Scalable and Distributed Metadata Extraction with Xtract

Tyler J. Skluzacek *(and Kyle, Ian, Ryan, Zhuozhao, Yadu, Logan, Ben B., …)*

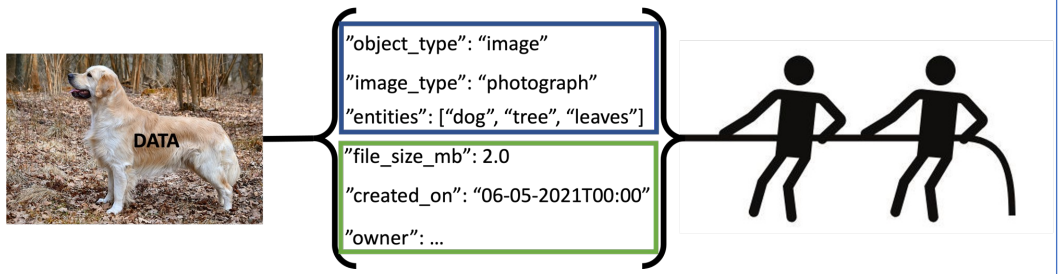
Data Lifecycle and Scalable Workflows Group
Oak Ridge National Laboratory

ParslFest & FuncX Fest 2022

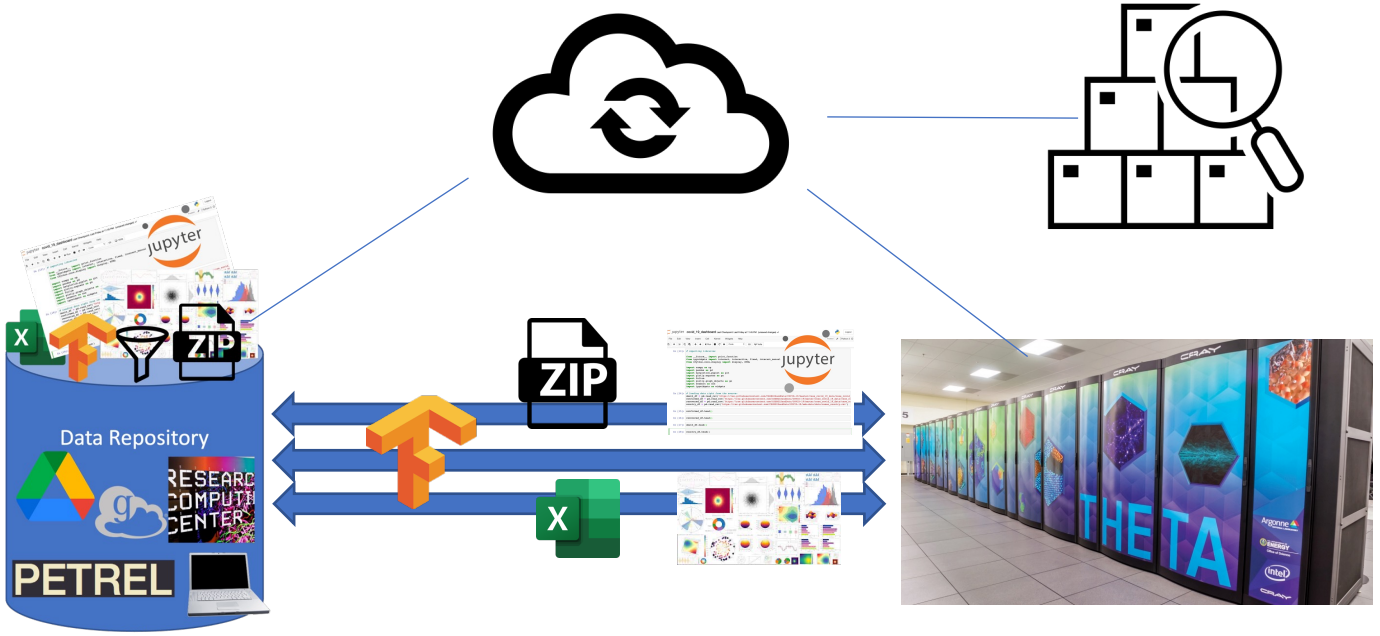
Scientists generate plentiful data artifacts along the research data lifecycle [Berman, ‘18]

Metadata extraction can help make these data navigable

<u>Option 1</u>: Manually
(olden days)
"Human annotation"



