Python Dask

Softwareentwicklung in der Wissenschaft

Alena Pils

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

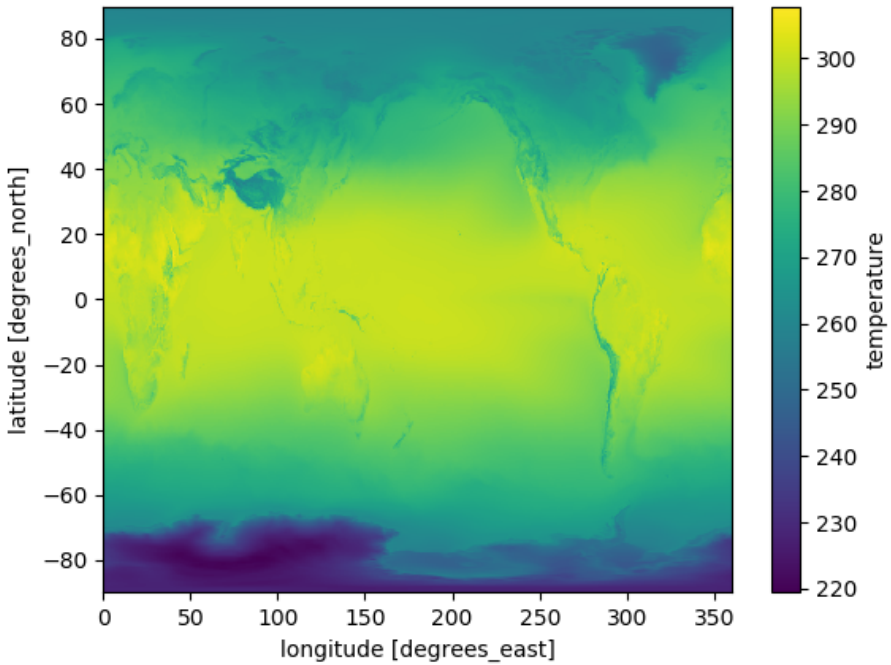
2020-06-08

Gliederung

Motivation

- Analysieren von Daten, die den Arbeitsspeicher übersteigen
- große Datenmengen führen in Python zu einem Memory Error

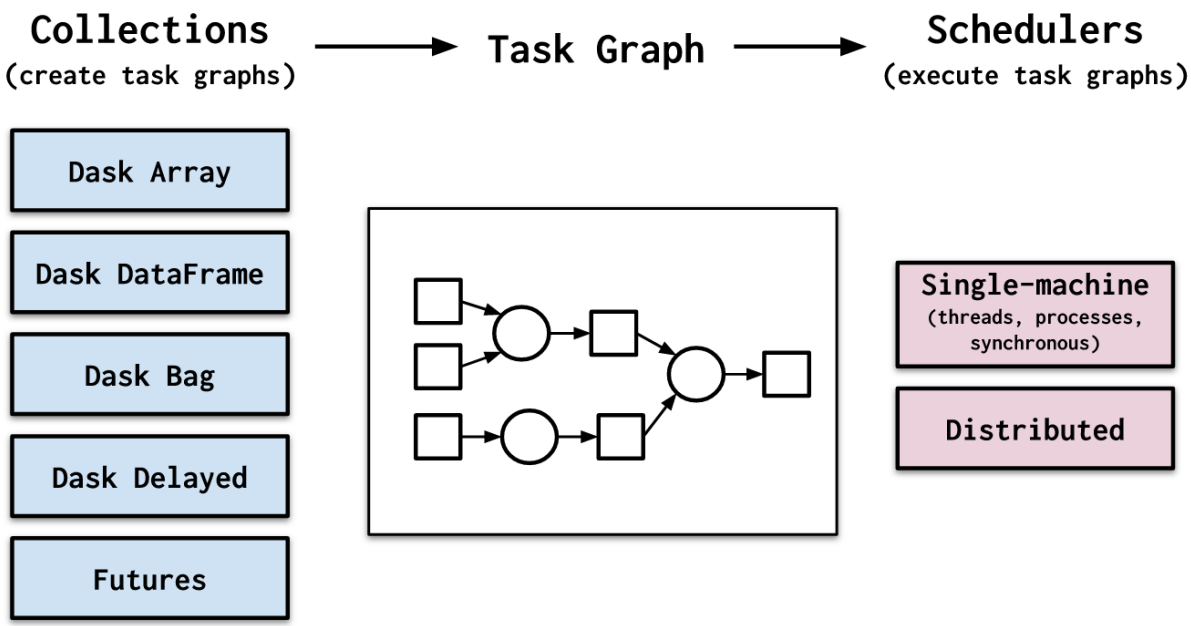
- Beispiel ERA5 (Klimadaten)[1]
  - stündliche Daten von 1979 bis heute
  - räumliche Auflösung von global 30 km



