How the openEO project unifies access to big Earth Observation data processing platforms

Christoph Friedrich, Matthias Mohr, Edzer Pebesma

The idea of openEO

Develop an open API that connects various clients to big EO cloud backends in a simple and unified way

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant No 776242.

TU WIEN TECHNISCHE UNIVERSITÄT WIEN

WESTFÄLISCHE WILHELMS-UNIVERSITÄT MÜNSTER

WAGENINGEN UNIVERSITY & RESEARCH

vito

eodc

mundialis

SINERGISE

eurac research

Joint Research Centre JRC

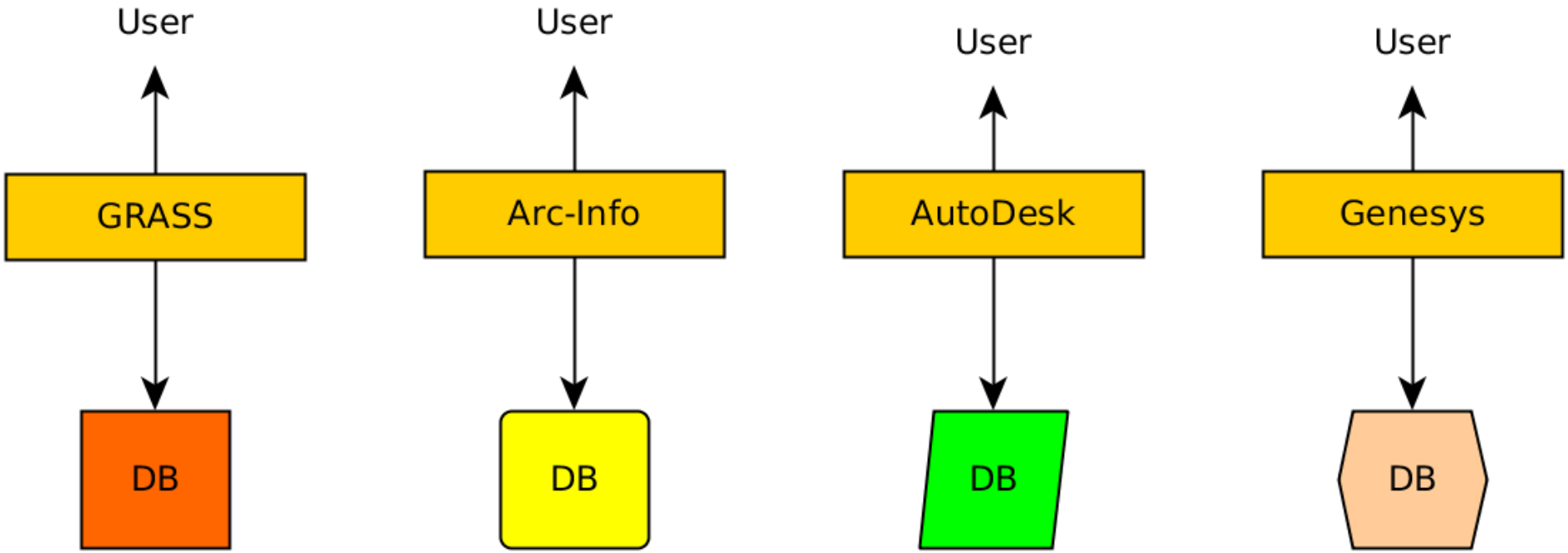
SOLENIX

Why?