<u>Option 2</u>: Automatically
(Today's wonder years)
Metadata Extraction Systems

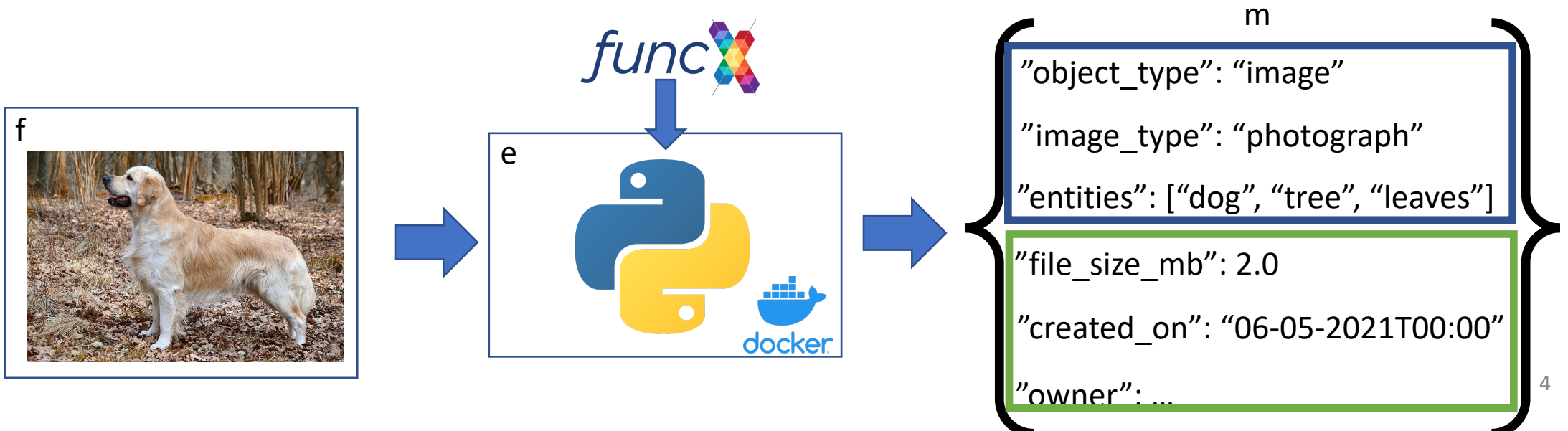


How to extract metadata from files of very different types?

Repository *R* has collection of files ***f*** $\in R$

M_f is universe of all possible metadata for ***f***

Set of **extractors** ε where $e \in \varepsilon$ is a function ***e(f)*** that returns a (potentially empty) set of new metadata elements $m \in M_f$



We have developed a broad extractor library that illuminates the long-tail of science data

General

JSON
ImageSort
Tabular
XML
C
Keywords
NetCDF
Python
Libmagic
ImageNER
HDF

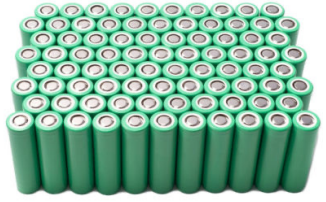
Principles for Extractor Design

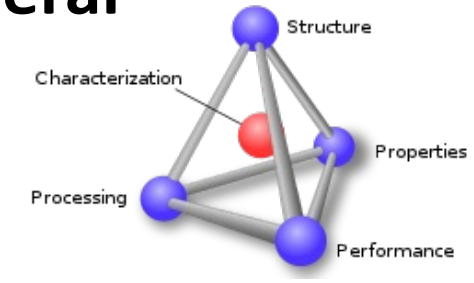
1. Relevant metadata
2. Correct metadata
3. Lightweight execution
4. Flexible to similar schema
5. Modular execution