[1] https://www.ecmwf.int/en/forecasts/datasets/reanalysis-datasets/era5

Überblick[2]

- Bibliothek zur Parallelen Programmierung in Python
- sowohl auf einem Laptop als auch auf verteilten Cluster möglich
- sinnvoll, wenn die Datenmenge den verfügbaren Arbeitsspeicher übersteigt

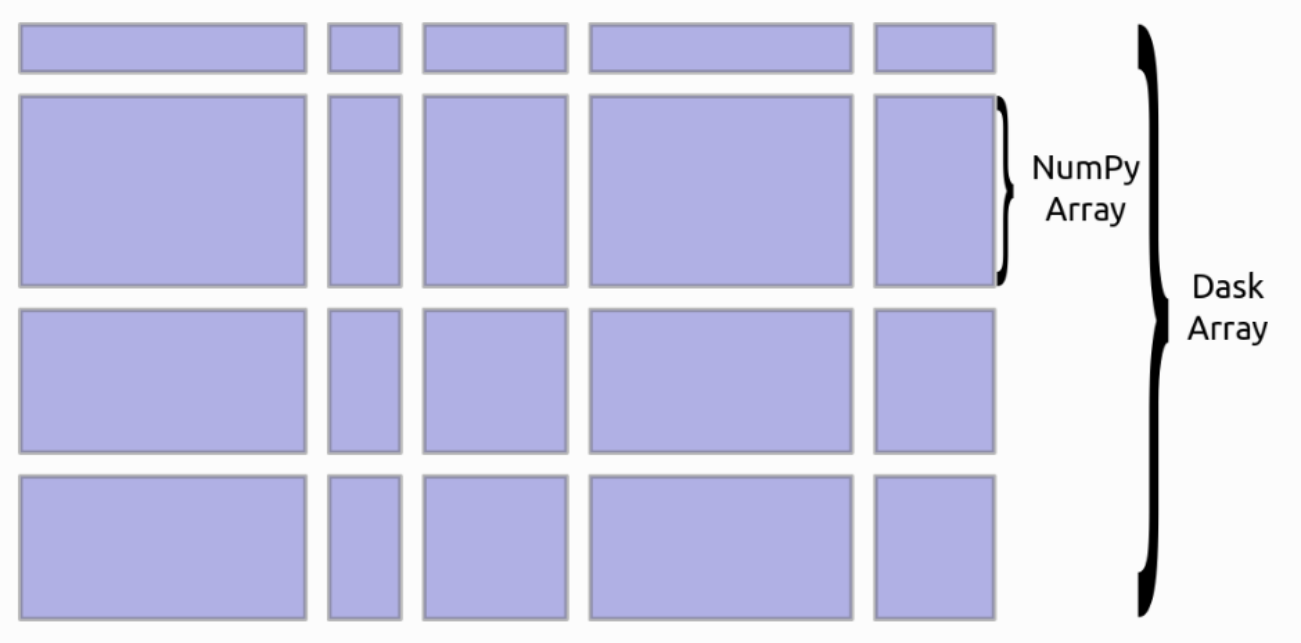


Quelle: `https://docs.dask.org/en/latest/`

[2] `https://www.youtube.com/watch?v=nnndxbr_Xq4`

Dask Array[3]

- ist n-dimensionalem NumPy Array ähnlich
- ermöglicht Arrays größer als der Arbeitsspeicher sowie parallele Berechnungen
- nutzt Aufteilung der Daten in Blöcke



