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

**Option 1:** Manually (olden days)  
“Human annotation”

**Option 2:** 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 $\varepsilon$ where $e \in \varepsilon$ is a function $e(f)$ that returns a (potentially empty) set of new metadata elements $m \in M_f$

```
"object_type": "image"
"image_type": "photograph"
"entities": ["dog", "tree", "leaves"]
"file_size_mb": 2.0
"created_on": "06-05-2021T00:00"
"owner": ...
```
We have developed a broad extractor library that illuminates the long-tail of science data.

**Principles for Extractor Design**
1. Relevant metadata
2. Correct metadata
3. Lightweight execution
4. Flexible to similar schema
5. Modular execution

**General**
- JSON
- ImageSort
- Tabular
- C
- Keywords
- Python
- NetCDF
- Libmagic
- ImageNER
- HDF

**Domain**
- Batteries
- Materials
- Spectroscopy
- Climate
- Maps

See Tyler’s thesis for more information.
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

Files can be moved to leverage idle resources via Globus Transfer

Endpoints configured via CLI

funcX providers enable automatic scaling on diverse research cyberinfrastructure
Extract here: full repository transfer


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.
Extract anywhere: automatically offload files to underutilized compute facilities

<table>
<thead>
<tr>
<th>System</th>
<th>Percentage Transferred (%)</th>
<th>Transfer Time (s)</th>
<th>Completion Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xtract</td>
<td>0%</td>
<td>0</td>
<td>1696</td>
</tr>
<tr>
<td></td>
<td>10%</td>
<td>374</td>
<td>1560</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>655</td>
<td>1662</td>
</tr>
<tr>
<td>Apache Tika</td>
<td>0%</td>
<td>0</td>
<td>2032</td>
</tr>
<tr>
<td></td>
<td>10%</td>
<td>384</td>
<td>1868</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>649</td>
<td>1935</td>
</tr>
</tbody>
</table>

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

10% is optimal, as Midway2 and Jetstream finish at approximately the same time.
Spectroscopy: Globus Search Portal Interface

Battery Modeling: Python SDK on JupyterHub

(shoutout to Nick Saint)
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