Designing Scalable Cyberinfrastructure for Metadata Extraction in Billion-Record Archives

Gregory Jansen
University of Maryland
jansen@umd.edu

Richard Marciano
University of Maryland
marciano@umd.edu

Smruti Padhy
NCSA/UIUC
spadhy@illinois.edu

Kenton McHenry
NCSA/UIUC
mchenry@illinois.edu

ABSTRACT

We present a model and testbed for a curation and preservation infrastructure, "Brown Dog", that applies to heterogeneous and legacy data formats. "Brown Dog" is funded through a National Science Foundation DIBBs grant (Data Infrastructure Building Blocks) and is a partnership between the National Center for Supercomputing Applications at the University of Illinois and the College of Information Studies at the University of Maryland at College Park. In this paper we design and validate a "computational archives" model that uses the Brown Dog data services framework to orchestrate data enrichment activities at petabyte scale on a 100 million archival record collection. We show how this data services framework can provide customizable workflows through a single point of software integration. We also show how Brown Dog makes it straightforward for organizations to contribute new and legacy data extraction tools that will become part of their archival workflows, and those of the larger community of Brown Dog users. We illustrate one such data extraction tool, a file characterization utility called Siegfried, from development as an extractor, through to its use on archival data.

Keywords

Computational archival science, Digital curation, Data mining, Metadata extraction, File format conversion, Brown Dog, Cyberinfrastructure, Big data

1. INTRODUCTION

1.1 The Data Observatory in Maryland

The Digital Curation Innovation Center (DCIC) at the UMD College of Information Studies ("Maryland's iSchool") is building a 100 Million-file data observatory (called CIBER – "cyberinfrastructure for billions of electronic records") to analyze big record sets, provide training datasets, and teach students practical digital curation skills. At 100 Million files we seek to anticipate the billion-file scale, testing approaches on collections one order of magnitude removed. The DCIC is contributing to a $10.5M National Science Foundation / Data Infrastructure Building Blocks (DIBBs)-funded project called "Brown Dog", with partners at the University of Illinois NCSA Supercomputing Center. The DCIC is also partnering with industry storage leader NetApp and an archival storage startup company, Archive Analytics Solutions, Ltd. As a newly formed center for digital curation, we are fortunate to collaborate on a project that addresses large-scale challenges and has extraordinary strategic potential. The Brown Dog project[1] is the largest of the implementation awards to date under the NSF Data Infrastructure Building Blocks (DIBBs) program. Brown Dog is creating web-scale infrastructure services that open up data collections for appraisal, analysis, and reuse. The approach has been described as creating a new infrastructure service for the web, like a domain name service (DNS) for data, the idea being that data-focused services are a missing component of the web we have today. The role of the DCIC in Brown Dog is to use these infrastructure services to enrich the collections and meet the curation challenges we face in data-driven research.

1.2 Digital Legacies and Format Debt

Heterogeneous data accumulate in research and memory institutions, but often remain locked in native formats and are not easily accessed or understood as rich, informative sources. Legacy files will often not open in current software readers or viewers. Moreover, their internal information remains opaque to modern search and analytic approaches. As the files accumulate, so do the missed opportunities to effectively exploit them. We refer to this accumulation of effectively opaque file formats as a type of institutional debt, "format debt", which we would quantify as the theoretical technology investment required to reveal the complete intellectual content of all the accumulated files.

The existence of a functionally opaque format is rarely due to the lack of available software. Many legacy and current software tools can process legacy file formats and reveal their intellectual content. From commercial Windows applications, such as CorelDraw, to Linux-hosted computer vision tools for image processing; the available software list goes on and on. The challenge with "format debt" is not the lack of software, but the instrumentation of all the associated software in a workflow.

Each software executes in a particular technical environment, including the required operating system and machine architecture. A technical expert must set up each software environment, devise a way of passing in a file, running the software, and interpreting the results. The myriad of old

[1] http://browndog.ncsa.illinois.edu/

and new software tools produce diverse output formats that rarely conform to current standards like JSON or RDF. These are the real barriers to instrumentation. It requires a significant investment to add each different software to the workflow. Unlike a digital collection of a single format, big archives of born digital materials contain thousands of formats. Big archives require a new strategy to tackle spiraling "format debt" and for that reason we explore integrations between archival collections and Brown Dog services.

1.3 CI-BER Testbed

Our testbed explores how the Brown Dog services [1, 7] can be applied within a large organization's archives, to reveal the data within the diverse file formats of archival collections. We present a model architecture for a born-digital repository that inserts Brown Dog services into a repository workflow that also includes a scalable mix of search and analysis services, namely Indigo (Cassandra), Elasticsearch, and Kibana.

Several extractor tools have been developed for our testbed, with archives in mind. We use these as examples of community-developed tools added to the Brown Dog tools catalog, which is designed for such user contributions.

Lastly, the 100 Million files in the CI-BER data set are being used to systematically test the Brown Dog service APIs. These tests include load tests, to ensure that performance does not degrade under web-scale load, and qualitative tests of the services' response to diverse file formats.

