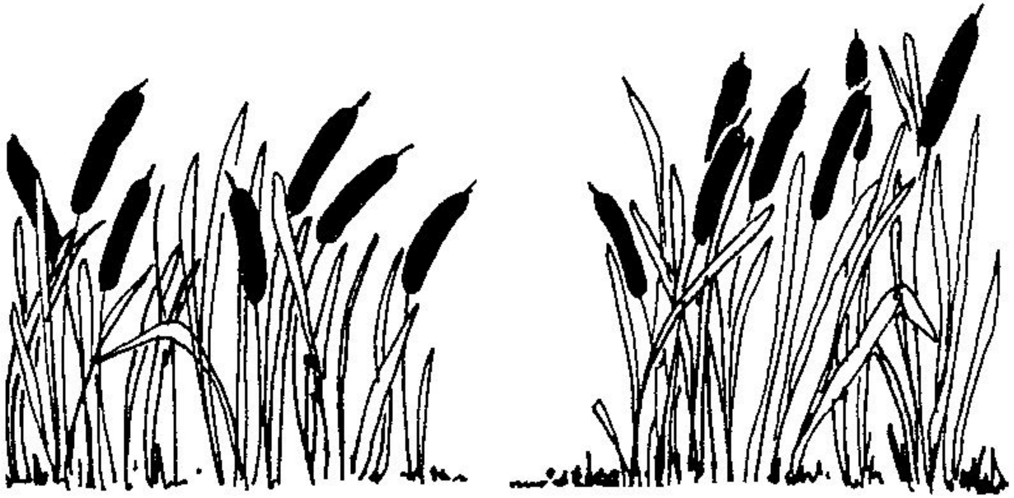
Serverless Workflows for Indexing Large Scientific Data

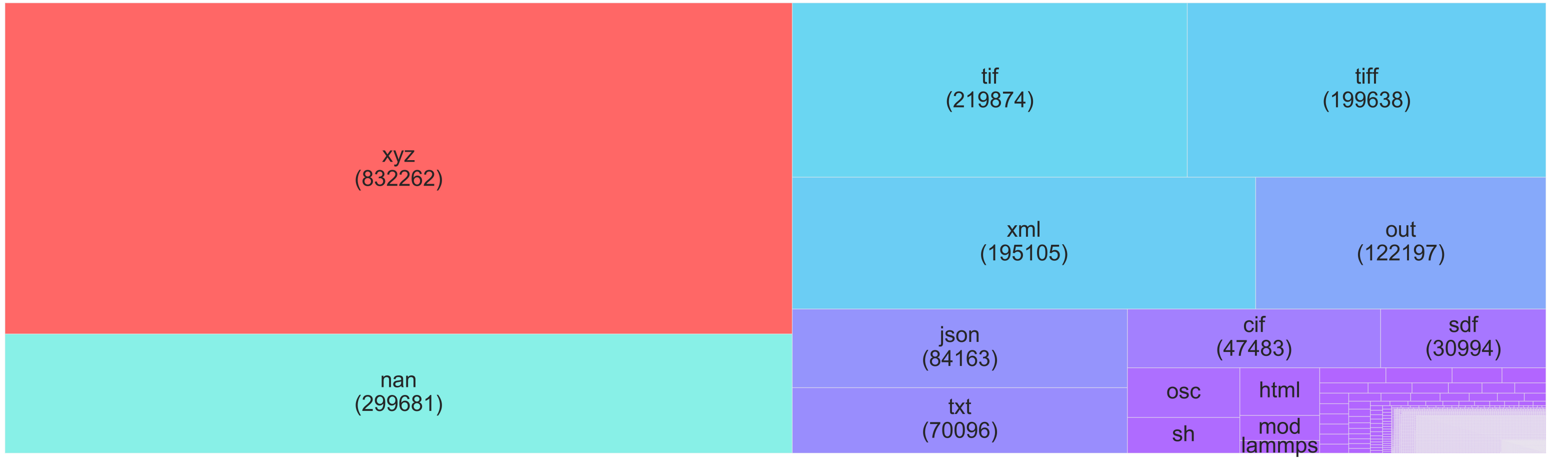
Tyler J. Skluzacek, Ryan Chard, Ryan Wong, Zhuozhao Li, Yadu Babuji, Logan Ward, Ben Blaiszik, Kyle Chard, Ian Foster



