ABCDEF - The 6 key features behind scalable, multi-tenant web archive processing with ARCH: Archive, Big Data, Concurrent, Distributed, Efficient, Flexible

Helge Holzmann[1] , Nick Ruest[2] , Jefferson Bailey[1] , Alex Dempsey[1] , Samantha Fritz[3] , Peggy Lee[1] , and Ian Milligan,[3]

[1] Internet Archive
[2] Digital Scholarship Infrastructure Department, York University
[3] Department of History, University of Waterloo

ABSTRACT

Over the past quarter-century, web archive collection has emerged as a user-friendly process thanks to cloud-hosted solutions such as the Internet Archive's Archive-It subscription service. Despite advancements in collecting web archive content, no equivalent has been found by way of a user-friendly cloud-hosted analysis system. Web archive processing and research require significant hardware resources and cumbersome tools that interdisciplinary researchers find difficult to work with. In this paper, we identify six principles - the ABCDEFs (Archive, Big data, Concurrent, Distributed, Efficient, and Flexible) - used to guide the development and design of a system. These make the transformation of, and working with, web archive data as enjoyable as the collection process. We make these objectives – largely common sense – explicit and transparent in this paper. They can be employed by every computing platform in the area of digital libraries and archives and adapted by teams seeking to implement similar infrastructures. Furthermore, we present ARCH (Archives Research Compute Hub)[1], the first cloud-based system designed from scratch to meet all of these six key principles. ARCH is an interactive interface, closely connected with Archive-It, engineered to provide analytical actions, specifically generating datasets and in-browser visualizations. It efficiently streamlines research workflows while eliminating the burden of computing requirements. Building off past work by both the Internet Archive (Archive-It Research Services) and the Archives Unleashed Project (the Archives Unleashed Cloud), this merged platform achieves a scalable processing pipeline for web archive research. It is open-source and can be considered a reference implementation of the ABCDEF, which we have evaluated and discussed in terms of feasibility and compliance as a benchmark for similar platforms.

[1]https://github.com/internetarchive/arch

JCDL '22, June 20–24, 2022, Cologne, Germany
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9345-4/22/06…$15.00
https://doi.org/10.1145/3529372.3530916

CCS CONCEPTS

• **Information systems → Digital libraries and archives**; **Data extraction and integration**.

KEYWORDS

web archives, big data, data processing, distributed computing

ACM Reference Format:
Helge Holzmann[1] , Nick Ruest[2] , Jefferson Bailey[1] , Alex Dempsey[1] , Samantha Fritz[3] , Peggy Lee[1] , and Ian Milligan,[3] . 2022. ABCDEF - The 6 key features behind scalable, multi-tenant web archive processing with ARCH: Archive, Big Data, Concurrent, Distributed, Efficient, Flexible. In *The ACM/IEEE Joint Conference on Digital Libraries in 2022 (JCDL '22), June 20–24, 2022, Cologne, Germany.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3529372.3530916

1 INTRODUCTION

Web archiving is an important component of modern digital libraries. It is essential for enabling future research into contemporary history and ensuring the long-term preservation of our documentary heritage [11] [3]. Yet while collecting web archive content has matured into a user-friendly process, thanks in no small part to cloud-hosted solutions such as the Internet Archive's Archive-It service, this ease-of-use has not been matched on the analysis side. We accordingly need a user-friendly system that can enable the creation of research datasets from web archives so that researchers can work with material at scale.

In this paper, we present the Archives Research Compute Hub (ARCH), a production system tightly integrated with the Internet Archive infrastructure and services. ARCH grew out of the Archives Unleashed Cloud: a proof-of-concept platform that demonstrated the ability of a web browser-based system to power backend Apache Spark-driven jobs on web archival datasets [12]. Powered by the Archives Unleashed Toolkit and the Internet Archive's Sparkling data processing library, the ARCH platform will become a complementary component of the Internet Archive's Archive-It system. ARCH is built around six key principles: archive, big data, concurrent, distributed, efficient, and flexible. We present these principles as considerations for projects and teams developing similar systems.

2 RELATED WORK AND PROJECT CONTEXT

Established in 2017, the Archives Unleashed project recognizes the collective need among researchers, librarians and archivists for analytical tools, community infrastructure, and accessible web archival interfaces. To this end, the project aspires to make petabytes of historical internet content accessible to scholars and others interested in researching the recent past. Between 2017 and 2020, the project focused on developing the "Archives Unleashed Cloud," a web-based interface for working with web archives at scale using the Archives Unleashed Toolkit and Apache Spark [12]. This work built on the project's long-standing interests in building exploratory search interfaces for web archive collections [8]. Similar noteworthy work includes the SolrWayback project from The Royal Danish Library. Combining Apache Solr with OpenWayback or pywb, SolrWayback provides search and discovery of web archive collections, as well as replay, and a number of analysis and visualization features [10].