2. BACKGROUND

Brown Dog (BD) [1, 7] is a set of extensible and distributed data transformation services, specifically, data format conversions, named Data Access Proxy (DAP), and metadata extraction from data content, named Data Tilling Service (DTS). With ever increasing varieties of data formats, data sometimes becomes inaccessible due to obsolete software/file formats. The DAP, through a set of REST APIs, allows users to convert inaccessible data to accessible formats, thus unlocking valuation information. Similarly DTS, through a set of REST APIs, allows users to extract metadata, signatures, tags or any other possible feature information from a file's content. Using the extracted information, files can be indexed and retrieved based on data content.

The scale and scope of the Brown Dog data service will often prompt comparisons with the SCAPE project for Scalable Preservation Environments[2]. Both are aimed at preserving data that resides in diverse file formats, but they are highly complementary. Brown Dog specifically focuses on building a cloud-based service, allowing data to broadly transcend format. In contrast SCAPE pursued diverse strategies, tools, and policies for digital preservation. SCAPE policies can provide a decision-making framework for ongoing preservation activities, whereas Brown Dog can provide the supporting metadata and format conversions. Brown Dog's DAP and DTS REST services are a natural fit for use in SCAPE preservation work flows.

The DAP, built on top of Polyglot framework [4, 5], does file format conversions, i.e. it converts an input file to a given output format. It encompasses several Software Servers (SS). A SS uses a wrapper script (alternatively, known as *Converter* within BD) which wraps any piece of code, third party software, or library, to provide access to their conversion capabilities (e.g. open/save/convert) through a consistent REST interface. The wrapper script also provides information on the input and output formats supported by the underlying software, and thus, available through the SS. The DAP performs a format conversion by obtaining the available input/output formats from all the different SS, chaining these together for all possible conversion paths. It then finds the shortest conversion path from a given input format to a given output format. Lastly, the DAP performs the format conversion by passing the file data through the chain of SS along the shortest path. A user can write a wrapper script (or a custom converter) for the software she uses and contribute that to the Tools Catalog. Then a Software Server (SS) containing her script can be deployed within BD services and can be made available for other users to leverage.

The DTS is built on top of the Clowder framework [3, 6] and performs metadata extractions on-demand from a given input file's content. The extraction process is triggered based on file mime type and then carried out by any appropriate extractors that are available. An extractor is a software process that listens for extraction requests from DTS on a message queue (RabbitMQ). It performs extraction of metadata from the file through analysis of the content, generating rich JSON-LD[3] metadata and tags, creating previews, and breaking files out into sections with more details. Each extractor then uploads the new information to Clowder, where it is combined with results from other extractors and made available through the DTS REST API. Using the pyClowder[4] library, a DTS user can write her own extractor that can use any piece of code, software, library, or webservice extraction functionality under the hood, and can potentially be deployed as a BD service for other users.

The DAP's SS and the DTS's extractors reside in a distributed environment such as the cloud. To handle heavy load, adapt to peak user demand and support heterogeneous architectures, another module, named the Elasticity Module (EM), has been incorporated into the BD. EM automatically scales up or down the DAP's SS and DTS's extractors based on user demands and loads. It monitors conversion and extraction requests in the RabbitMQ queues for SS and extractors. If any queue length exceeds a particular threshold, it launches another instance of that specific extractor or SS listening to the respective queue. Current implementation of EM uses the NCSA OpenNebula Openstack cloud for launching a new VM with a SS or extractor. It also has the option of using Docker as another layer of virtualization. Thus, this EM design allows BD services to dynamically grow/shrink based on demand and also ensures scalability. A detailed description of the BD services architecture can be found in [7].

3. BROWN DOG WITHIN THE CI-BER DATA INFRASTRUCTURE

The DCIC has accumulated a large-scale archival data repository in collaboration with the National Archives and Records Administration (NARA) consisting primarily of federal and community-sourced digital archives, both born-digital

[2]http://scape-project.eu/

[3]http://json-ld.org/

[4]opensource.ncsa.illinois.edu/bitbucket/projects/CATS/repos/pyclowder

and digitized, which were part of an earlier NSF-funded CI-BER project [2]. The DCIC's Digital Archives Repository for Research and Analytics (DARRA) houses more than 100 Million files and 72 Terabytes of unique, heterogeneous data.

The DCIC staff rely on assistance from the Division of Information Technology (Div IT) staff on campus and our industry partner for the Indigo repository software, Archival Analytics, to maintain the DARRA facility. DARRA equipment occupies half of a rack in a campus data center. Our storage array is maintained there on-site by NetApp services. The four virtual machine hosts are administered by Div IT staff. These relationships allows us to build and maintain the facility and carry out Brown Dog research in virtual machines, with a single software architect on staff. For production operations of this kind a dedicated system administrator is also required, providing for more formal change management, reporting, and vacation coverage.