Data are big, diverse, and distributed

Big: petabytes → exabytes

Diverse: thousands → millions of unique file extensions



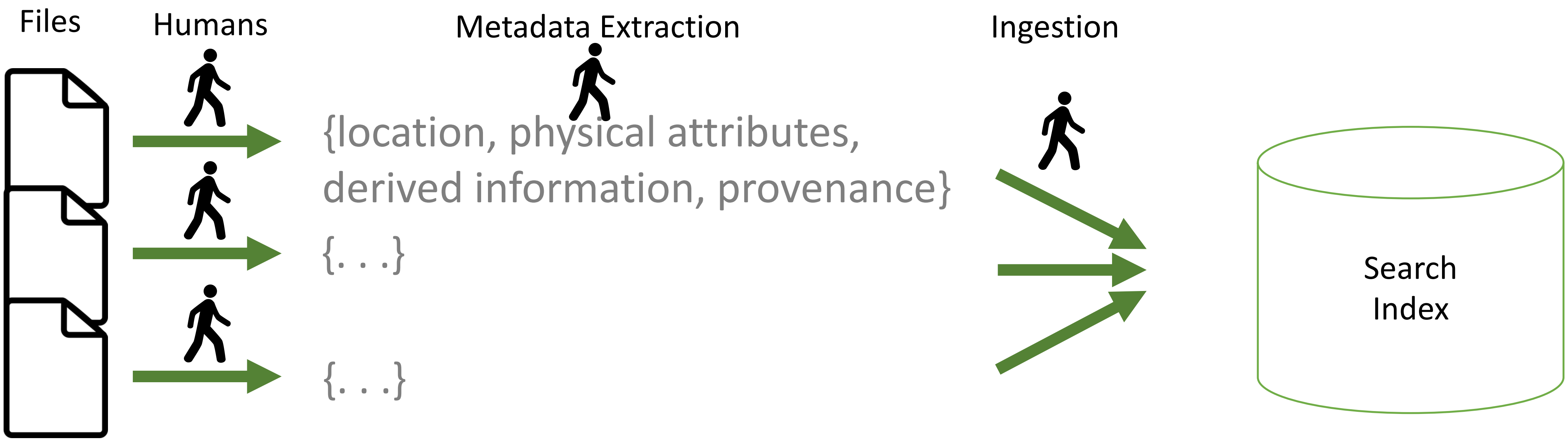
Distributed: IoT (edge), HPC, cloud; from many individuals

Generally, scientific data are not FAIR

Findable , **A**ccessible, **I**nteroperable, **R**eusable

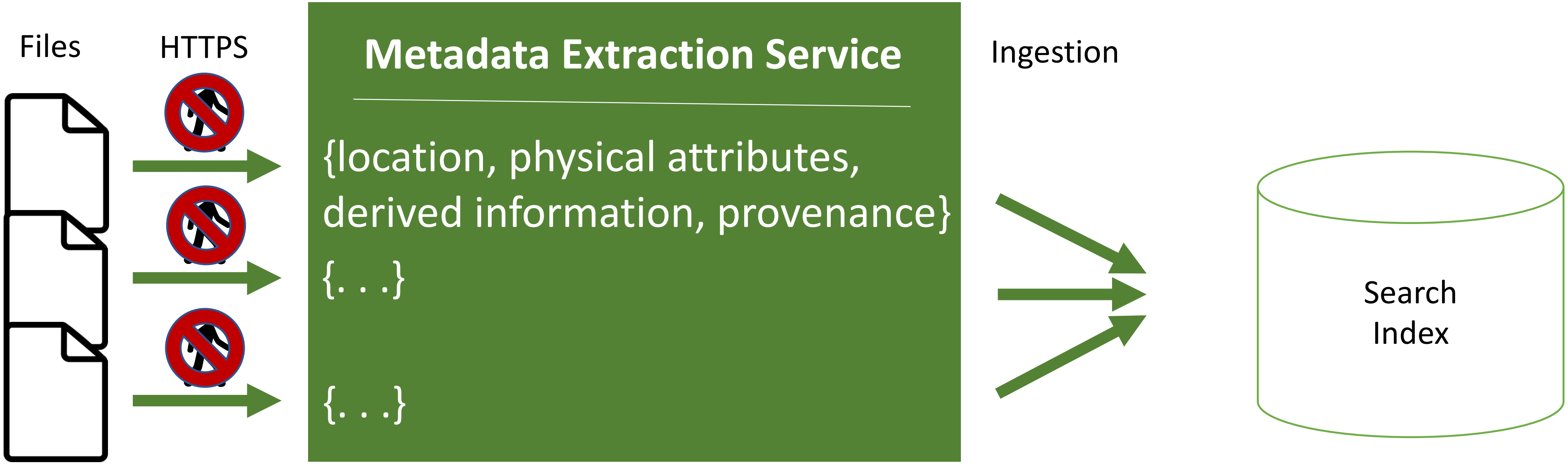
Root of the problem: files lack descriptive metadata