In 2020, the project's first phase was completed. The next phase involved exploring integration and collaboration with the Internet Archive [13]. We were influenced by the global adoption of the Internet Archive's Archive-It subscription service and the stability of the Apache Spark platform [6].

Since the launch of the Internet Archive's subscription service in 2006, over 700 institutions from 23 countries have used Archive-It to preserve over two petabytes of data consisting of over 40 billion born-digital, web-published records in over 12,000 public collections. It is a successful service. A survey by the National Digital Stewardship Alliance reported that by 2017, 94% of surveyed institutions were using Archive-It to preserve web material – and an additional 4% were using other services provided by the Internet Archive [9]. Archive-It is thus effectively the de-facto platform for web archiving, used by nearly all Association of Research Library members, hundreds of other higher education, memory institutions, public libraries, governments, and non-profit organizations.

Despite this widely-accepted solution for the capture of web material, the problem of analysis remains. By this, we refer to at-scale explorations of data that require more than the replay interface of the Wayback Machine. While web archive data is captured and preserved in the ISO-standard WARC file format, the formation of a scholarly ecosystem around web archive analysis has been slow.[2]

The Archives Unleashed project aims to address this problem [13] by being for web archive analysis as Archive-It is for web archive capture: powerful, scalable, and above all, accessible and intuitive for users. The Archives Unleashed Cloud (2017-2020) provided user access to the features of the Archives Unleashed Toolkit in a cloud-hosted environment [12]. The Cloud worked with Archive-It collections, using APIs to transfer data from the Internet Archive to Compute Canada cloud-hosted infrastructure. Yet the initial approach of having a separate analysis service presented shortcomings. When a user wished to carry out analysis, data had to be transferred. More importantly, connections between Archive-It and the Cloud required a complicated interplay of APIs, bulk data transfers, and other workflows, leaving a separate analysis service vulnerable to network disruptions or changing standards. These factors combined to make it an interesting proof-of-concept but one that presented considerable sustainability challenges.

Our goal, then, was to integrate Archives Unleashed tools with the Internet Archive's Archive-It service. As a novel collaboration, this would require new approaches to building at-scale infrastructure.

[2]The best place to learn about available tools is the "Web Archiving Awesome List" maintained by the International Internet Preservation Consortium and researchers across the field. See https://github.com/iipc/awesome-web-archiving.

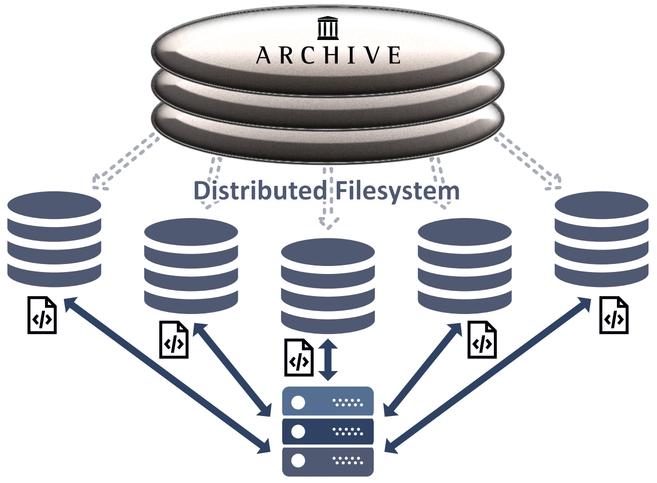


Figure 1: Archive-powered computing.

3 THE "ABCDEFS" OF DIGITAL LIBRARIES AT SCALE

When starting a new project, a learner needs to understand their "ABCs." As we began to develop the at-scale implementation, we realized that a basic understanding of web archiving and data processing on their own was insufficient. Therefore, we present our next steps – the "ABCDEF"s – which are based on intense requirement engineering upfront and informed by user experience surveys and extensive prototyping. While "ABCDEF" is primarily designed as a helpful mnemonic, the lessons learned here will be broadly applicable to other at-scale processing pipelines throughout the digital libraries field.