Quelle: `https://docs.dask.org/en/latest/array.html`

[3] `https://tutorial.dask.org/03_array.html`

Importieren von numpy

```
import numpy as np
```

Erstellen eines Arrays

```
x = np.arange(10)
x
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

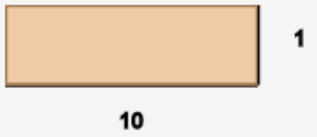
Importieren von dask.array

```
import dask.array as ds
```

Erstellen eines Arrays

```
x = ds.arange(10)
x
```

	Array	Chunk
Bytes	40 B	40 B
Shape	(10,)	(10,)
Count	1 Tasks	1 Chunks
Type	int32	numpy.ndarray

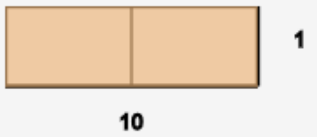



```
x.compute()
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
y = ds.arange(10,chunks=(5,))
y
```

	Array	Chunk
Bytes	40 B	20 B
Shape	(10,)	(5,)
Count	2 Tasks	2 Chunks
Type	int32	numpy.ndarray



Array

```
x
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Slicing

```
x[::2]
```

```
array([0, 2, 4, 6, 8])
```

Mittelwert bilden

```
x[::2].mean()
```

```
4.0
```

Array

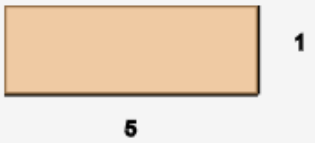
```
x.compute()
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Slicing

```
x[::2]
```

	Array	Chunk
Bytes	20 B	20 B
Shape	(5,)	(5,)
Count	2 Tasks	1 Chunks
Type	int32	numpy.ndarray




```
x[::2].compute()
```

```
array([0, 2, 4, 6, 8])
```

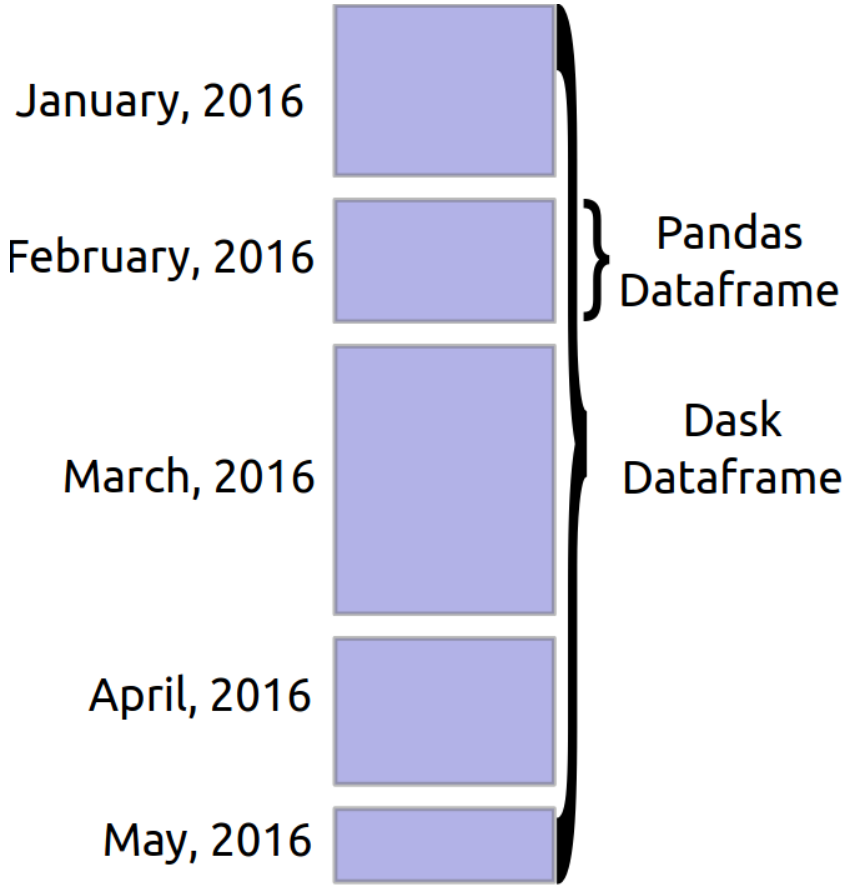
Mittelwert bilden