Root **of the root** of the problem: humans are lazy, metadata are hard



We need an automated metadata extraction system

Ideally, to cancel* humans



We need a flexible, decentralized, scalable metadata extraction system

1. Send metadata extraction functions to data
No need to ship big data

```
wc –l $FILE1
```

2. Decentralized
Extract the data in their natural habitats (e.g., edge)

```
wc –l $FILE1
wc –l $FILE2
```

3. Scalable

Run many concurrent metadata extraction processes

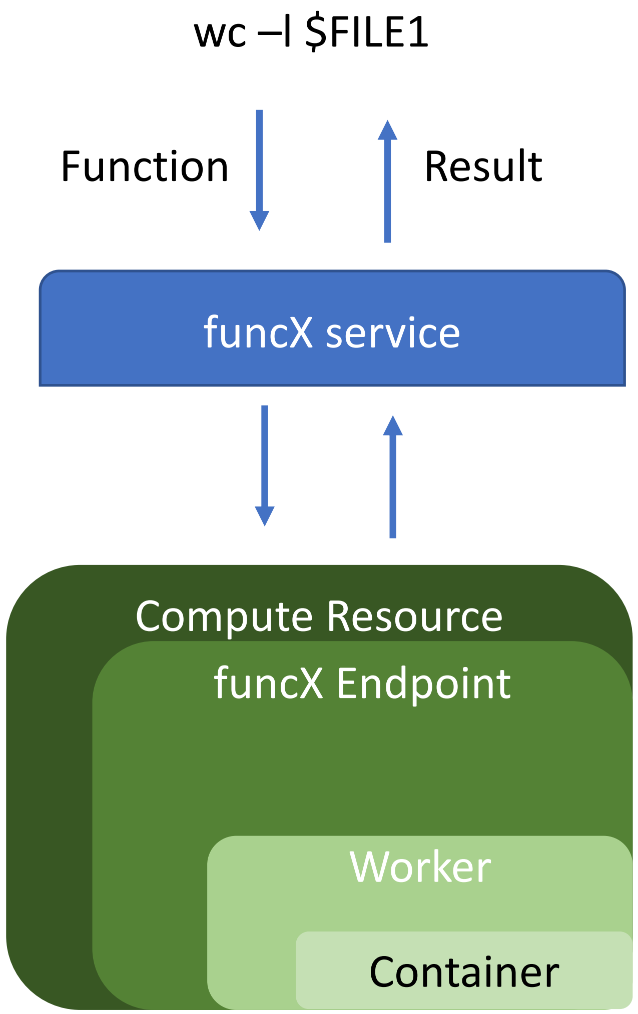
```
wc –l $FILE1
wc –l $FILE2
. . .
wc –l $FILE600000
```

funcX for FaaS anywhere

Enable secure, isolated, on-demand function serving on myriad compute resources (cloud, HPC, laptops, edge)

