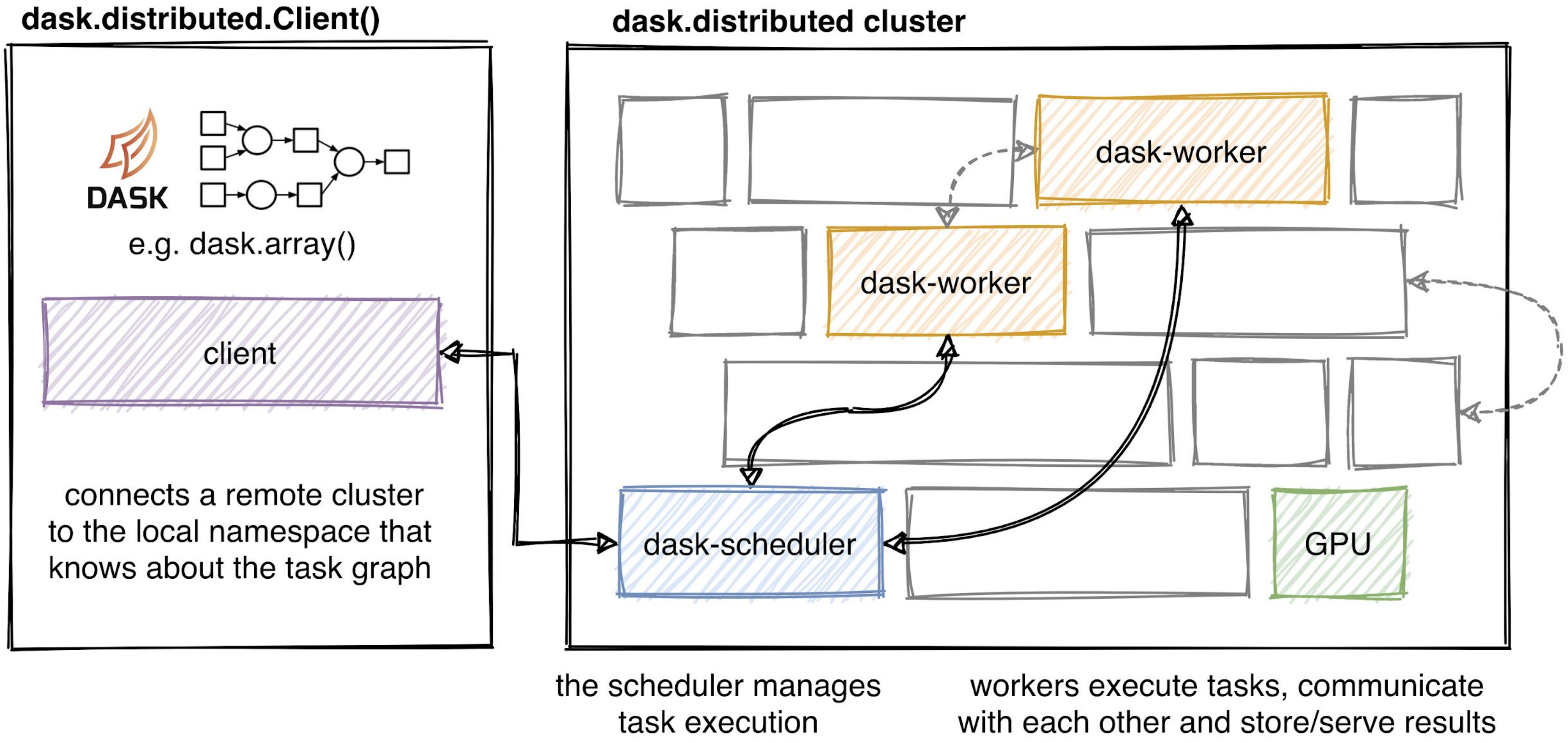
Parallelize machine learning with Dask *[45 minutes?]*

- Parallelize machine learning workloads.

→ https://github.com/mardatade/Course-Data-Science-with-Dask

Dask distributed scheduling

- For now, we have only worked with Dask single-machine schedulers.
- The necessary entities to go remote and to scale out are provided by the Dask distributed package (https://distributed.dask.org/en/latest/).
- Key strength of Dask: it is designed for the utilization of compute/memory resources of hundreds of separate machines in a common network.
- Let’s have a look now… and put together some machines manually!


dask.distributed.Client()
DASK
e.g. dask.array()
client
connects a remote cluster
to the local namespace that
knows about the task graph
dask.distributed cluster
dask-worker
dask-worker
dask-scheduler
GPU
the scheduler manages
task execution
workers execute tasks, communicate
with each other and store/serve results

Semi-Automatic Ways of Creating Dask Clusters

- For high-performance computers / other compute servers
    - Dask jobqueue: Manages clusters with a single-node granularity. Probably the most mature and most convenient of the HPC deployment solutions.
    - Dask MPI: Uses MPI to distribute scheduler and workers within an HPC job.
(but does not communicate via MPI!)
    - Dask SSH: Starts scheduler and workers on hosts you can access via SSH
    - Dask DRMAA: Uses DRMAA (a common interface to many different HPC scheduling softwares) for deploying Dask clusters.
- For cloud-computing platforms
    - Dask kubernetes: Creates and manages Dask clusters with Kubernetes
    - Dask cloudprovider: High-level tool that deploys clusters to (Amazon Web Services, Digital Ocean, Google Cloud Platform, Microsoft Azure).
    - Dask yarn: Deploys Dask clusters on Hadoop clusters.
    - Dask gateway: Provides Dask clusters via a web service / API.

… and counting

Summary: Key points to remember

We want you to understand the following:

- Dask is all about *task-graphs*.
- Dask provides various ways of *building task graphs*
  some of which can *replace existing toolboxes* like Numpy or Pandas.
- Dask provides various ways of executing task-graphs
  in *parallel* on a single machine or on *distributed clusters*.

And: Know when <u>*NOT to use*</u> Dask!

What's missing here?

- Dask as backend for other libraries
  - scikit-learn (partially covered here)
  - Xarray (labeled ndimensional arrays): http://xarray.pydata.org
  - Workflow management: https://prefect.io
- Actors
- Optimizing / tuning graphs
- Debugging
- Input / Output
- Resilience
- ...

Practical Problems

- Task graph should reflect <u>flow of information</u> between functions / methods rather than operations within calculations. Example high-order polynomial with a few coefficients is best wrapped into generalized ufuncs.
- Keep task graphs below a few 100.000 tasks, because scheduling takes time and resources.
- How to choose chunk sizes for Dask arrays?
- ...

Thank you!
What else do you want to know?