The DCIC approach to curation infrastructure relies upon distributed databases, messaging, and virtual machine technology to eliminate bottlenecks and create linear scalability. The archival repository and related services are run on a cluster of four physical servers. The servers have high bandwidth connections to a peta-scale NetApp storage array. These physical servers play host to a constellation of guest virtual machines that run all the software. The DCIC is working with industry partners NetApp and Archive Analytics, Ltd., a big data startup, to build a scalable storage facility. Our catalog uses Archive Analytics' repository software, Indigo; a resilient and scalable solution for storage virtualization and workflow automation. Based on the Apache Cassandra distributed database[5], Indigo gives us high performance data access over a standard cloud storage API (Cloud Data Management Interface – CDMI), which is critical to data processing activities. The Indigo repository software is to become a community open source initiative.

The Brown Dog service is integrated with the catalog through its two main web endpoints. The Data Access Proxy (DAP) exposes an API for arbitrary, on-demand file format conversion. The Data Tilling Service (DTS) provides an API that runs all of the metadata extraction tools that are appropriate to a submitted file. In our workflow the DAP and DTS APIs are called by very simple worker scripts that are written in Python. The worker scripts submit CI-BER data to the Brown Dog APIs and place the resulting metadata back into the Indigo catalog. A pool of Python workers are always available to handle this work, which is queued and tracked in a local RabbitMQ message queue. The Brown Dog workflow may be triggered automatically by placing a new file into Indigo, or it may be run on-demand, when the existing repository hierarchy is traversed.

For those building systems on a similar scale, the hardware in the data center rack totals 166,000 US dollars, which breaks down into $29,000 for the four servers and $137,000 for the NetApp storage array. The raw storage costs are a little over $190 per terabyte. The NetApp storage is used for the archives and also parceled out for other virtual machine needs, such as databases and index space.

[5]http://cassandra.apache.org/

4. CONTRIBUTE YOUR TOOL TO BROWN DOG AND SHARE

Researchers often build new tools for their research, in order to extract useful information from unstructured or semi-structured data and to do necessary file format conversions in the process. A lot of effort goes into developing such tools and such efforts are often unacknowledged. In addition, similar tool development efforts are repeated by multiple researchers within the same domain of research. Towards acknowledging such tool development efforts, the BD Tools Catalog (TC) was designed and implemented to be a framework for these contributions. BD TC is a web application where a user can upload any existing/new tool with conversion or extraction capabilities, e.g., imagemagick[6], and tesseract[7] and can share it with the research community. It has a web interface to upload BD tools (alternatively, known as BD scripts). A BD tool/script is a script that wraps the original software developed by a researcher, or third-party software, and make it deployable within the BD service. The TC can also deploy these BD tools to the cloud environment in an automated way. Thus, members of different research communities can contribute and share their tools or BD scripts within the BD framework using the TC. In the TC web user interface, users can provide citation information about their tool and will get proper credit for their effort in creating the software

A BD script that wraps the tool's conversion capability and exposes it within BD service is known as converter; while a BD script that wraps the extraction capability of the tools and makes it available within the BD system is known as extractor. In subsequent subsections we will explain how to write an extractor or a converter through examples pertaining to archival data. In [7], creating of a BD tool (extractor or a converter) has been described in brief. To make writing of an extractor easy, a python package known as pyClowder was developed that handles common interactions with Clowder.

4.1 Create BD Tools

The CI-BER data observatory primarily consists of archival data, records from many federal agencies, cities, and civic organizations. These data are in many formats and in a variety of original folder structures. The unique challenge for the digital archivist at this scale is simply to know what they have in these collections and where, such that they can take appropriate preservation actions and provide access to researchers. We looked at the many extractors provided by the NCSA team, a compendium of computer vision and 3D modelling feature extractors, amongst others. We found that we needed additional extractors more germane to digital preservation practice, namely file characterization and digest. We created three extractors specific to archival data with the aim of applying them within CI-BER.

4.1.1 Siegfried extractor

The first extractor is based on Siegfried [8], which is a fast file characterization tool. It identifies sub-formats or format versions and other format characteristics about each file. These formats are discovered through byte matching with

[6]http://www.imagemagick.org/
[7]github.com/tesseract-ocr

Code 1: Connects to RabbitMQ with proper credentials

```
# connect to rabbitmq
extractors.connect_message_bus(extractorName =
    extractorName, messageType = messageType,
    processFileFunction = process_file,
    rabbitmqExchange = rabbitmqExchange,
    rabbitmqURL=rabbitmqURL)
```

the file patterns found in a registry of format signatures, PRONOM [9]. PRONOM, run by the National Archives of the United Kingdom, was the first web-based, public format registry in the world. Siegfried in particular is a project of Richard LeHane [8]. We use PRONOM-based file characterization in order to understand the exact format used in a file. The formats identified by file extension can be arbitrary, as data files can be renamed in arbitrary ways and as they often are unrecognized in older archival data. The Siegfried extractor is a BD tool that uses Siegfried, signature-based format identification tool, under the hood. Now provided as a BD extraction service, this is helping us obtain format data from the 100 million files that make up CI-BER.