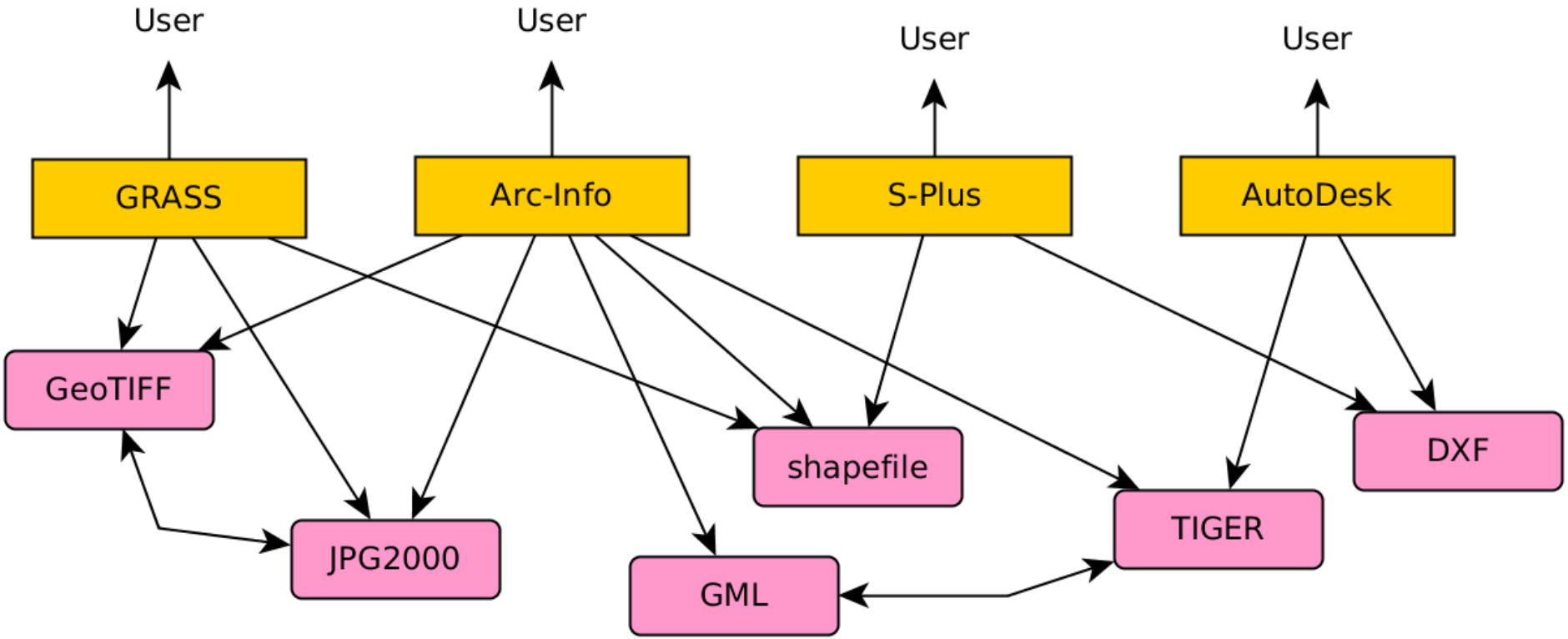
- EO data now **too large to download/handle**
  → EO processing **increasingly cloud-based**