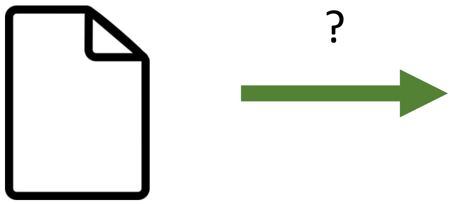
Abstract away underlying infrastructure via Parsl parallel scripting language

Users deploy endpoints on available compute resources, use Globus Auth for access control, and access a library of containers for running functions



Metadata Extractor = Function

Metadata Extractor: Instructions to create a mapping from input file to output JSON – e.g., looks like a function



{location, physical attributes, derived information, provenance}

Function: Python/BASH metadata extraction instruction

Payload: File or group of files which from which to extract

Function Containers: Containers containing all execution dependencies

Xtract: the serverless metadata extraction system

Built atop funcX

Deploy endpoints at heterogeneous compute resources
on cloud, laptops, HPC, scientific instruments

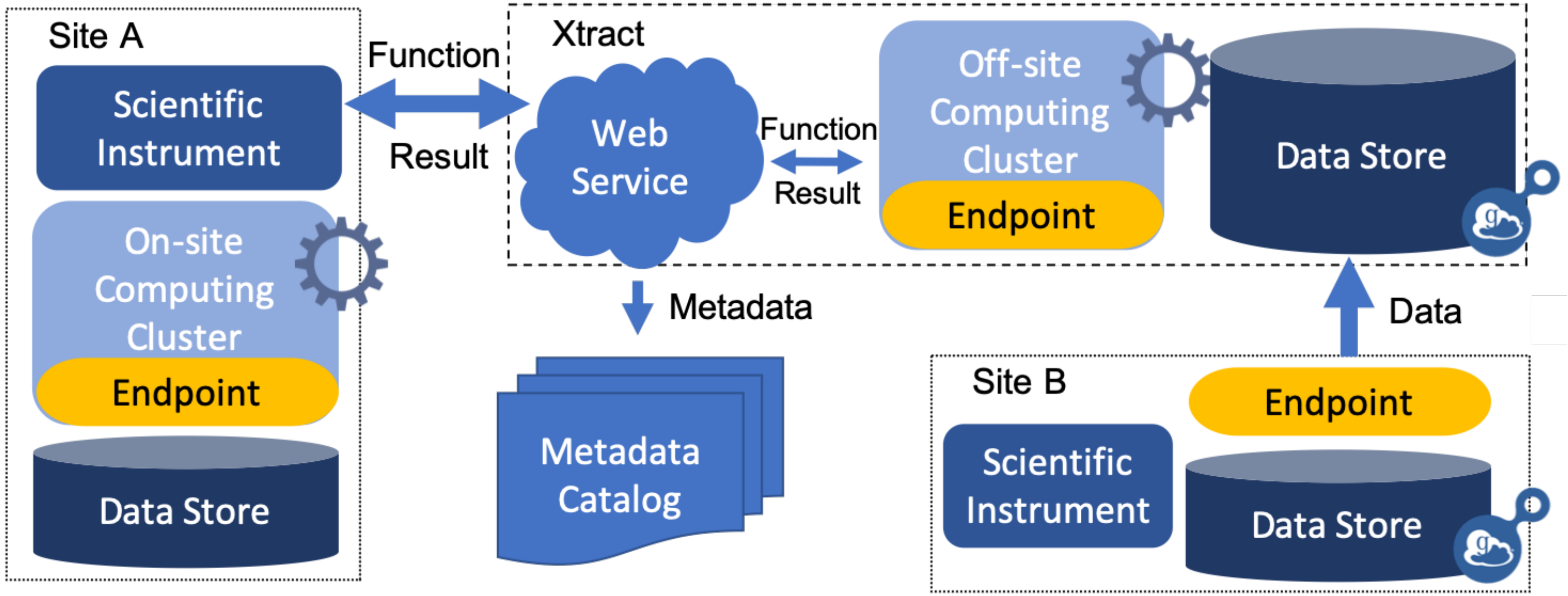
Central web service determines extractors to send to endpoints
Send extractors to data, receive results, determine future extractors

Secure
Use Globus Auth for access control on data collections and compute

Crawls any Globus-connected endpoints
Recursively generates file groups dir-by-dir