To write a Siegfried extractor, we use the Clowder integration package for Python, pyClowder; and the Siegfried tool. Code snippet 1 shows the way to connect to RabbitMQ with proper credentials. Code snippet 2 shows the implementation of the *process_file* method based on the Siegfried tool and also how to upload the extracted metadata to the Clowder web app. The metadata extracted can be accessed using DTS API. Note the way Siegfried is called within the *process_file* method (Line 8). *connect_message_bus* and *upload_file_metadata_jsonld* are methods from pyClowder package.

4.1.2 FITS extractor

The second extractor is based on File Information Tool Set (FITS)[9], a file characterization toolkit. The FITS wraps several of the known digital preservation tools within it. They all run on a given file and the results are presented in one report where they can be compared, including points of agreement and disagreement. FITS is slower to run than Siegfried, but produces more data for analysis. It includes DROID, which does exactly the same PRONOM-based format identification as Siegfried. So a FITS file report will allow an archivist or an archival analytics tool to compare PRONOM identification with other tools, such as the Linux FileInfo tool, which has its own internal list of formats and byte patterns.

4.1.3 Byte Digest

The third extractor we created was for computing byte digests for files. We created a python based extractor that efficiently computes the MD5, SHA1, SHA256, SHA384, and SHA512 digests in a single pass through the data. Repositories rely on these kinds of digests to ensure the fixity of data across a variety of storage systems. Repositories may want to rely on the DTS for all forms of data extraction, or as a third-party cross-check to compare with digests created

[8]github.com/richardlehane/siegfried
[9]http://projects.iq.harvard.edu/fits

Code 2: BD script- Siegfried Extractor's process file implementation and upload methods

```
# Process the file and upload the results
def process_file(parameters):
    global extractorName

    inputfile = parameters['inputfile']

    # call the Siegfried (sf) program
    resultStr = subprocess.check_output(['sf',
        '-json', inputfile],
        stderr=subprocess.STDOUT)
    result = json.loads(resultStr)

    afile = result['files'][0] # always one file
        only

    content = {} # assertions about the file
    content['dcterms:extent'] = afile['filesize']

    matches = []
    for match in afile['matches']:
        _logger.info(match)
        m = {}
        if 'id' in match:
            m['@id'] = 'info:pronom/'+match['id']
        if 'format' in match:
            m['sf:name'] = match['format']
        if 'version' in match:
            if len(match['version'].strip()) > 0:
                m['sf:version'] = match['version']
        if 'mime' in match:
            m['sf:mime'] = match['mime']
        if 'basis' in match:
            m['sf:basis'] = match['basis']
        matches.append(m)

    if len(matches) > 0:
        content['dcterms:conformsTo'] = matches

    #wraps the metadata in JSON-LD format
    jsonld_metadata = jsonld_wrap (content)

    # upload metadata (metadata is a JSON-LD array
        of dict)
    extractors.upload_file_metadata_jsonld(mdata =
        jsonld_metadata, parameters = parameters)
```

within proprietary systems. By comparing a locally computed digest with the digest coming back from the DTS, we can also ensure that the file data sent to the DTS was able to reach the extractors intact.

4.1.4 Imagemagick Converter

In this subsection we provide an example of a converter to be deployed within the BD system. We chose imagemagick, a third-party software with file format conversion capabilities. As described in [7] to write a converter, we provided in the comment of the script - line 2 : software name with version number, line 3: data type supported, i.e., image, in this case, line 4: list of input formats supported by imagemagick, line 5: list of supported output formats. Line 12 and 14 contain the actual imagemagick *convert* function call that converts an input file in supported input format to specific supported output format. For example, 3 script allows conversion of an image in pcd format to svg format.

Code 3: BD Script - Imagemagick Converter

```
#!/bin/sh
#ImageMagick (v6.5.2)
#image
#bmp, dib, eps, fig, gif, ico, jpg, jpeg, jp2, pcd,
     pdf, pgm, pict, pix, png, pnm, ppm, ps, rgb,
     rgba, sgi, sun, svg, tga, tif, tiff, ttf, x,
     xbm, xcf, xpm, xwd, yuv
#bmp, dib, eps, gif, jpg, jpeg, jp2, pcd, pdf, pgm,
     pict, png, pnm, ppm, ps, rgb, rgba, sgi, sun,
     svg, tga, tif, tiff, ttf, x, xbm, xpm, xwd, yuv

output_filename=$(basename "$2")
output_format="${output_filename##*.}"

#Output PGM files as ASCII
if [ "$output_format" = "pgm" ]; then
 convert "$1" -compress none "$2"
else
 convert "$1" "$2"
fi
```

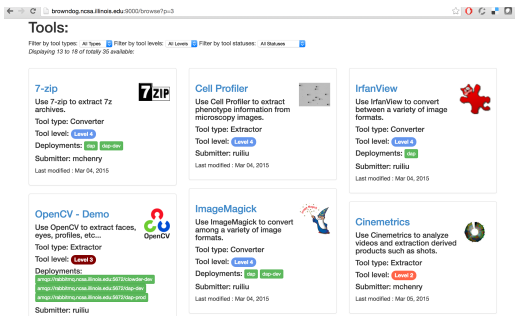

Figure 1: Tools Catalog web user interface showing list of tools/BD tool available in TC

4.2 Contribute and Share tool

To enable users to contribute and share a tool and its corresponding BD scripts, a Tools Catalog (TC) web application has been designed and is provided as a web service for BD users. Figure 1 shows the TC web user interface where a user can browse all tools information and BD tools/scripts that are being shared through TC, and can download BD scripts. It also has options to add tools information and contribute BD scripts and for admin to approve/disapprove a submitted BD tool/script. Figure 2 shows the specific tool information, e.g., Siegfried software information, after it has been added to TC.