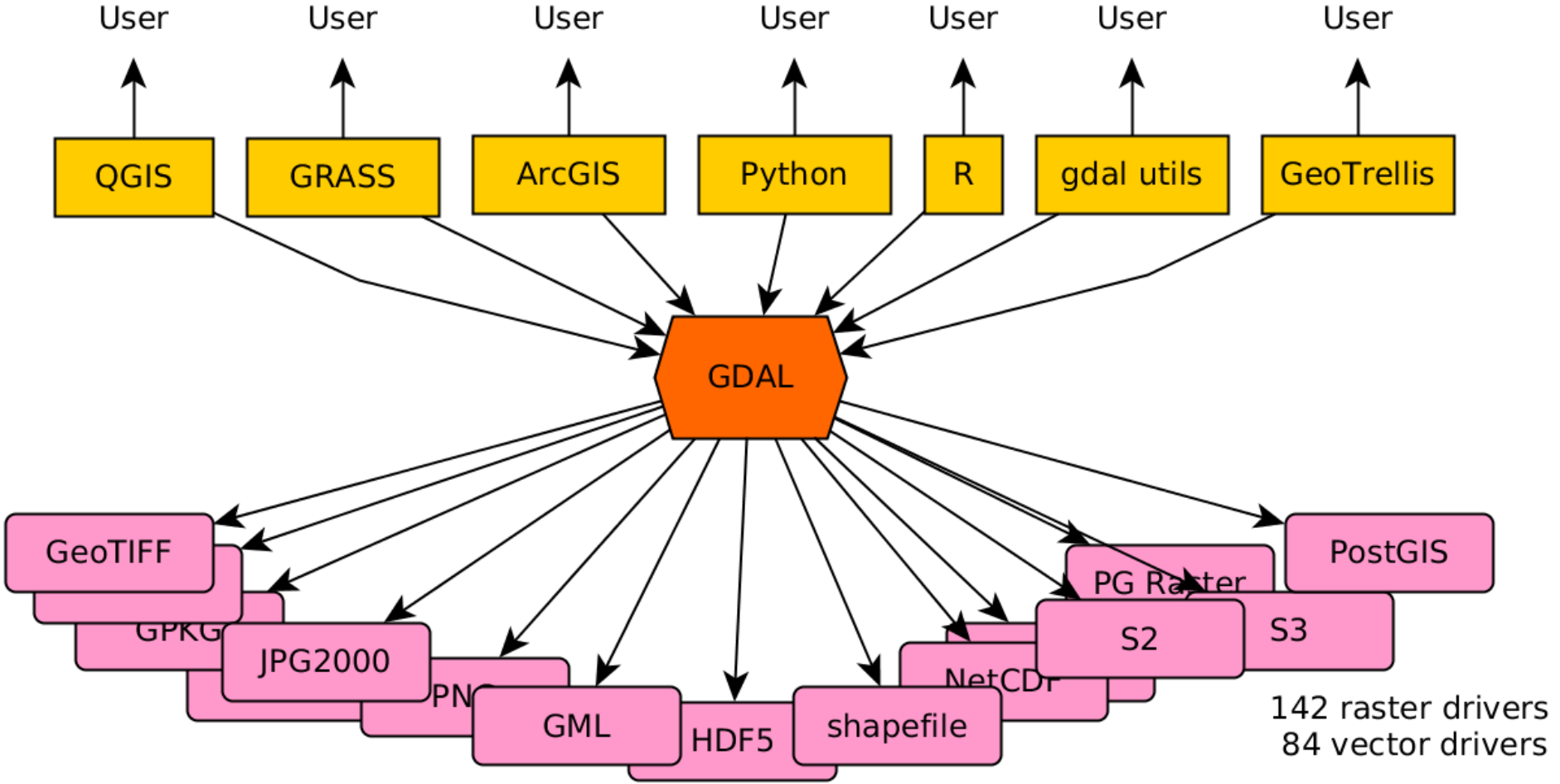
- Cloud-based EO **solutions pop up like mushrooms**
  - DIASs, TEPs, Google