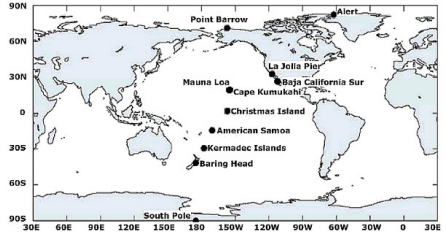
see Tyler's thesis for more information

Domain

Batteries
Materials
Spectroscopy
Climate
Maps







Extractors as funcX functions

```python
def base_extractor(event):
    from xtract_sdk.agent.xtract import XtractAgent

    # Load endpoint configuration. Init the XtractAgent.
    xtra = XtractAgent(xtract_dir=event['xtract_dir'],
                       sys_path_add=event['sys_path_add'],
                       module_path=event['module_path'],
                       recursion_depth=event['recursion_limit'],
                       metadata_write_path=event['metadata_write_path'])

    # Execute the extractor on the family_batch.
    xtra.execute_extractions(family_batch=event['fam_batch'], input_type=event['type'])

    # All metadata are held in XtractAgent's memory. Flush to disk!
    paths = xtra.flush_metadata_to_files(writer=event['writer'])
    stats = xtra.get_completion_stats()
    stats['mdata_paths'] = paths

    return stats
```

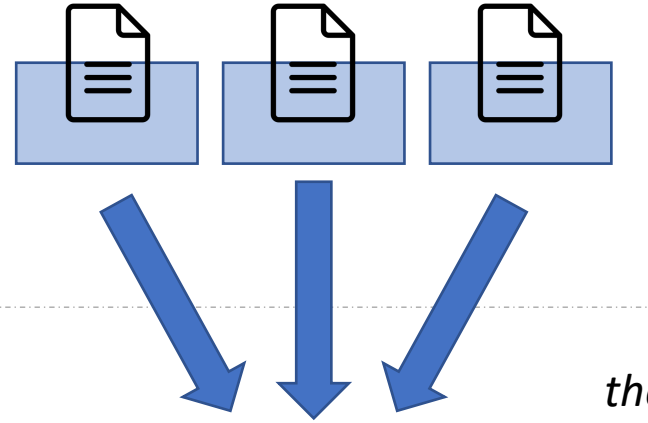
Uses funcX “container” workers

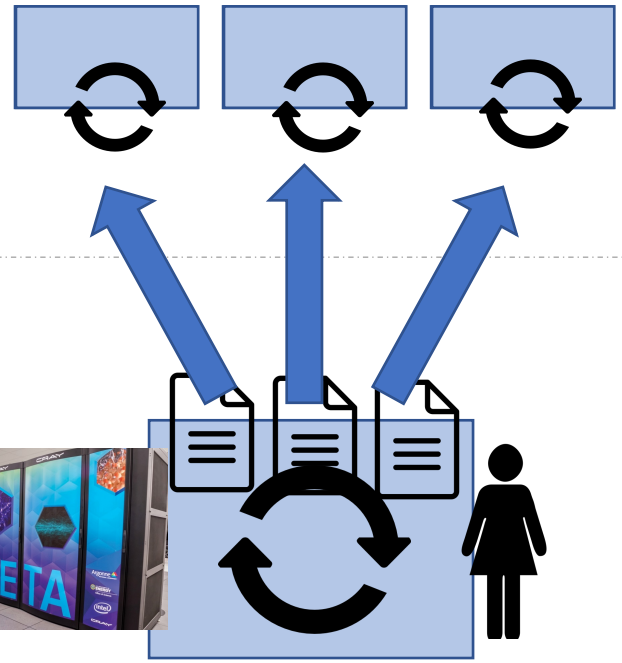
Check out our extractor library to see how hotdogs are made

https://github.com/xtracthub/xtract-tabular
https://github.com/xtracthub/xtract-keyword
https://github.com/xtracthub/xtract-python
https://github.com/xtracthub/xtract-images
https://github.com/xtracthub/xtract-matio
… and more!

