Pangeo community platform and its use at CNES

BiDS 2019, Guillaume Eynard-Bontemps, CNES/Pangeo
Problems
- Data volume crisis in (geo)sciences
- Software multiplication, non reproducibility
- Many copies of the same datasets
- Local vs HPC vs Cloud
- Technology gap: industry vs academia

Mission
To cultivate an ecosystem in which the next generation of open-source analysis tools for the geosciences can be developed, distributed, and sustained.

Goals/ vision
- Foster collaboration around the open source Scientific Python ecosystem:
  - open and collaborative development
  - Welcoming and inclusive culture
- Support the development with domain-specific (geo)science and transverse packages
- Improve scalability of these tools to handle gigabytes to petabyte scale datasets
• **Set of tools that will facilitate science at all scales**
• **Platform agnostic**
• The core of the Pangeo ecosystem includes:
  • **Xarray** (data-model and toolkit for working with N-dimensional labeled arrays)
  • **Dask** (parallel computing)
  • **Jupyter** (interactive computing)
• Extensible: Series of 3rd party packages that build on top of core libraries
• Flexible: Individual components may be swapped in/out

**Examples of 3rd party packages in the Pangeo Ecosystem:**
• Data discovery
• Regridding and GIS
• Vector calculus
• Signal processing
• Thermodynamics
**Pangeo diversity**

## Build Your Own Pangeo

<table>
<thead>
<tr>
<th>Storage Formats</th>
<th>HDF</th>
<th>OPeNDAP</th>
<th>Cloud Optimized COG/Zarr/Parquet/etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ND-Arrays</td>
<td>NumPy</td>
<td>DASK</td>
<td>More coming...</td>
</tr>
<tr>
<td>Data Models</td>
<td>xarray</td>
<td>Iris</td>
<td>pandas</td>
</tr>
<tr>
<td>Processing Mode</td>
<td>Jupyter</td>
<td>Interactive</td>
<td>Batch</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Serverless</td>
</tr>
<tr>
<td>Compute Platform</td>
<td>HPC</td>
<td>Cloud</td>
<td>Local</td>
</tr>
</tbody>
</table>
Pangeo vs state of the art

Spark vs DASK

Mature
Robust
JVM/Python
Query optimized
Collections & Dataframes
Python overhead
For big tabular data
Hadoop/Cloud/HPC

Less Mature
Pretty strong
Python only
Science optimized
Collections, DF, Arrays, Futures…
Python only
For science data
Hadoop/Cloud/HPC

Laptop to cluster
Serverless
NetCDF/TIFF no ingestion
Scales with Dask
Python only
Can build array db with Pangeo (Open data Cube)

I don’t know much about Array DBs!!!
Pangeo public cloud deployment

hub.pangeo.io
pangeo.binder.io

JupyterHub/BinderHub running on the Google Cloud

- Kubernetes for both Jupyter and Dask-distributed
  - Dask-kubernetes
- Exploring/evaluating:
  - Cloud storage
  - User environment customization
  - Data discovery
- Kubernetes Helm-chart (github.com/pangeo-data/helm-chart)
- CI/CD with Hubploy and CircleCI
- Deployments exist on AWS and Azure.
HPC (HAL)
- 300 Tflops
- 380 batch servers / 8400 cores
- 4 interactive servers pre/post processing w/ GPU
- 6,2 PB GPFS / 200 TB burst buffer / 50 Gbps bandwidth
- Low latency network
- GPGPU Nvidia Volta V100

HPC DRSF (Ktulu)
- 20 Tflops
- 2 interactive servers pre/post processing w/ GPU
- 24 servers / 576 cores
- 120TB GPFS
- Low latency network
HPC use cases in CNES

Two main kinds of processing

Numerical simulation (HPC)
- Upstream phase, R&D
- Highly optimized technics
- Fine grain parallelism
Trends: multiscale, multiphysics

Data Processing (HTC)
- Downstream phase, operation
- Sensors data → scientific data
- Coarse grain parallelism
Trends: data volume explosion
• JupyterHub and notebooks for interactive computing
  • Hub on a VM with qsub access
  • Batchspawner, Wrapspawner
• dask.distributed: parallel workers across many HPC nodes
• Xarray for computational toolkit and I/O
• New tool for deploying dask clusters on HPC: dask-jobqueue
  • Start a cluster from a notebook
  • Interactive (or not) distributed computing
  • Auto scaling capabilities
Demo: Dask and dask-jobqueue basic example

```python
from dask_jobqueue import PBSCluster
cluster = PBSCluster(cores=4, memory="15GB", local_directory='TMPDIR',
queue='qdev', project='DaskTest',
walltime='01:00:00', interface='ib0')
cluster.adapt(minimum=2, maximum=16)

from dask.distributed import Client
client = Client(cluster)

def my_costly_simulation(args):
    ## Do something
    return sum(args)

input_params = [...] 

futures = client.map(my_costly_simulation, input_params)
results = client.gather(futures)
```

Takeways:

- No use of shared storage for syncing results → less bad IOs.
- Simplified high level orchestration, all with Python and Dask

---

404: Not Found
You are requesting a page that does not exist!
Demo: Multi temporal NDVI with Xarray

Takeways:

- Interactive processing over temporal series with Jupyter/Dask/Xarray
- Interactive visualization
- Recompute upon user Action
- Statistical analysis
- JPEG2000 is not a good interactive processing format
- Share data in cloud/distributed processing ready format
Conclusions

- Pangeo ecosystem greatly facilitates distributed computing and data analysis at scale
- It changes ways of doing it too
- Non monolithic platform built on top of existing Scientific Python stack and new related packages
- Community is always here to help
- Dask more versatile and easy to use than Spark.

Next steps

- Broaden users and use cases at CNES
- Encourage people to get in touch with Pangeo community
- Work in cooperation with others (Ongoing with Ifremer and CLS on SWOTaVal data processing)

Pangeo website and discussions:

- [https://pangeo.io](https://pangeo.io)
- [https://github.com/pangeo-data/pangeo/issues](https://github.com/pangeo-data/pangeo/issues)
- [https://medium.com/pangeo](https://medium.com/pangeo)
- Pangeo Example + Binder:
  - [https://github.com/pangeo-data/pangeo-example-notebooks](https://github.com/pangeo-data/pangeo-example-notebooks)
- Dask jobqueue:
  - [https://github.com/dask/dask-jobqueue](https://github.com/dask/dask-jobqueue)
- Dask simple examples:
  - [https://github.com/dask/dask-examples](https://github.com/dask/dask-examples)

My email

Guillaume.Eynard-Bontemps@cnes.fr