Earth Engine, Sentinel Hub, GeoTrellis, Rasdaman, …

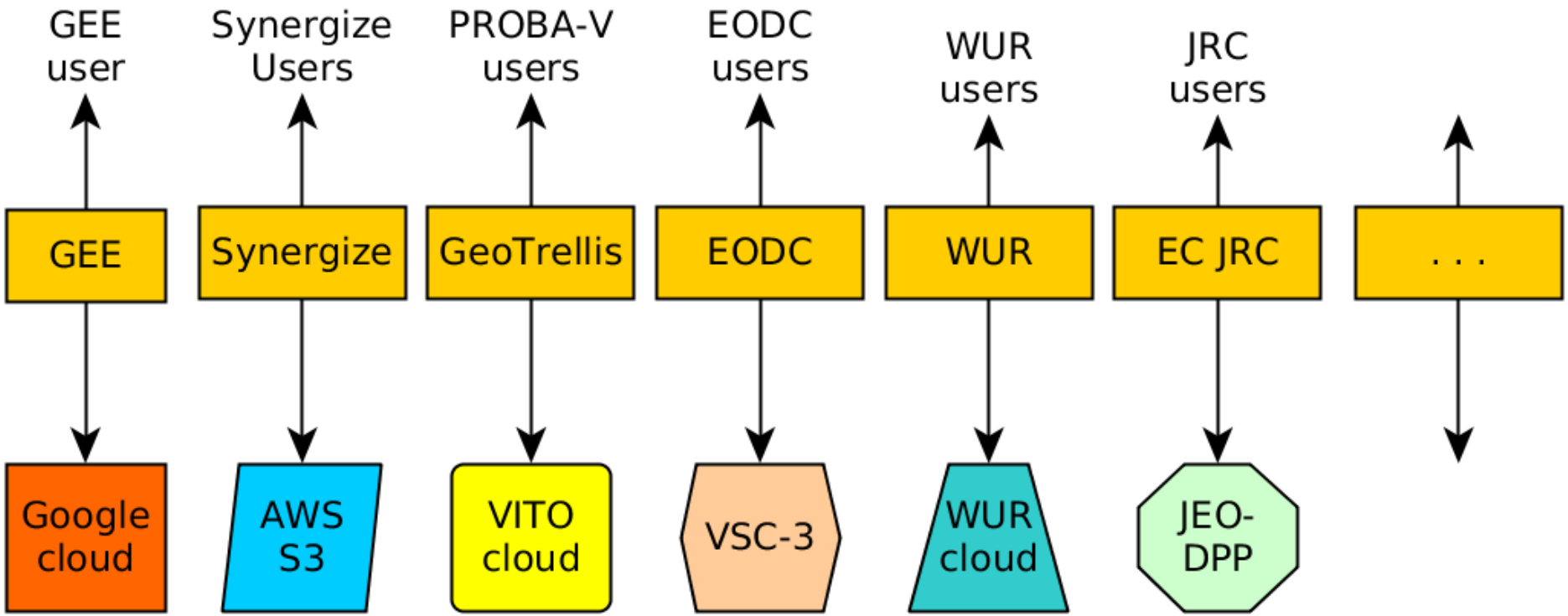
Who knows GDAL?