Flexible computing models made possible by funcX & globus

the edge





the computing center

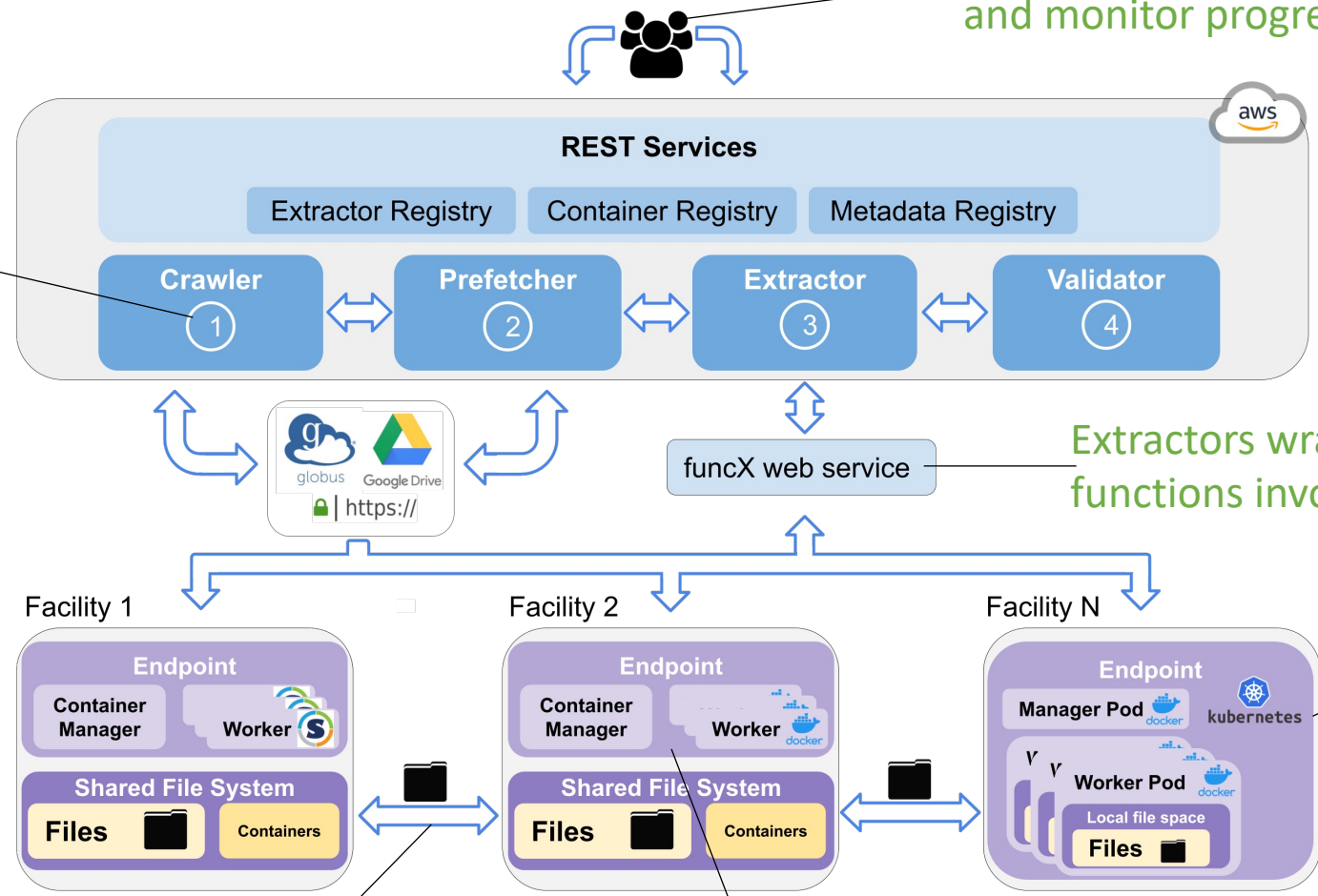
extract here

extract there

extract anywhere

Xtract: the metadata extraction system for science

Users submit extractors, launch jobs, and monitor progress via **xtract_sdk**



Crawler recursively scans each file in the repository

Extractors wrapped as FaaS functions invoked at facilities