5. INTEGRATED BROWN DOG TOOLS WITH REPOSITORIES

The DCIC team has integrated a number of services around the Indigo archival repository in Maryland. For demonstration purposes we have installed several Elasticsearch[10] nodes and the Kibana visualization tool[11]. In addition to rich search, these give us metrics and visualizations of the collections as they are enhanced with new data from Brown Dog.

[10]www.elastic.co
[11]www.elastic.co/products/kibana

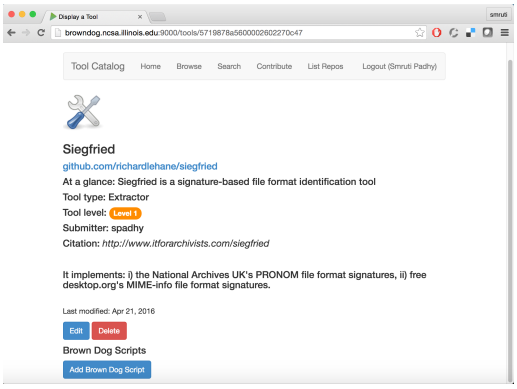

Figure 2: Displays Siegfried tool information after it has been added to Tools Catalog using Add Tool form with proper citation.

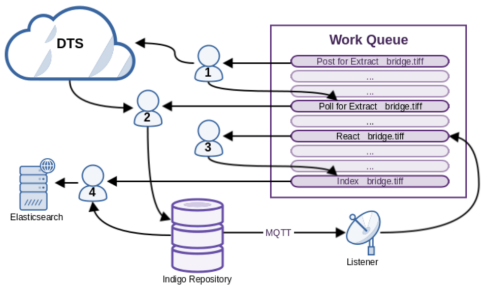

Figure 3: Workers and Task Queue

5.1 Simple Middleware

In order to coordinate the workflows we want around the Indigo repository, Brown Dog services, and Elasticsearch, we needed to create additional middleware. This middleware is mostly a work queue system, which allows us to perform work asynchronously, and at a predictable rate, instead of having to perform all operations immediately upon deposit or immediately in response to Indigo data changes.

When a user makes a change to the data in the Indigo repository, say they upload a new file, a message is generated and broadcast to any software that is listening to the Indigo message feed. The first step in the DCIC workflow is to listen for these Indigo messages. Our listener converts each message into a task and places that task in our work queue. The first task, called React, can be described as "respond to this Indigo message". Our simple listener has no trouble keeping up with all of the Indigo changes, as the significant work has been postponed for later.

Next in the workflow we have a pool of workers, see Figure 3. These are software processes that are waiting to pick up and perform any work from the work queue above. At the DCIC we normally have ten workers running, but if the queue of work keeps growing due to a large number of deposits, we can increase the number of workers. Workers can be distributed across multiple servers if necessary. There may be a variety of tasks added to the work queue and these workers can perform any of them. Let's look at some of the tasks that make up our workflow.

5.2 Tasks That Support Workflow

- **React** - Responds to the content of an Indigo message. This task is created by the Indigo listener. The worker will figure out what change was made in Indigo and what tasks should be performed as a result. For instance, any metadata change will result in an index task. This task adds other tasks to the work queue.
- **Index** - Indexes (or re-indexes) a file or folder in Elasticsearch. The worker will fetch any necessary data from Indigo.
- **Deindex** - Removes a file or folder from Elasticsearch.
- **Post for Extracts** - Uploads a file to Brown Dog's Data Tilling Service (DTS) for feature extraction. Adds a Poll for Extracts task to the queue.
- **Poll for Extracts** - Checks to see if the DTS extract operations above are complete yet. If incomplete, this task is scheduled to run again after a delay. If complete, the worker downloads the extracted metadata from DTS and puts this metadata into Indigo.
- **Text Conversion** - Uploads a file to the DAP for conversion into a text file, if possible. Schedules a Poll for Text task to run after a delay.
- **Poll for Text** - Checks to see if text conversion is available from DAP yet. If text is not yet available, this task is scheduled to run again after a delay. If text is available, it downloads the text and puts the text into a full text field in Indigo metadata.

Each step in the workflow is separated into a discrete task and each task is only performed when it gets to the front of the work queue. We can monitor the work queue to see how long it has become and how long tasks must wait to be performed. Organizations with available server resources can scale up the number of workers to keep up with demand. Organizations with few server resources can control server load and still eventually process the queued jobs.