Prototype

Xtract: the serverless metadata extraction system



Site A: Compute at data
Site B: Compute elsewhere

Consider how one could automatically move data to available endpoints to suit constraints

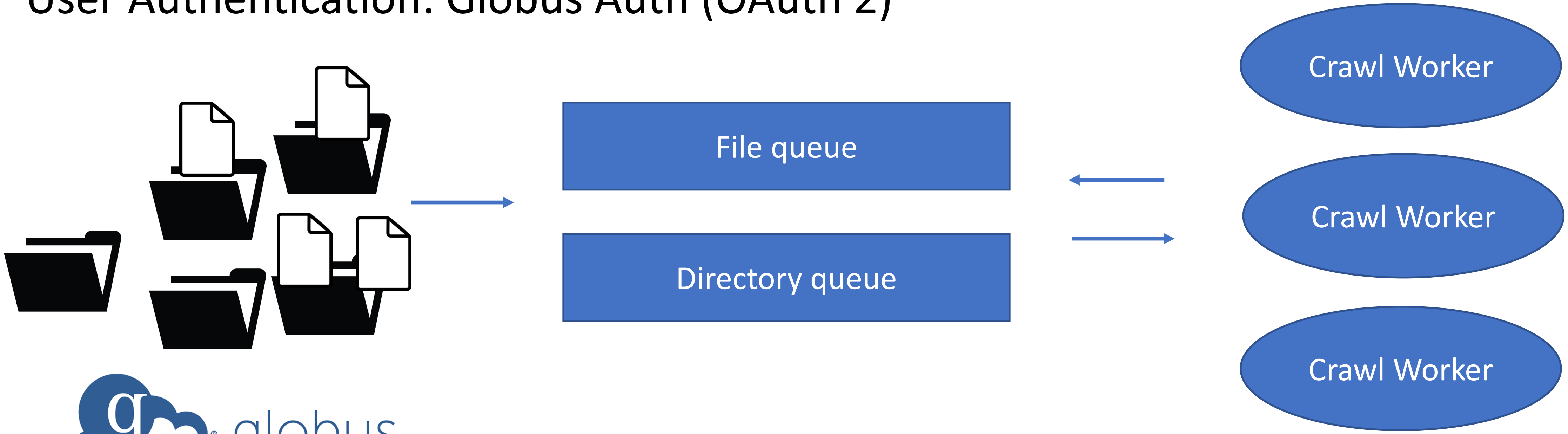
Step 1: Crawl the File System

Recursively crawls all files in all nested dirs located on Globus Endpoint

Generate an initial metadata index for each file/file-group

Extracts physical metadata (path, size, extension, last-edited)

User Authentication: Globus Auth (OAuth 2)



Step 2: File Type Identification

Need to “guess” a file’s type

Impractical to apply all extractors to all files (most yield no metadata)
Applying an incorrect extractor to a file can waste significant time

Random Forests model trained on 5% of files in a given repo

Features: 512 bytes from header

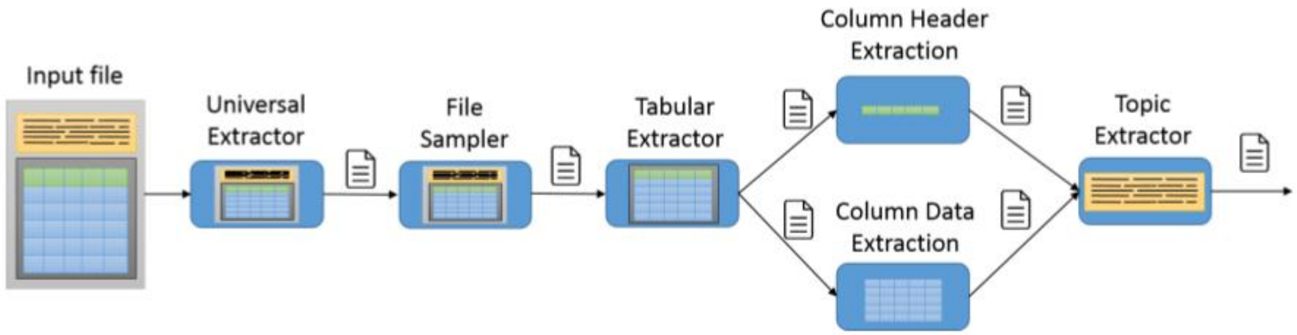
Training:

File’s type determined by first applicable metadata extractor to file

Feasible because extractors can find other applicable extractors

Step 3: Metadata Extractor Orchestration

Xtract uses file type identity to choose the first appropriate extractor



Extractors return results to service and may immediately deploy additional extractors to endpoint. This can be done recursively.