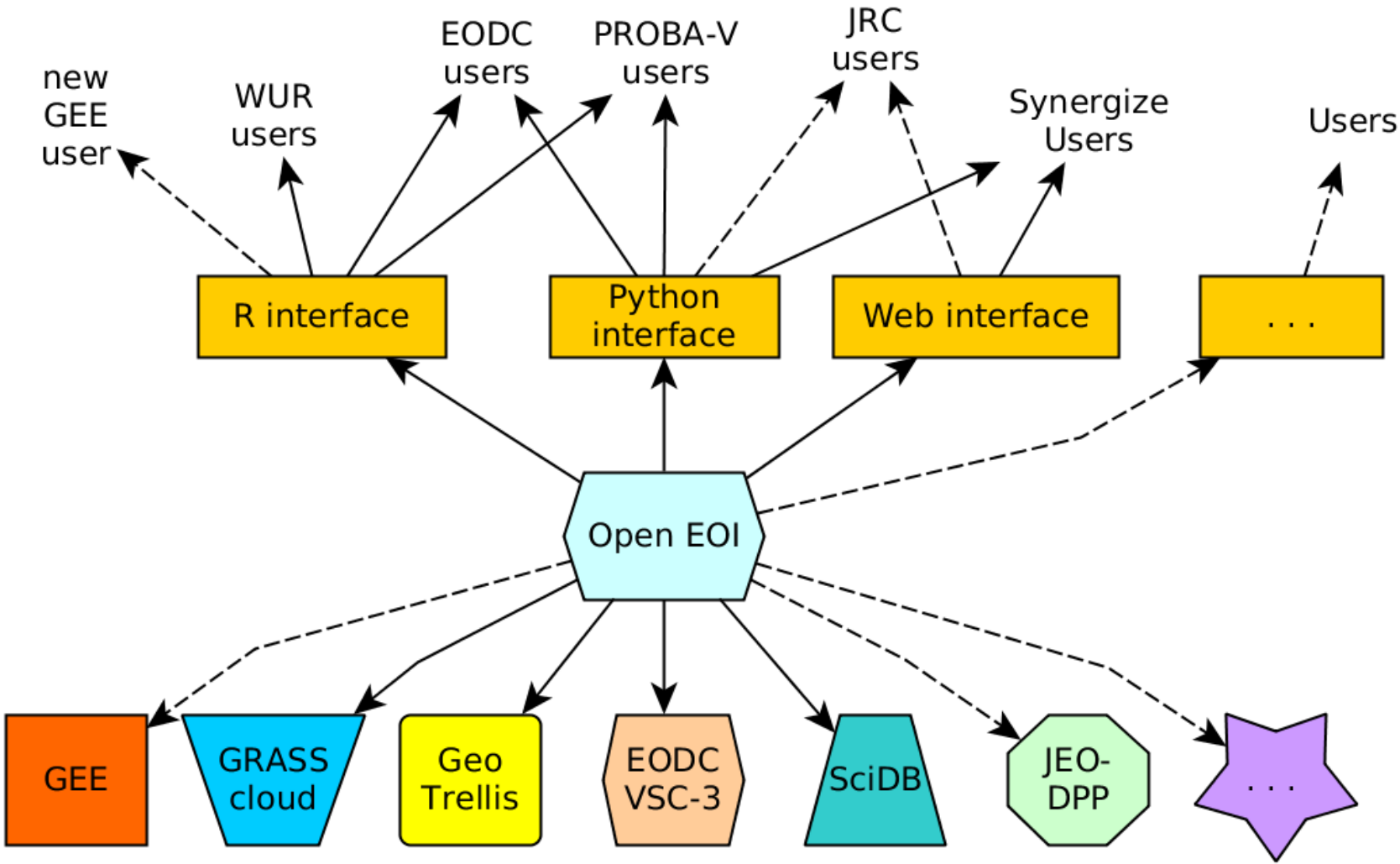


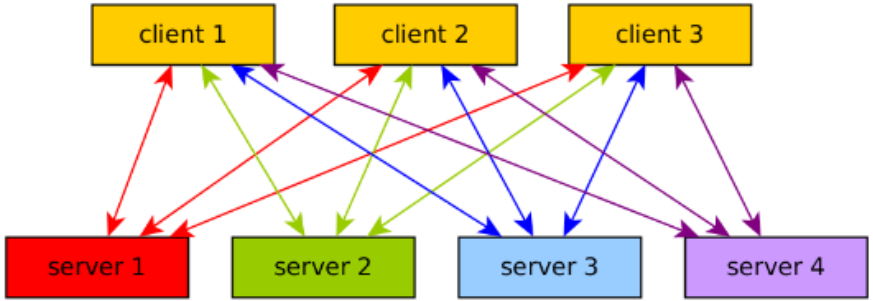
graphics from http://r-spatial.org/2016/11/29/openeo.html

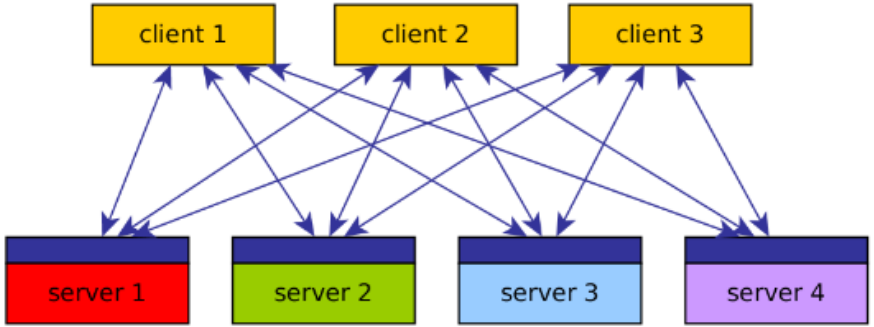

User
User
User
User
GRASS
Arc-Info
S-Plus
AutoDesk
GeoTIFF
shapefile
DXF
JPG2000
TIGER
GML


User
User
User
User
User
User
User
QGIS
GRASS
ArcGIS
Python
R
gdal utils
GeoTrellis
GDAL
GeoTIFF
GPKG
JPG2000
PN
GML
HDF5
shapefile
NetCDF
PG Raster
S2
S3
PostGIS
142 raster drivers
84 vector drivers


GEE user
Synergize Users
PROBA-V users
EODC users
WUR users
JRC users
GEE
Synergize
GeoTrellis
EODC
WUR
EC JRC
. . .
Google cloud
AWS S3
VITO cloud
VSC-3
WUR cloud
JEO-DPP


new GEE user
WUR users
EODC users
PROBA-V users
JRC users
Synergize Users
Users
R interface
Python interface
Web interface
. . .
Open EOI
GEE
GRASS cloud
Geo Trellis
EODC VSC-3
SciDB
JEO-DPP
. . .





graphic from openEO deliverable 5

Why?