funcX providers enable automatic scaling on diverse research cyberinfrastructure

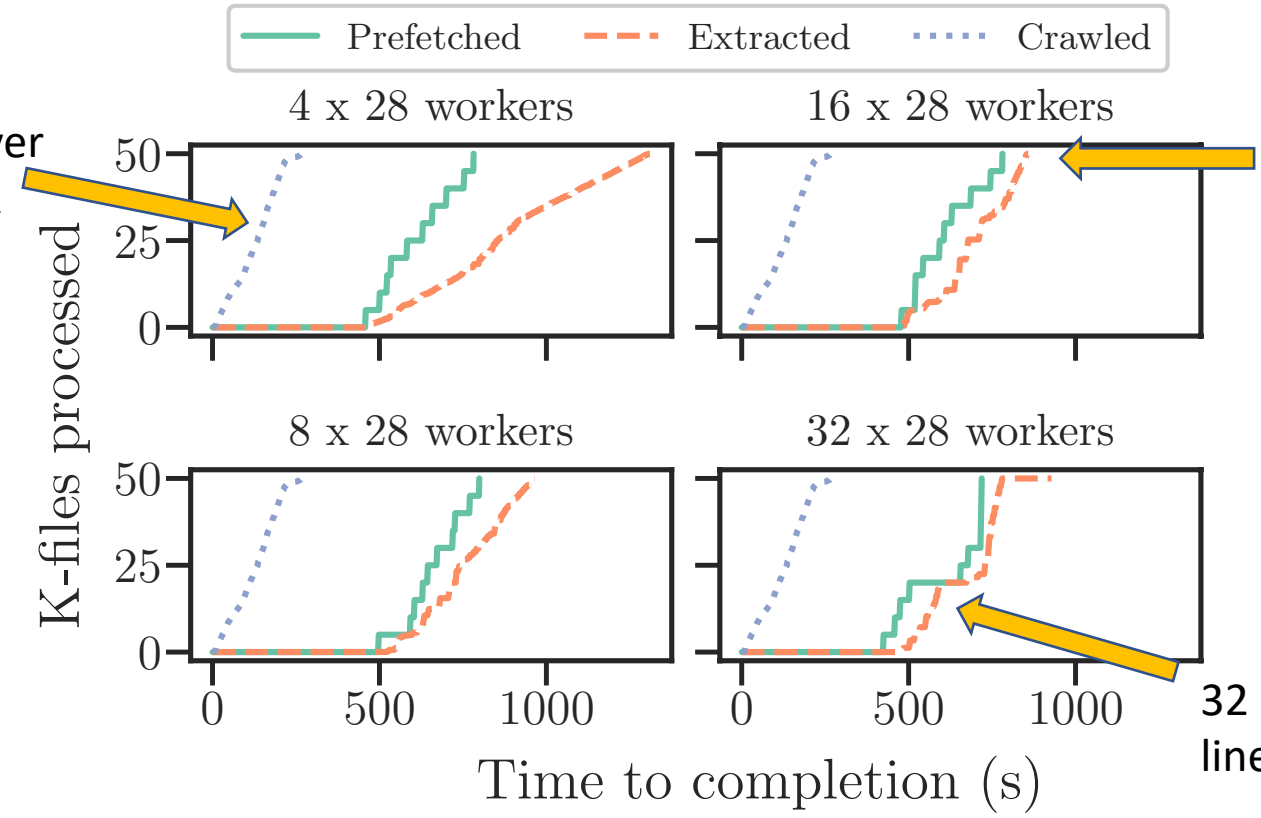
Files can be moved to leverage idle resources via Globus Transfer

Endpoints configured via CLI

Extract here: full repository transfer

The Materials Data Facility (MDF)

A simple way to publish, discover, and access materials datasets



crawling never a bottleneck

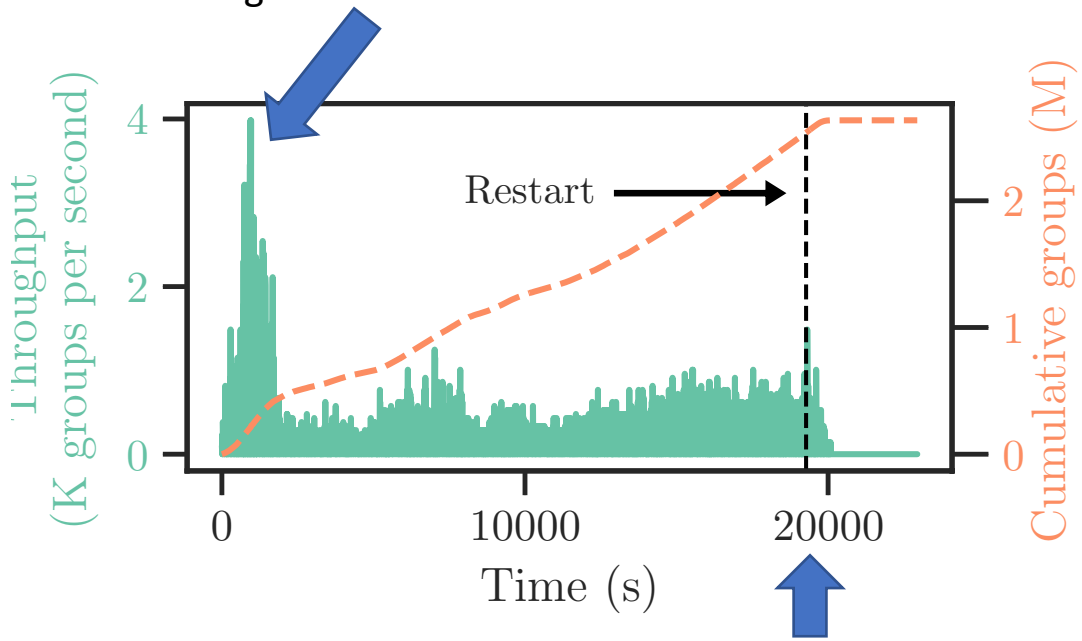
transfers packaged as ‘blocks’ with maximum 10GB or 20,000 files