The asynchronous middleware we describe here was implemented in the Python language and uses Celery[12] task queues. The work queues that are managed by Celery are persisted in a RabbitMQ messaging service. Each task may be relatively simple. For example Code 4 shows the complete code for the React task.

As shown, the React task schedules other tasks on repository paths in response to the Indigo operation. Other tasks are longer and involve requests to web services, either Brown Dog, Indigo or Elasticsearch. In some cases further workflow steps will result indirectly, via calls to Indigo services. For example, after full text is added to an Indigo metadata field, then the listener will be notified and will schedule a React task, then the React task will schedule an Index task, which will update Elasticsearch to include a full text field.

5.3 Workflow On Demand

As we incorporate more services into the workflow, or add new fields to our Elasticsearch index, we will add new workflow reactions in the React task. These will respond to new file deposits and changes in the data. However, we also want

[12]http://www.celeryproject.org/

Code 4: Code Sample for the React Task

```
@app.task
def react(operation, object_type, path,
    stateChange):
 if 'create' == operation:
     index.apply_async((path,))
     if 'resource' == object_type:
         postForExtract.apply_async((path,))
 elif "update_object" == operation:
     index.apply_async((path,))
 elif "delete" == operation:
     deindex.apply_async((path, object_type))
```

to trigger these new workflows on existing repository data. For this we turn to a special task called Traverse:

- **Traverse** - Traverse schedules another task for each file or folder within a given part of the repository tree, starting at the root and extending in a breadth-first manner to the branches. Traverse lets you perform workflow on demand for large areas of the repository. Traverse works recursively, making use of the work queue to schedule a further Traverse task for each subfolder at a given level. Recursive traverse tasks can reliably process folders of arbitrary depth without any long running worker processes.

For instance, if we add new fields to our Elasticsearch, we will traverse the entire repository to apply the Index task. If we add a new workflow, such as conversion to plain text, to the React tasks, we can apply the new workflow to existing data through the Traverse task. We add Traverse tasks to the queue directly, via a command-line script, rather than through the Indigo listener.

5.4 Taking Incremental Steps

With the 100 Million files in CI-BER collections we approach the problems of billion-file scale. Even given an asynchronous work queue, if we traverse a large collection without pausing, we will quickly overload the work queue with pending tasks, bringing the machine it runs on to a halt. Instead a traverse must proceed in stages. The traverse task has special logic that checks the length of the pending task queue and postpones itself whenever the queue is too large. A traverse will only proceed with creating more tasks while the queue is of a manageable size. In Figure 4 you can see the size of the overall queue over time, including all tasks, as we gradually traversed a collection. Each bump was created by a traverse operation that waited until the queue was small and then added more tasks.

In this way the workers can gradually traverse the entire repository to bring all materials up to date with respect to the current workflow.

6. VISUALIZATION OF EXTRACTED METADATA

The integration between Indigo, Brown Dog, and Elasticsearch creates an expanded set of metadata fields in the repository. When these are indexed in Elasticsearch, we can ask new questions and understand the collections and folders at every level in greater level of detail. All of the

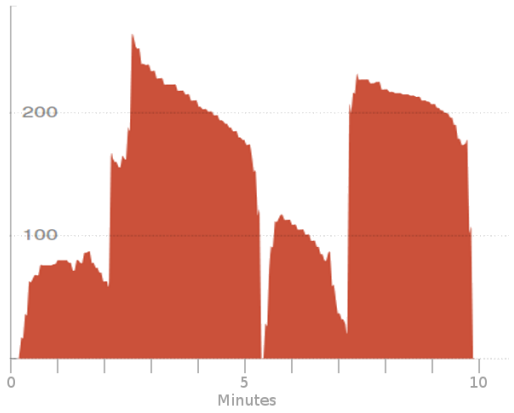

Figure 4: Incrementally Adding Work to the Task Queue over Time

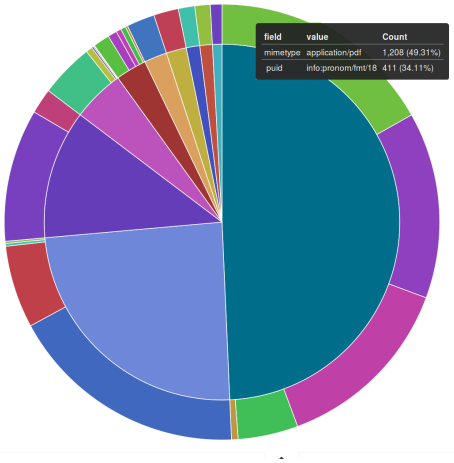

Figure 5: Kibana visualization of mimetype (inner sections) and format (outer sections)

following charts were created in the Kibana visualization tools for Elasticsearch, using data drawn from Brown Dog services. Each chart is part of the overview provided by a Kibana dashboard. The Kibana dashboard, our overview, can be redrawn with arbitrary index fields as filters, much like the drill-down feature of a faceted search system. Most importantly we can look at the dashboard of visualizations for any folder in CI-BER to better understand the contents.