```
x[::2].mean().compute()
```

```
4.0
```

Dask DataFrame[4]

- ermöglicht DataFrames, die größer sind als der Arbeitsspeicher sowie parallele Berechnungen
- Dask DataFrame besteht aus vielen Pandas DataFrames
- Aufteilung eines Dask DataFrames in Pandas DataFrames geschieht entlang des Indexes



Quelle: `https://docs.dask.org/en/latest/dataframe.html`

[4] `https://tutorial.dask.org/04_dataframe.html`

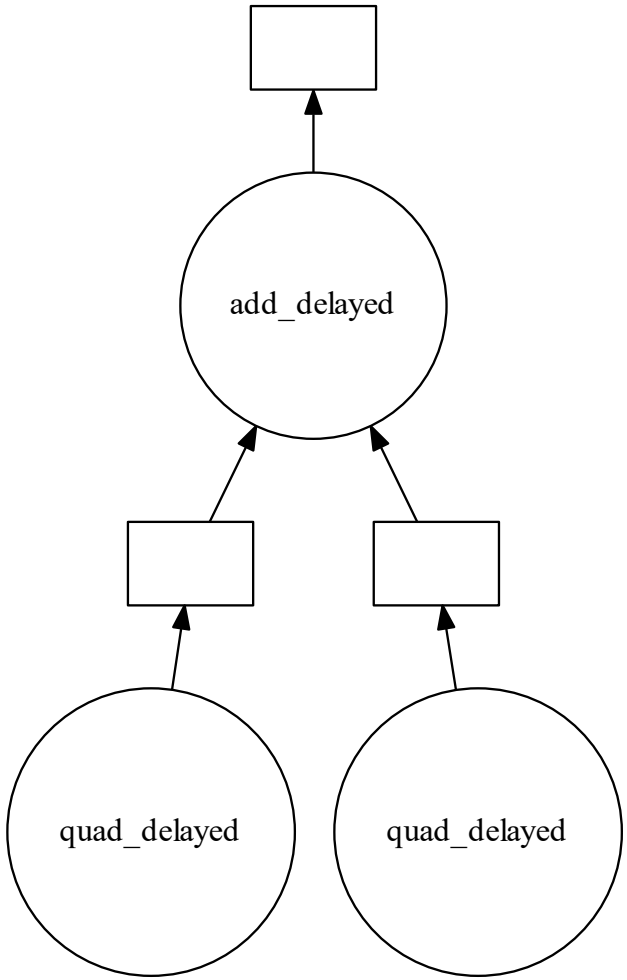
Dask DataFrame[6]

- Dask DataFrame besteht aus mehreren Partitions, die jeweils ein Pandas DataFrame sind
- Dask DataFrame deckt einen kleinen Anteil der Pandas API ab
- größter Unterschied zu Pandas: Erstellen eines Task Graphen anstelle einer unmittelbaren Berechnung
- keine Dask DataFrames mit mehreren Indices möglich[5]

[5] http://xarray.pydata.org/en/stable/dask.html
[6] https://tutorial.dask.org/04_dataframe.html

Dask Delayed[7]

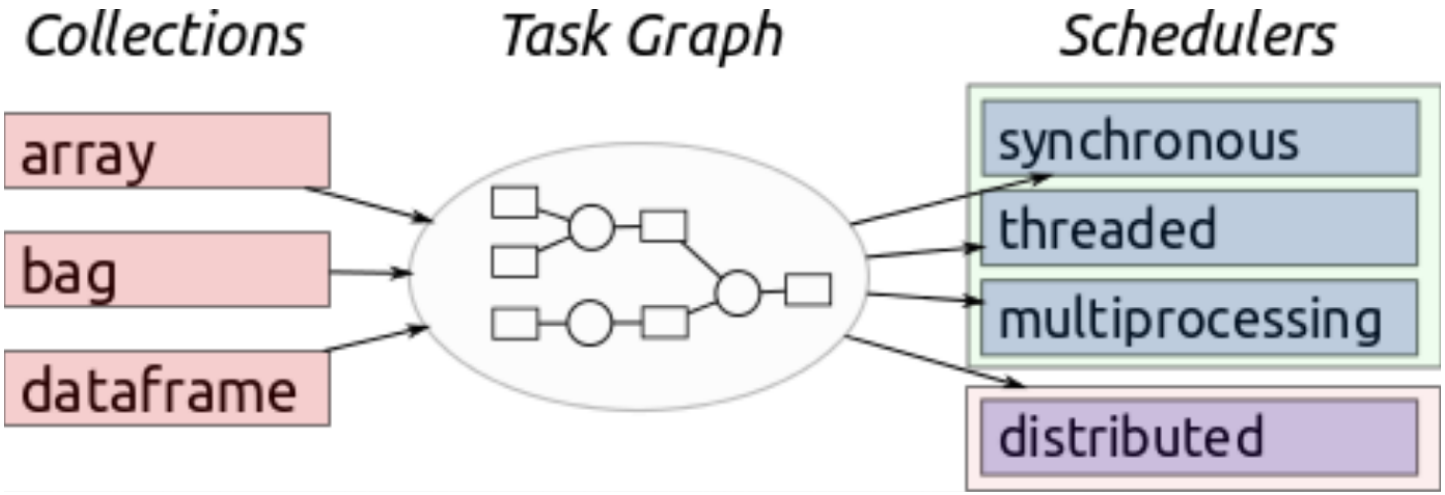
- Parallelisieren von Funktionen
- Parallelisierung möglich, wenn Funktionen unabhängig voneinander
- Delayed object: zeichnet aufzurufende Funktion und die an sie zu übergebenden Argumente auf
- Erstellen eines Delayed object mit:
  - @delayed über der Funktion oder
  - beim Funktionsaufruf: delayed(func)(arg)



[7] https://tutorial.dask.org/01_dask.delayed.html

Scheduler[8]

- Namen der Scheduler: “threaded“, “processes“, “single-threaded“, distributed
- Auswahl des Schedulers:
  - beim Funktionsaufruf: wert.compute(scheduler=“threaded“)
  - innerhalb eines Blocks: with dask.config.set(scheduler=“threaded“)
  - global: dask.config.set(scheduler=“threaded“)



Quelle: https://docs.dask.org/en/latest/scheduling.html

[8] https://tutorial.dask.org/05_distributed.html