One file will likely receive multiple metadata extraction functions

Step 4: Ingest Metadata Document

Currently Xtract supports ingesting JSON directly to Globus Search

Diverse, Plentiful Data in Materials Science

The Materials Data Facility (MDF):

- is a centralized hub for publishing, storing, discovering materials data
- stores many terabytes of data from myriad research groups
- is spread across tens of millions of files
- is co-hosted by ANL and NCSA (at UIUC)

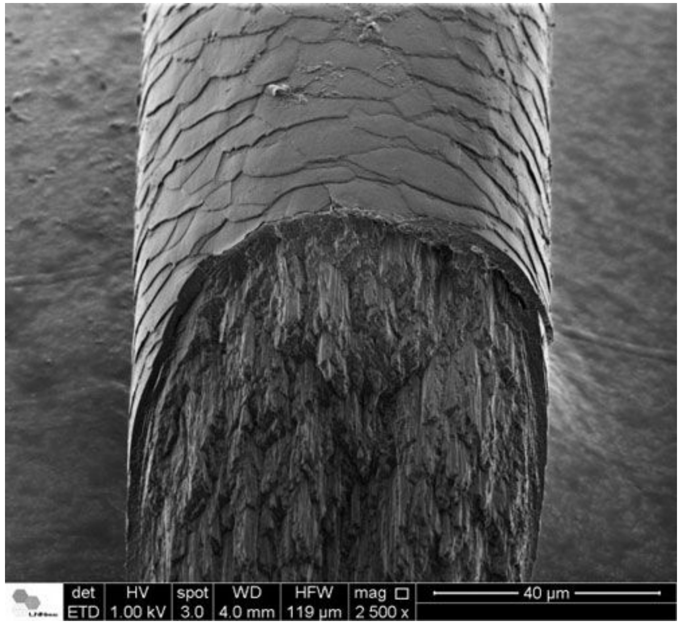
Thus, manual metadata curation is difficult

The Materials Extractor

Atomistic simulations, crystal structures, density functional theory (DFT) calculations, electron microscopy outputs, images, papers, tabular data, abstracts, . . .

MaterialsIO is a library of tools to generate summaries of materials science data files

We developed a 'materials extractor' to return summary as metadata



https://materialsio.readthedocs.io/en/latest/

Extractor Library

We operate a (growing!) suite of metadata extractors, including:

Extractor	Description
File Type	Generate hints to guide extractor selection
Images	SVM analysis to determine image type (map, plot, photo, etc.)
Semi-Structured	Extract headings and compute attribute-level metadata
Keyword	Extract keyword tags from text
Materials	Extract information from identifiable materials science formats
Hierarchical	Extract and derive attributes from hierarchical files (NetCDF, HDF)
Tabular	Column-level metadata and aggregates, nulls, and headers

Experimental Machinery

Xtract Service

AWS EC2 t2.small instance (Intel Xeon; 1 vCPU, 2GB RAM)

Endpoint
funcX deployed at ANL's PetrelKube
14-node Kubernetes cluster

Data
Stored on the Petrel data service (3 PB, Globus-accessible endpoint at ANL)

255,000 randomly selected files from Materials Data Facility

We evaluate Xtract on the following dimensions:

1. Crawling Performance
2. File Type Training
3. Extractor Latency

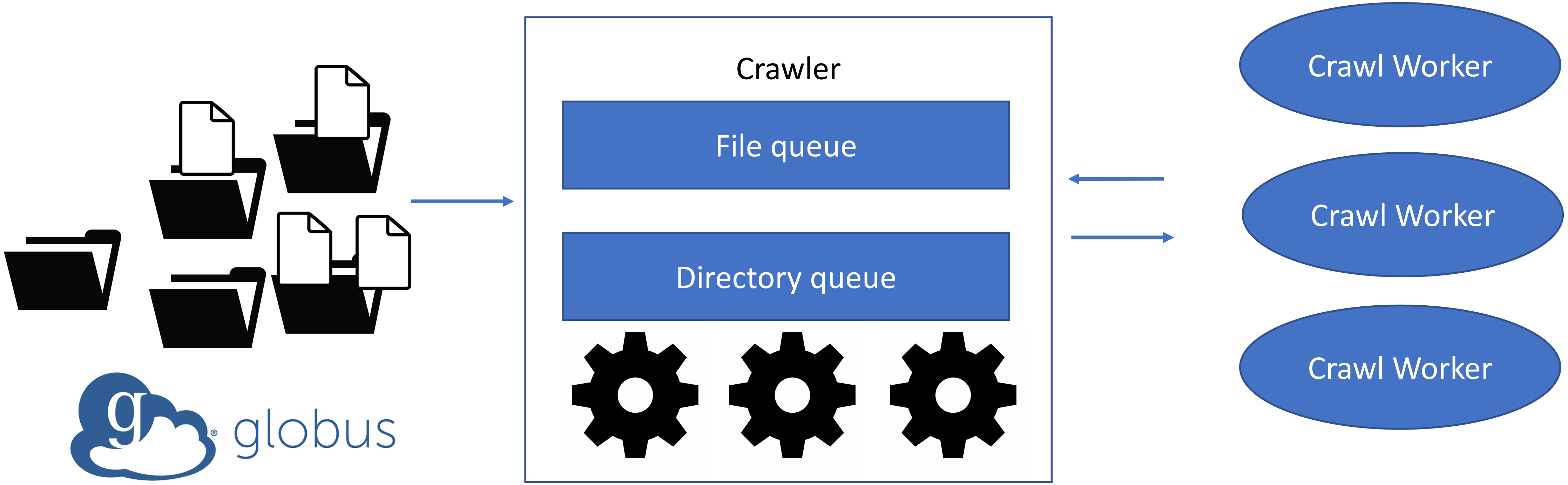
Future work will evaluate:

4. Metadata quality
5. Tradeoff optimization (transfer or move if nonuniform resource usage)

1. Crawling Performance

Sequential crawling: 2.2 million files in ~5.2 hours

Parallelization? Soon. The remote ls command was previously rate-limited, and a majority of directories have 0 or 1 files.



2. File Type Training

Train file type identification model on 110,900 files in MDF

Total time: 5.3 hours (one-time cost)

Label generation: 5.3 hours
Feature collection + random forests training: 45 seconds