6.1 Format Distribution

One of the insights we gain from the Siegfried extractor is detailed format information for every file we process. This chart captures a high level view of the most common file formats in the repository. The concentric pie chart shows mimetypes in the inner circle and then breaks these mimetypes down into specific sub-formats in the outer ring.

In its web-based interactive form, this chart has pop-up labels and can be used to target further preservation and access enhancements, such as format migration or format specific extraction and indexing. In the collection shown above the most common mimetype is application/pdf, with a distribution of subformats from PDF v1.2 through v1.6.

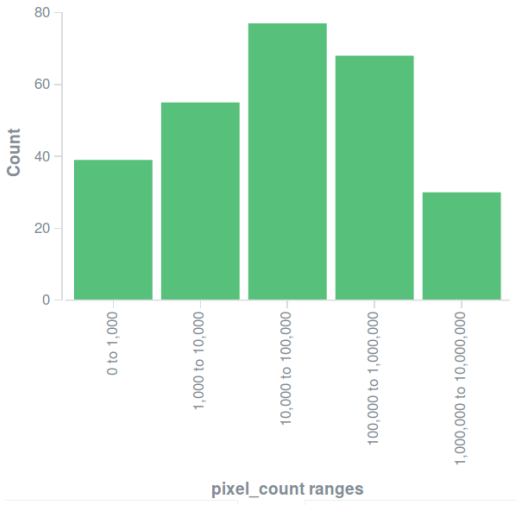

Figure 6: Total Images by Pixel Count (Orders of 10)

6.2 Image Features

Our Elasticsearch cluster includes fields that the DTS has derived from images. Using these metrics we can formulate queries based on image content. For instance we can easily formulate an Elasticsearch query that will find all megapixel images. Below we have a graph showing the numbers of images in a collection falling within pixel count ranges, in orders of ten.

There are a number of other visual features extracted by Brown Dog that may be useful. Visual symmetry is reflected in skewness and kurtosis factors. Human features are tagged and delimited by box regions, including faces, eyes, profiles, and close-ups. Note that this will include both photographs and realistic drawings of people. By indexing these features, we can find all of the images that feature people, or that feature a certain number of people.

6.3 Textual Features

We leveraged both the OCR and format conversion offerings of Brown Dog to acquire as much of the text content from the CI-BER files as possible. The text was recorded in Indigo metadata fields and indexed by Elasticsearch. We have done little beyond a search index with these text fields so far, but we see much more potential for text analysis, now that the text is no longer locked in a file format. For one example, we can find unusual terms to understand how text in one part of our repository is different from elsewhere.

In the table generated by Kibana above there are two rows for every folder, we see that the most unusual terms in U.S. Supreme Court documents are, unsurprisingly, “denied” and “v”, as in “motion was denied” and “Marbury v. Madison”. The OSTP is more concerned with “science” and “budget”, while NIST is more concerned with “specification” and “diagram”. Unusual terms is an especially interesting approach for archival material because it is comparative, showing those terms that are distinctive for each “bucket” within the result set. You can define what your “buckets” are through any other indexed field, be it the folder path, the author, or the year files were created.

The Kibana portal used to create the graphs above exists

parentURI: Descending	Top 2 unusual terms in fulltext	Count
/Archive/ciber/RG 267 - Records of the Supreme Court of the United States/Orders and Journals/www.supremecourtus.gov/orders/courtorders/	denied	606
/Archive/ciber/RG 267 - Records of the Supreme Court of the United States/Orders and Journals/www.supremecourtus.gov/orders/courtorders/	v	670
/Archive/ciber/RG 359 - Records of the Office of Science and Technology/Office of Science and Technology Website/www.ostp.gov/pdf/	science	305
/Archive/ciber/RG 359 - Records of the Office of Science and Technology/Office of Science and Technology Website/www.ostp.gov/pdf/	budget	288
/Archive/ciber/RG 167 - Records of the National Institute of Standards and Technology/Visualization of Structural Steel Product Models, Construction Sites and Equipment, and the Virtual Cybernetic Building Testbed/cic.nist.gov/vrml/cis/lpm6/structural_frame_schema/lexical/	specification	314
/Archive/ciber/RG 167 - Records of the National Institute of Standards and Technology/Visualization of Structural Steel Product Models, Construction Sites and Equipment, and the Virtual Cybernetic Building Testbed/cic.nist.gov/vrml/cis/lpm6/structural_frame_schema/lexical/	diagram	284

Figure 7: Unusual Terms in Folders Containing the Most Texts

as a separate analytics application that is not integrated into the access repository. When we bring these additional search and analytics features into our access repository, they will provide an overview in the context of the collection structure. We will render a dashboard on demand for any folder or collection in the repository, showing analysis of the contents. This brings a pre-made set of relevant analytics into view for every repository user, not just the person crafting the visualizations.

7. DISCUSSION

The workflows described here and their application to appraisal and access are in the early stages. There are several direct steps we will take next to further explore our study.