32 nodes x 28 workers: extraction line “hugs” the prefetch line

Bulk metadata extraction times for an MDF subset (50,000 files) processed on 4—32 UChicago Midway2 nodes.

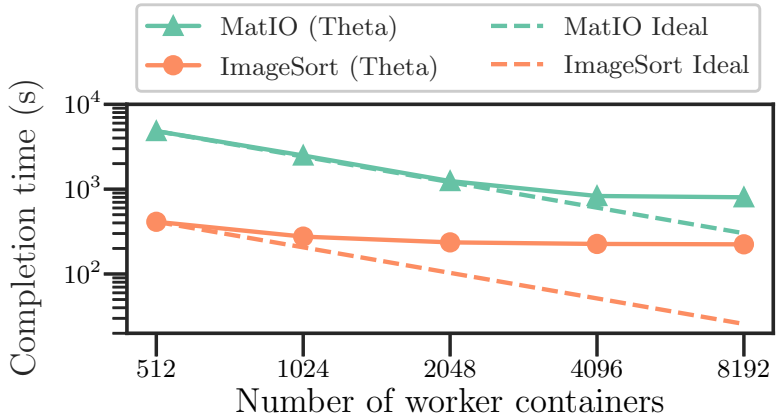
Extract there: process 60TB (2.2 million groups) using the Theta supercomputer in **just over 6 hours**