Accuracy: 97%

Precision: 97%

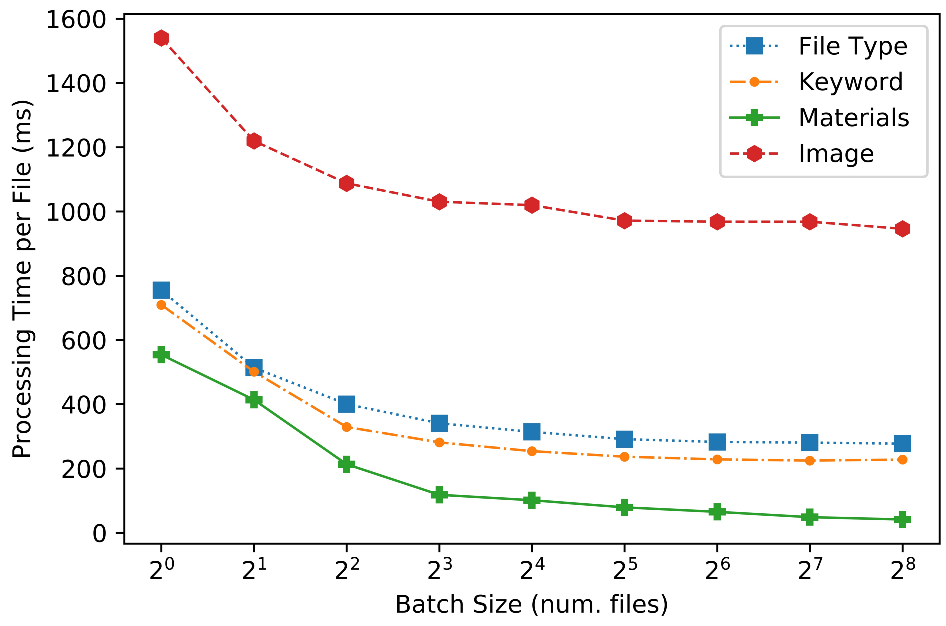
Recall: 91%

3. Extraction Performance

Extractor Latency

Extractor	# Files	Avg. Size (MB)	Avg. Extract Time (ms)	Avg. Stage Time (ms)
File Type	**255,132**	1.52	3.48	714
Images	**76,925**	4.17	**19.30**	1,198
Semi-Str.	29,850	0.38	8.97	412
Keyword	25,997	0.06	**0.20**	346
Materials	**95,434**	**0.001**	**24**	1,760
Hierarch.	3,855	695	1.90	9,150
Tabular	1,227	1.03	113	625

Batching



Conclusion

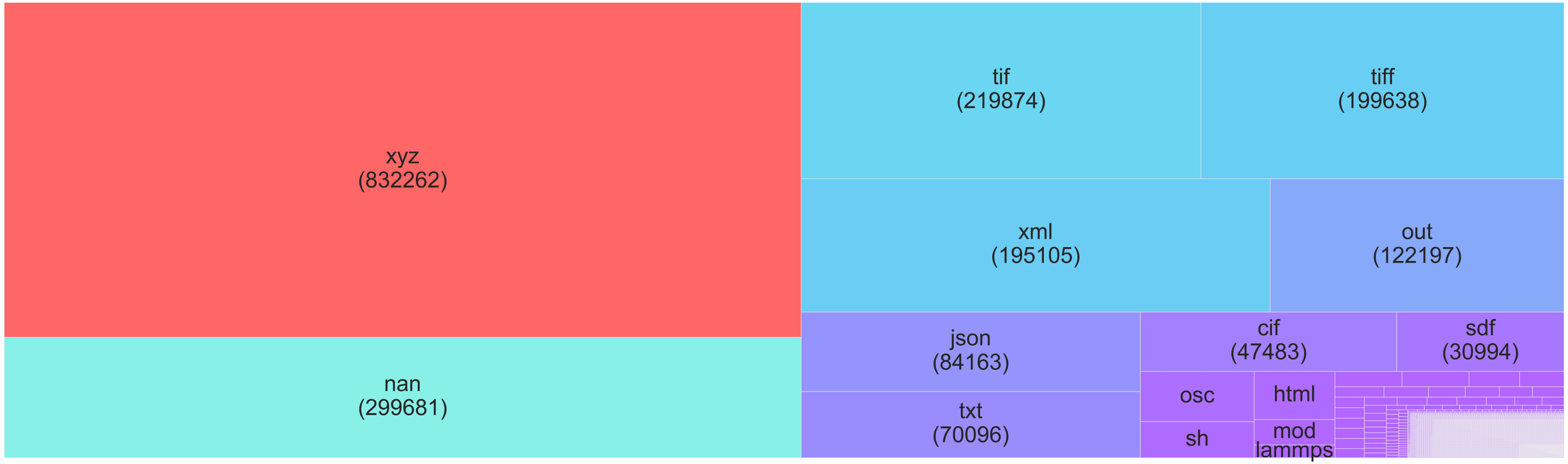
Data are big, diverse and distributed and are not FAIR (by default)

Xtract is a prototype that enables scalable, distributed metadata extraction on heterogeneous data stores and compute resources

Future work predicates on taking advantage of heterogeneous, distributed resources subject to a number of usage and cost constraints

Next up: index the full 30+ million file Materials Data Facility

Learn more about future work at the Doctoral Symposium



skluzacek@uchicago.edu

Doctoral Symposium Article:

"Dredging a Data Lake: Decentralized Metadata Extraction". Tyler J. Skluzacek. Middleware '19