We have begun to scratch the surface of the data in CI-BER, running the extractors, etc. on a sampling of our collections. However, we have not run the workflow on the bulk of the scientific data in CI-BER, which will pose different challenges and opportunities. The Elasticsearch index and Kibana visualization tool, give us significant analysis features "out of the box" and have promise as an investigative tool for born digital materials, but the dashboards are not integrated into our user-facing access interface. Finally, we can connect repository users to the DTS Clowder item and collection interface, which delivers the complete superset of extracted data for each file, unfiltered by our local repository design and indexing choices. With these straightforward next steps we will improve our understanding of the potential for Brown Dog.

Another avenue to explore is the looping of data through DTS and DAP to extract more knowledge. For instance, we can first convert a document into full text via DAP, then feed the full text into the DTS for all manner of text analysis extractions, including natural language processing to discover dates and the names of people and places. The same text analysis can be applied to OCR text or transcripts extracted from audio. This text mining across diverse formats is hard to achieve traditionally, requiring a dedicated repository and software effort. Within the Brown Dog framework we may be able to bring it within reach of more institutions. A similar combining of Brown Dog services can be used to split out and process sections of files, such as the detailed content items within an MBOX, ZIP or disk image file.

The DTS provides us with metadata in the form of JSON-LD graphs. Presently we only pull certain field values from the JSON-LD, treating it as JSON. A triple store or graph database can be used to index all of the extracted data, from all of the files, in a larger graph. A graph of all of the extracted data opens the door to graph reasoning across the collections. For instance, you might establish that a set of people were working in a team for a time, since they have frequently corresponded or shared authorship on documents. Furthermore, a linked data store allows you to coordinate and query your local data alongside linked data in other places, such as dbpedia[13] One simple example is to link recognized place names with their matching resource in Geo Names. This gives you the ability to query for and index all files that pertain to any level of administrative region on a map. For example a document that mentions "Brooklyn" could be discovered via New York City and New York State.

8. CONCLUSION

We have demonstrated a model architecture, consisting of cloud-based Brown Dog services, Maryland/DCIC middleware, the Indigo repository, and the Elasticsearch applications, that function together at scale to populate the CI-BER collections with enriched metadata records.

We contributed our own extractors to Brown Dog, adding key digital preservation functions. We deployed the Siegfried extractor into the DTS, wrapping the functions of the Siegfried format identification software. While contributing the extractor required programmer effort, the integration of the extracted data into workflows was automatic, as Siegfried's format-related findings merged with the rest of our DTS-supplied metadata. The only change to the Maryland workflow was to decide which Siegfried data to put in the search index. This experience further shows us that Brown Dog is a potent aggregator of extraction and migration tools under one API, capable of multiplying the value of the tool building efforts in the broad data curation community.

Lastly, we find that an enriched supply of metadata directly extracted from digital materials can yield tremendous benefits in the analysis of collections. Data analytics software, such as Kibana, can be used without much domain-specific configuration to gain insight into collection contents. This gives us our first glimpse of what we can do with the expanding workflows and metadata.

Acknowledgments

This research & development has been funded through National Science Foundation Cooperative Agreement ACI-1261582. We would also like to thank the following partners and software projects: Archive Analytics Solutions, Inc. - creators of the Indigo Repository software; University of Maryland Division of IT - system administrator support; Richard LeHane - creator of the Siegfried Software.

References

[1] NCSA Brown Dog. http://browndog.ncsa.illinois.edu/, 2013 (accessed April 21, 2016).

[2] J. R. Heard and R. J. Marciano. A system for scalable visualization of geographic archival records. In *Large Data Analysis and Visualization (LDAV), 2011 IEEE Symposium on*, pages 121–122, Oct 2011.

[13] http://linkeddata.org/

[3] L. Marini, R. Kooper, J. Futrelle, J. Plutchak, A. Craig, T. McLaren, and J. Myers. Medici: A scalable multimedia environment for research. *The Microsoft e-Science Workshop*, 2010.

[4] K. McHenry, R. Kooper, and P. Bajcsy. Towards a universal, quantifiable, and scalable file format converter. *The IEEE Conference on e-Science*, 2009.

[5] K. McHenry, R. Kooper, M. Ondrejcek, L. Marini, and P. Bajcsy. A mosaic of software. *The IEEE International Conference on eScience*, 2011.

[6] J. Myers, M. Hedstrom, D. Akmon, S. Payette, B. Plale, I. Kouper, S. McCaulay, R. McDonald, A. Varadharaju, P. Kumar, M. Elag, J. Lee, R. Kooper, and L. Marini. Towards sustainable curation and preservation: The sead project's data services approach. *Interoperable Infrastructures for Interdisciplinary Big Data Sciences Workshop, IEEE eScience*, 2015.

[7] S. Padhy, G. Jansen, and etal. Brown dog: Leveraging everything towards autocuration. In *IEEE Big Data*, 2015.

[8] Richard Lehane. Siegfried: A signature-based file format identification tool. http://www.itforarchivists.com/siegfried, 2014 (accessed April 21, 2016).

[9] The UK National Archives digital preservation department. PRONOM. http://www.nationalarchives.gov.uk/PRONOM/, (accessed April 21, 2016).