Analyse mehrdimensionaler Arrays auf Hochleistungsrechnern

xarray und dask

Aaron Spring

Max-Planck-Institut für Meteorologie

2019-07-01

2019-06-20 Analyse mehrdimensionaler Arrays auf Hochleistungsrechnern

Anwedungsvortrag, Sales pitch, Show Demo

Ökosystem des wissenschaftlichen Rechnens mit Python

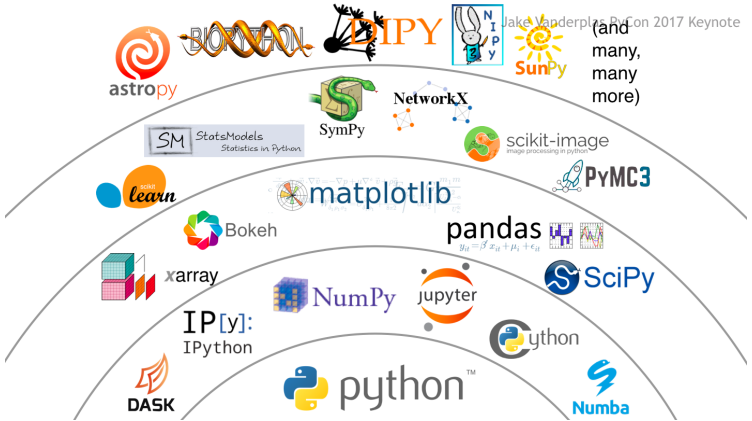
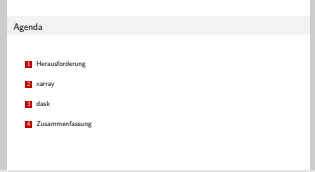


Abbildung: *Python Visualization Landscape*

Als Anfang: wie ist xarray eingebettet in diesen Kurs und PSE
xarray als alternative für pandas (Exceltabellen-like) basierend auf numpy
performanter mit numpy, cpython, dask
iteraktiv mit ipython, jupyter notebook
darstellbar mit matplotlib, seaborn, bokeh
erweiterbar mit sklearn, scipy, statsmodels
kein einzelnes Modul, eingebettet ins okosystem
übersicht, aber wofür braucht man das?

Agenda



Mehrdim. Daten zur verfuegung gestellt fuer BSc, MSc oder PhD-Arbeit
(ich: MPI-M)
Live-Demo wie mit xarray
was steckt konzeptionell hinter xarray
Anwendung xarray
was geht nicht, dafür dann dask
was steckt konzeptionell hinter dask
dask skalierung
tips

Herausforderung

- Analyse mehrdimensionaler Daten
  - Wettervorhersage, Satellitendaten, Klimamodeloutput
  - (Börsendaten)
- Aufbereiten der Daten zu einer schlüssigen Story
- $\hookrightarrow$ technisch möglichst einfach und intuitiv

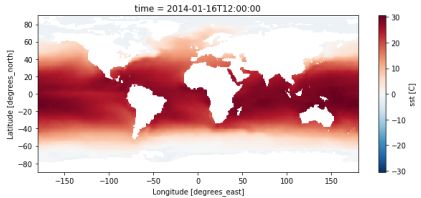
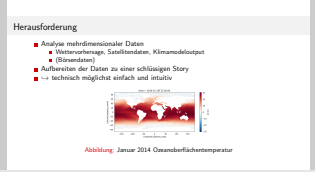


Abbildung: Januar 2014 Ozeanoberflächentemperatur

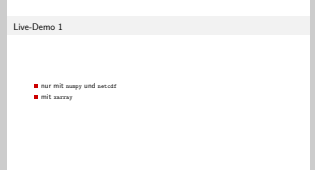


Plotten und IO sind keine wissenschaftlichen Ergebnisse. Erst das was dabei rumkommt, wenns gut gemacht ist.

hier heute klimadaten.