fast-running ASE extractions

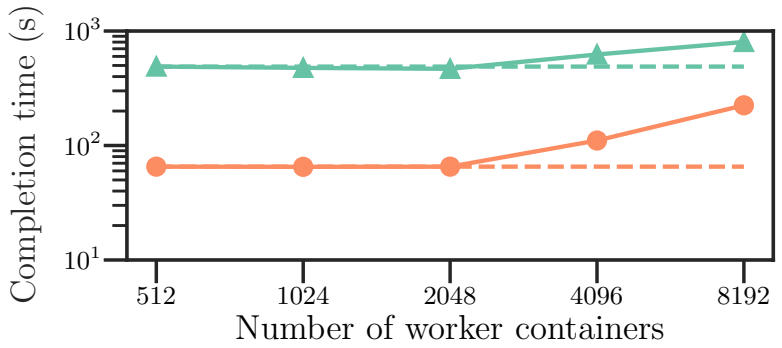


Checkpoint by writing metadata to file system until full batch completed

MatIO sees *reasonable* scaling up to 4,096 workers

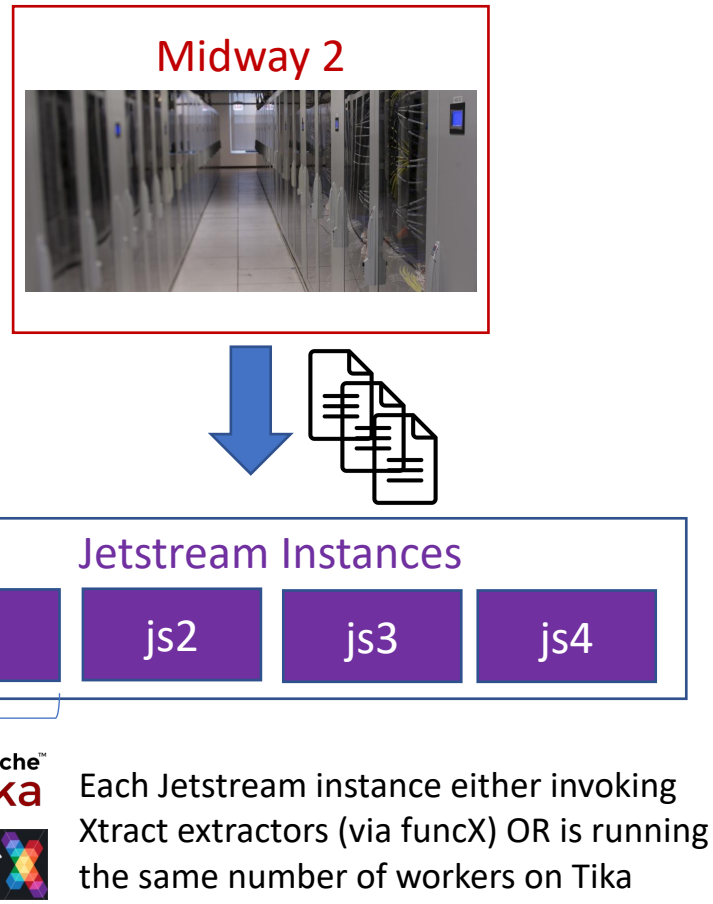


(a) Strong scaling



(b) Weak scaling

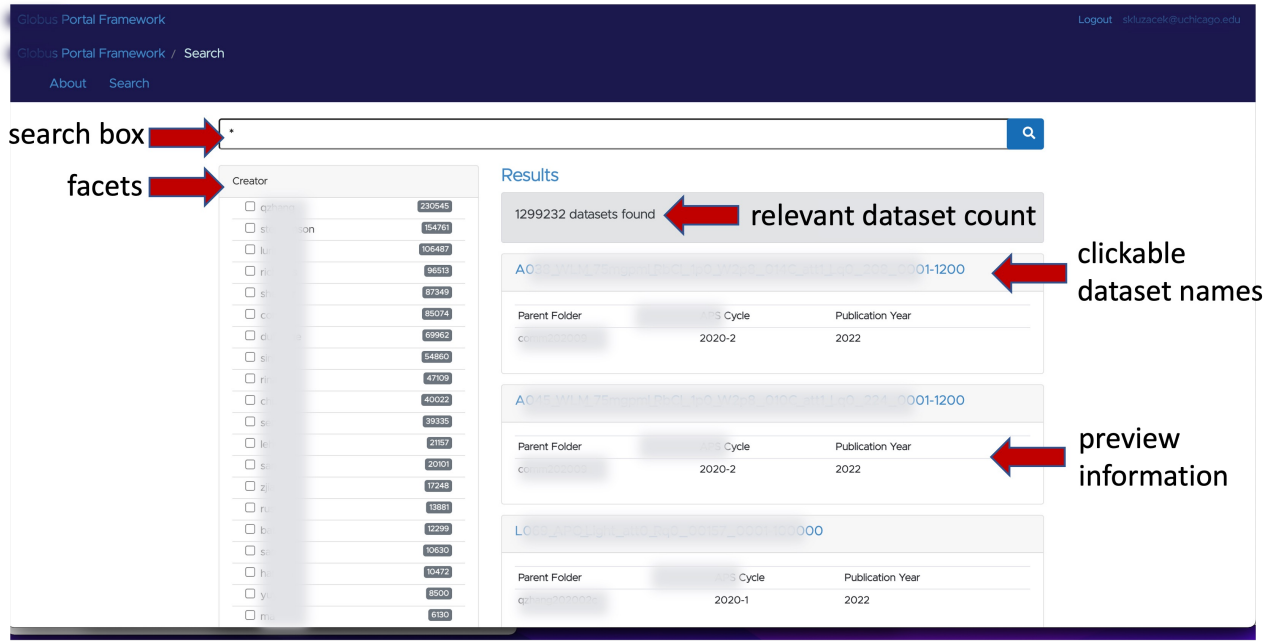
Extract anywhere: automatically offload files to underutilized compute facilities



Each Jetstream instance either invoking Xtract extractors (via funcX) OR is running the same number of workers on Tika