- EO data now **too large to download/handle**

  → EO processing **increasingly cloud-based**
- Cloud-based EO solutions **pop up like mushrooms**
  - DIASs, TEPs, Google Earth Engine, Sentinel Hub, GeoTrellis, Rasdaman, …
- **Combine** different backends
- **Extensibility**
- Compare/validate results → **reproducibility**
- So far: Google Earth Engine the only feasible offering?
  - It’s easy and ready -- **but not open.**

What we do

- Define a **RESTful API**
- Implement **reference implementations**
  - 7 backends
  - 3 client libraries
- Define a **process catalogue**
  - extendable by user-defined functions
- Everything's **open source**: https://github.com/Open-EO/

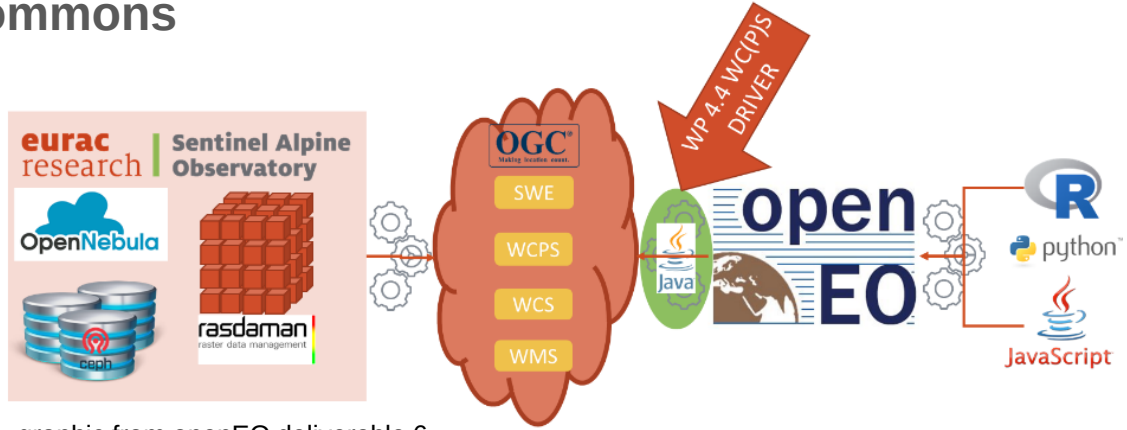
Live Demo

http://hub.openeo.org/

Standards

- Used existing standards **where possible**
  - REST, OpenAPI, GeoJSON, OpenID Connect, EPSG codes…
- Processing: **not WPS** (doesn't support chaining) -- but driver for WCPS!
- Results: Exposing **web services** possible (WMS, WMTS, WCS, XYZ, ...)
- Compliant to **OGC API Commons**
- Working on **STAC**

  (data discovery)



graphic from openEO deliverable 6

What we had

- 3 clients
- 7 backends
- 3 use cases
- a handful of processes

Progress over the last year

- **New process graph structure**
  - Parallelised processing
  - Callbacks
  - Multiple results
  - Process graph variables
- **Adaption** of client libraries and backend drivers
- Compatibility to **OGC API - Commons**
- Full process catalogue with **100+ processes defined**
  - Backends can support arbitrary subset
  - Extendable by own definitions
- **UDF reference implementations** in R and Python

My lessons learned

- Programmers don’t read the docs
- Choosing and detailedly describing processes takes time
- Standardising the algorithms is feasible
- But it’s quite hard to standardise the data (“Analysis Ready Data”)
  - Non-uniform naming (e.g. “SENTINEL-2” vs. “COPERNICUS/S2” etc.)
  - Scientific vs. easy-to-use

Still to come

- Implementation of processes in backends
- Full compatibility to newest API version
- Fully tested, stable, 1.0-release-ready versions of everything
- ...

Challenges (content)

- Uniform **naming** of data collections
- Incorporating **everything into one API**
  - Dropped the idea of including everything, e.g. user management, settings, payments, ...
- **User-defined functions**
  - reference implementations in Python and R
- **Efficient access** to big data
  - data cube (raster and vector)
- **Validating** backends against each other (**reproducibility**)
  - Master thesis on comparing output
- **Cost estimates**
  - Billing model, but giving a “What would this cost?” quote seems hard

Challenges (project)

- Ensure **user adoption** (clients *and* servers)
  - Conferences, workshops, hackathons, social media, science slam ...
- Extend openEO to **more backends**
  - There are many more than the ones we address -- we can’t solve this alone
- Ensure **project continuation**
  - Big players on board (e.g. Google Earth Engine, Sinergise SentinelHub)
  - Consortium includes companies that will use openEO in production → interest to maintain it
  - Community project

Thanks for your attention!

Any questions?

https://openeo.org/ https://**github**.com/Open-EO https://**twitter**.com/open_EO