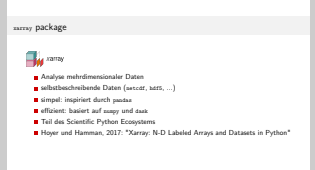
Live-Demo 1

- nur mit `numpy` und `netcdf`
- mit `xarray`



`xarray` package

- Analyse mehrdimensionaler Daten
- selbstbeschreibende Daten (`netcdf`, `hdf5`, ...)
- simpel: inspiriert durch `pandas`
- effizient: basiert auf `numpy` und `dask`
- Teil des Scientific Python Ecosystems
- Hoyer und Hamman, 2017: "Xarray: N-D Labeled Arrays and Datasets in Python"



Bottom line

`xarray` Datentypen

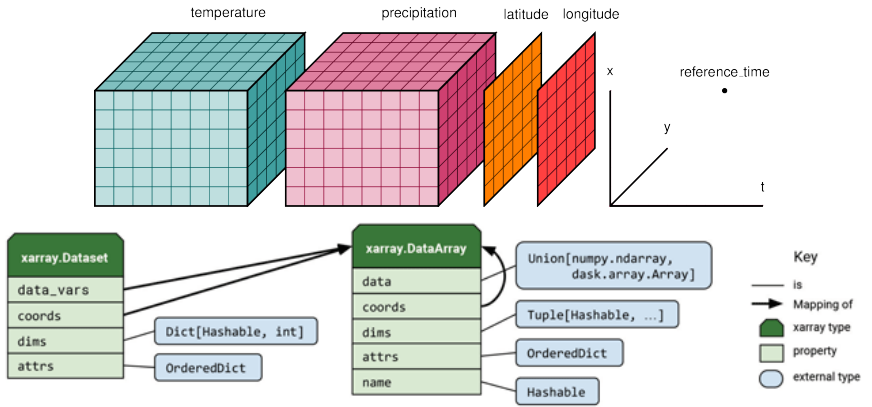
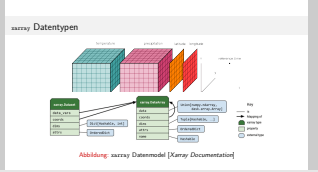


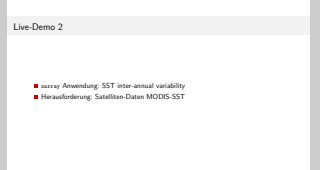
Abbildung: `xarray` Datenmodel [*Xarray Documentation*]



angepasst an netcdf Konventionen, wie Daten gespeichert sindDataArrays teilen sich Koordinaten

Live-Demo 2

- `xarray` Anwendung: SST inter-annual variability
- Herausforderung: Satelliten-Daten MODIS-SST



`dask` package

- Dynamischer Task Scheduler
- nutzt `multiprocessing`, `threading` und `concurrent`
- Chunking von "Big Data"für Parallelisierung
- intuitiv: bekannte API
- skaliert: vom Laptop zum Supercomputer
- Teil des Scientific Python Ecosystems
- Rocklin, 2015: "Dask: Parallel Computation with Blocked Algorithms and Task Scheduling"

Bottom line

`dask` Datenmodel

- `dask.array` → `numpy.ndarray`
- `dask.bag` → `iterable`
- `dask.dataframe` → `pandas.DataFrame`

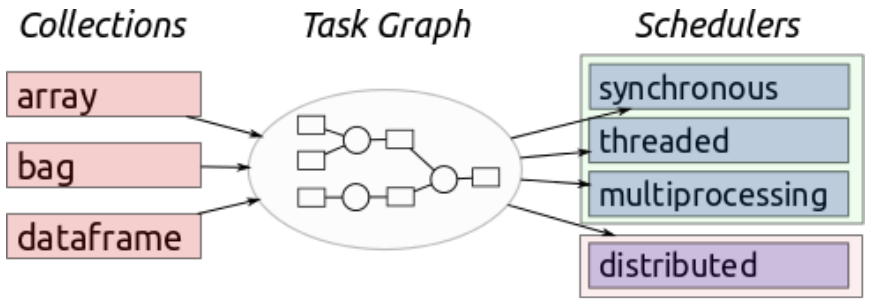
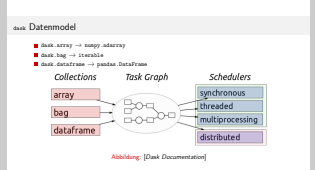


Abbildung: [*Dask Documentation*]



dask baut auf bekanntem auf und skaliert

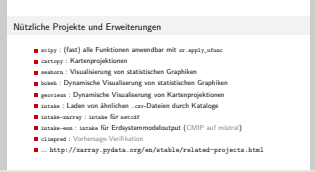
Live-Demo 3: dask

- chunking of lazy data
- dask task graphs
- (optional) Benchmark
- SST inter-annual variability
- dask.distributed: MODIS-SST inter-annual variability