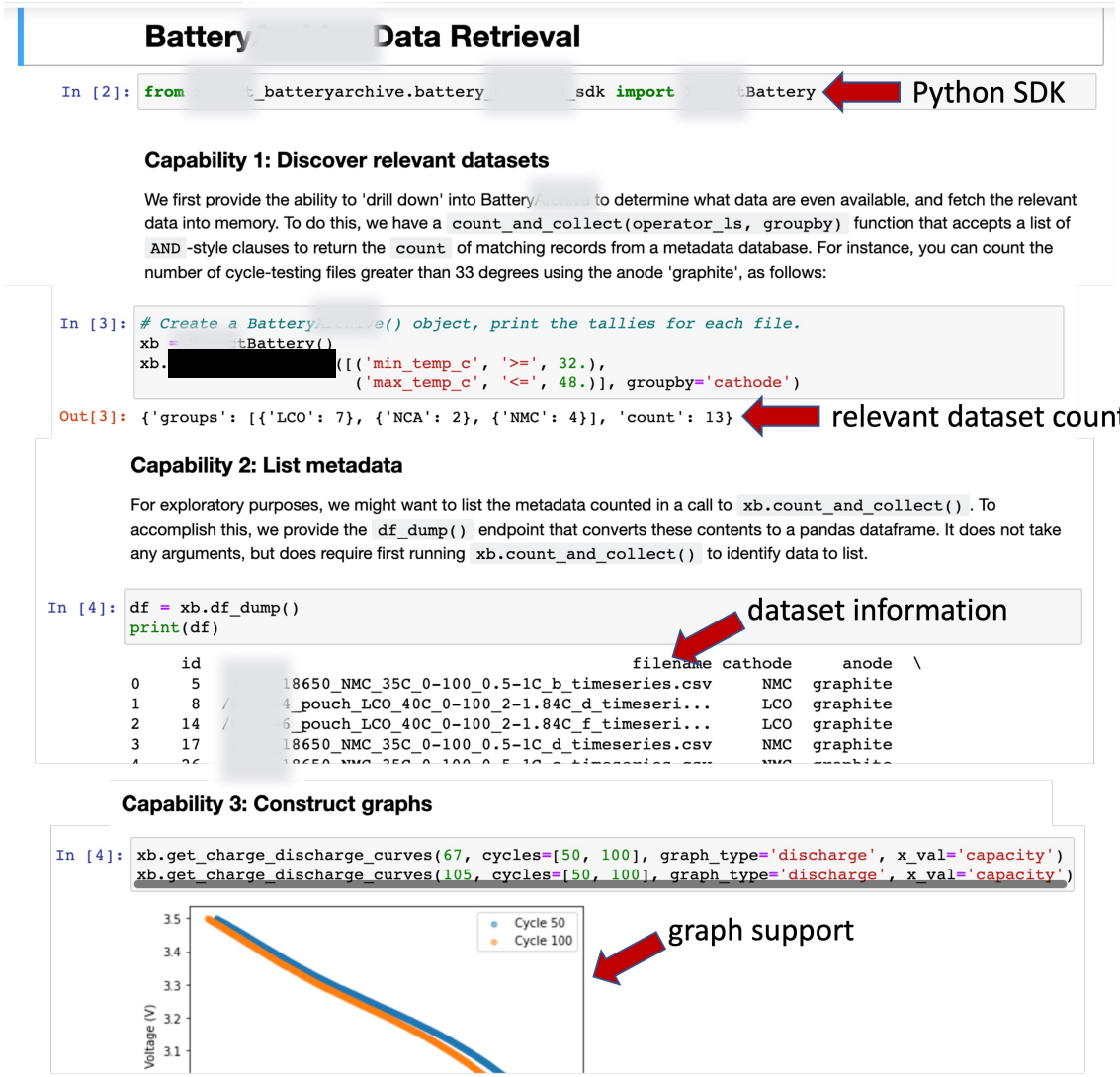
System	Percentage Transferred (%)	Transfer Time (s)	Completion Time (s)
Xtract	0%	0	1696
	10%	374	1560
	20%	655	1662
Apache Tika	0%	0	2032
	10%	384	1868
	20%	649	1935

10% is optimal, as Midway2 and Jetstream finish at approximately the same time



(shoutout to Nick Saint)

Spectroscopy: Globus Search Portal Interface



Battery Modeling: Python SDK on JupyterHub

Things I’m particularly excited about

- the funcX container service
- service-owned endpoints
- Minnesota Vikings 2023 Super Bowl run

Xtract is (soon-to-be, again) available for use

Future work

- Shared micro-extractor utilities to support shared extraction logic
- Combine ‘workflow’ and file metadata
- **Application:** AI-enhanced storage to support ML applications

Want to learn more?

skluzacektj@ornl.gov

We’re building in the open:

https://github.com/xtracthub