Zusammenfassung

- Übersteigt die Datenmenge den Arbeitsspeicher ist es sinnvoll Dask zu nutzen
- Dask ist eine Bibliothek zur parallelen Programmierung in Python, die sowohl auf dem eigenen PC als auch auf einem Cluster funktioniert
- Dask Collections erzeugen Task Graphen, die dann von Schedulern ausgeführt werden
- große Überschneidung der API zwischen Dask und NumPy sowie Pandas

Informationen

Dokumentation und Links zu Tutorials:
`https://dask.org/`

Tutorial:
`https://github.com/dask/dask-tutorial`

Kurze Einführungsvideos:
`https://www.youtube.com/channel/UCj9eavqmvwaCyKhIlu2GaoA`

Beispiele aus den Geowissenschaften:
`http://pangeo.io/use_cases/`

Literatur I

[1] Inc. Anaconda and contributors Revision ab99d961, 2014-2018.
https://docs.dask.org/en/latest/.

[2] Inc. Anaconda and contributors Revision ab99d961, 2014-2018.
https://docs.dask.org/en/latest/array.html.

[3] Inc. Anaconda and contributors Revision ab99d961, 2014-2018.
https://docs.dask.org/en/latest/dataframe.html.

[4] Inc. Anaconda and contributors Revision ab99d961, 2014-2018.
https://docs.dask.org/en/latest/scheduling.html.

[5] Dask, 2019.
https://www.youtube.com/watch?v=nnndxbr_Xq4.

[6] Dask Developers, 2018.
https://tutorial.dask.org/03_array.html.

Literatur II

[7] Dask Developers, 2018.
https://tutorial.dask.org/04_dataframe.html.

[8] Dask Developers, 2018.
https://tutorial.dask.org/01_dask.delayed.html.

[9] Dask Developers, 2018.
https://tutorial.dask.org/05_distributed.html.

[10] European Centre for Medium-Range Weather Forecasts.
https://www.ecmwf.int/en/forecasts/datasets/reanalysis-datasets/era5.

[11] xarray Developers Revision, 2014-2020.
http://xarray.pydata.org/en/stable/dask.html.