Live-Demo 3: dask
- chunking of lazy data
- dask task graphs
- (optional) Benchmark
- SST inter-annual variability
- dask.distributed: MODIS-SST inter-annual variability

Nützliche Projekte und Erweiterungen

- `scipy` : (fast) alle Funktionen anwendbar mit `xr.apply_ufunc`
- `cartopy` : Kartenprojektionen
- `seaborn` : Visualisierung von statistischen Graphiken
- `bokeh` : Dynamische Visualiserung von statistischen Graphiken
- `geoviews` : Dynamische Visualisierung von Kartenprojektionen
- `intake` : Laden von ähnlichen `.csv`-Dateien durch Kataloge
- `intake-xarray` : `intake` für `netcdf`
- `intake-esm` : `intake` für Erdsystemmodeloutput (CMIP auf mistral)
- `climpred` : Vorhersage-Verifikation
- ... `http://xarray.pydata.org/en/stable/related-projects.html`



In demos gesehen, xarray allein ok, verbunden mit anderen packages top fast alles was NCL kann und mehr. Projekte auschecken und Beispiele angucken.

Kurze Einarbeitung für Pythonkenner, dann massiver Zeitgewinn.

Datentyp-Kompatibilität

- `ds.to_dataframe()` : `xarray` → `pandas`
- `ds.from_dataframe(df)` : `pandas.df` → `xarray`
- `ds['var].values` : `xarray` → `numpy.ndarray`
- `ds.to_netcdf()` : `xarray` → `netcdf`
- `ds.to_zarr()` : `xarray` → `zarr` (Cloudspeicherformat)
- `intake.cat.item.to_dask()` : Katalogisierte `netcdf` → `xarray.dask`
- `cdo.operator(input=ifile, returnXDataset=True)` : `cdo-py` Output → `xarray.dataset`
- ... `http://xarray.pydata.org/en/stable/api.html#io-conversion`

Datentyp-Kompatibilität
ds.to_dataframe() : xarray → pandas
ds.from_dataframe(df) : pandas.df → xarray
ds['var].values : xarray → numpy.ndarray
ds.to_netcdf() : xarray → netcdf
ds.to_zarr() : xarray → zarr (Cloudspeicherformat)
intake.cat.item.to_dask() : Katalogisierte netcdf → xarray.dask
cdo.operator(input=ifile, returnXDataset=True) : cdo-py Output → xarray.dataset
... http://xarray.pydata.org/en/stable/api.html#io-conversion

Das meiste funktioniert easy.
Deutlich einfacher als mit NCL.
Einfacher Dimensionen und Achsen zu Ändern als mit CDO.

Zusammenfassung

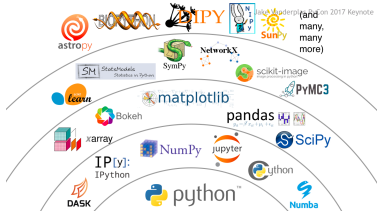


Abbildung: *Python Visualization Landscape*

- Read the fucking manual (RTFM).
- Ökosystem ausnutzen.
- High-level nutzer-freundlicher als low-level.
- Chunking bei Big Data.
- Parallelisierung ist nicht automatisch schneller.

Zusammenfassung
Abbildung: *Python Visualization Landscape*
- Read the fucking manual (RTFM).
- Ökosystem ausnutzen.
- High-level nutzer-freundlicher als low-level.
- Chunking bei Big Data.
- Parallelisierung ist nicht automatisch schneller.

Handlungsempfehlungeneingewoehnungszeit brauchts aber immer.

Literatur

Dask Documentation. URL: https://docs.dask.org/en/latest/ (besucht am 04. 06. 2019).

Hoyer, Stephan und Joe Hamman (2017). "Xarray: N-D Labeled Arrays and Datasets in Python". In: *Journal of Open Research Software* 5.1. DOI: 10/gdqdmw.

Python Visualization Landscape. *Jake VanderPlas The Python Visualization Landscape PyCon 2017*. URL: https://www.youtube.com/watch?v=FytuB8nFHPQ (besucht am 04. 06. 2019).

Rocklin, Matthew (2015). "Dask: Parallel Computation with Blocked Algorithms and Task Scheduling". In: Python in Science Conference. Austin, Texas, S. 126–132. DOI: 10/gfz6s5.

Xarray Documentation. URL: http://xarray.pydata.org/en/stable/index.html (besucht am 04. 06. 2019).