3.1 Archive

The main data source in large-scale computing infrastructures of libraries and archives is the digital data repository of those institutions. In the case of born-digital organizations, such as the Internet Archive, it is the digital library or digital archive itself. Hence, these data lakes should be considered the backend that power such infrastructure in the first place. However, since access to those long-term preservation systems is commonly slow, additional access and caching layers are required to achieve efficiency.

At the same time, big data computing applications operating on this data should not need to deal with these additional layers. Rather, they should work with the archival data as their primary data source. It should be the job of the computing platform to abstract away the intermediate layers and allow for seamless data access to archival, long-term preserved data.

Data locality is another aspect to consider in large-scale computing systems, which usually run on distributed clusters consisting of multiple machines to enable parallel computing. That is the mechanism to move the code, which is smaller than data, to the data, which in turn can stay and does not need to be moved. While this is a common paradigm [5] and easily applicable in clusters that are used for both storage and computation, this is not the case in library and archive environments. There, digital preservation systems constitute the primary data source, independently of the computing infrastructure. Even though those preservation systems may be distributed across multiple machines, they're widely not designed and/or well-suited to run computations.

Thus, the archival processing system should bring the code and computation as close to the data as possible. In the architecture, as described above and depicted in Figure 1, that is the hidden layer that serves as the primary data source for external collections and a hot cache for otherwise cold long-term preservation storage.

3.2 Big Data

Big Data is widely characterized by some number of Vs. Consider three of them: Volume, Variety, Velocity, which all naturally intersect in digital libraries and archives, and in particular, web archives. Web data grows as fast as the web does, and the web is constantly evolving. The vast amount of URLs on the web - along with their dynamic and temporal aspect in web archives - lead to enormous volumes and a variety of data types inherent to the web, including text, images, videos, code, styles, and the like.

Nevertheless, web data and web archive data are unique in many ways. Compared to more traditional Big Data, this is mainly due to its heterogeneity and lack of a natural order. In the world of web archives, every archival record looks the same because the WARC records [4] present a standard structure: headers and properties, along with easily parsable metadata. What's in a WARC record is never the same, though, making the data very heterogeneous. Compounding this, although their URLs give a logical structure, websites can have very different graph structures, some are formed as a tree, while others exhibit a mesh, and the order across sites is widely illogical. Eventually, crawlers preserve the web as they hit the pages, controlled by their own prioritization and scoping strategies [2] [1].

All of this requires advanced selection, filtering and sampling mechanisms, performed by pre-processing steps prior to the actual work. Given the sheer amount of data in such web archives, even by building sub-collections, filtering temporarily through metadata facets, and sampling down based on certain conditions, the result is often too big to work with locally.

The solution to this is derivation. While the above tasks, which can be considered horizontal operations, reduce the number of input records, as shown in Figure 2, by performing derivation — a vertical operation — only a task-specific essence of the records is extracted, kept, or even generated. Common examples are named entities mentioned in a corpus (i.e., persons, locations, organizations) or hyperlinks extracted from web pages. The combinations of the above filtering steps and the final derivation, result in more compact datasets that users can more comfortably deal with. Big Data is thus made more manageable.

3.3 Concurrent

A platform hosted on central servers and not running as an individual local instance can be used by multiple users concurrently. That means multiple users run the same tools, use the same hardware, and potentially operate on the same data simultaneously. This requires both capable hardware and coordination. For instance, a number of users launching a derivation job at the same time should not overload resources or memory. Scheduling systems, similar to those found in every computer's operating system, are required to control when a job can run or should be queued for later execution if there's no free capacity.

At the same time, concurrent systems should not be blocking. A data processing job that uses only resources on the distributed cluster nodes must not block post-processing jobs that could run on local server resources in parallel. Such behavior requires explicitly-defined job types, which let the system know what kind of job it will run or queue up. Multiple, disjunct queues per type allow a job of a type that's ready to run to bypass others that are queued on busier queues. Furthermore, complex jobs consisting of consecutive phases, such as (pre-)processing and post-processing, may be designed as multiple chained jobs to use such an architecture efficiently. Similarly, certain jobs, such as examples running on smaller sample inputs, could be prioritized using dedicated queues as they can finish quicker with little impact on the runtime of other, bigger jobs waiting for free resources.

As shown in Figure 3, this relatively complex task of coordinating and monitoring is the job of a central job manager. This can be considered the core component of every multi-tenant data processing system. While processing resources may be fully employed by running derivation jobs, it is essential to ensure the job manager remains responsive. Job managers also supply interfaces - computer-readable APIs and (graphical) user interfaces - with reports that identify the system's current state and currently running and launched derivation jobs. These ultimately convey the expected start and run time of a user's derivative.

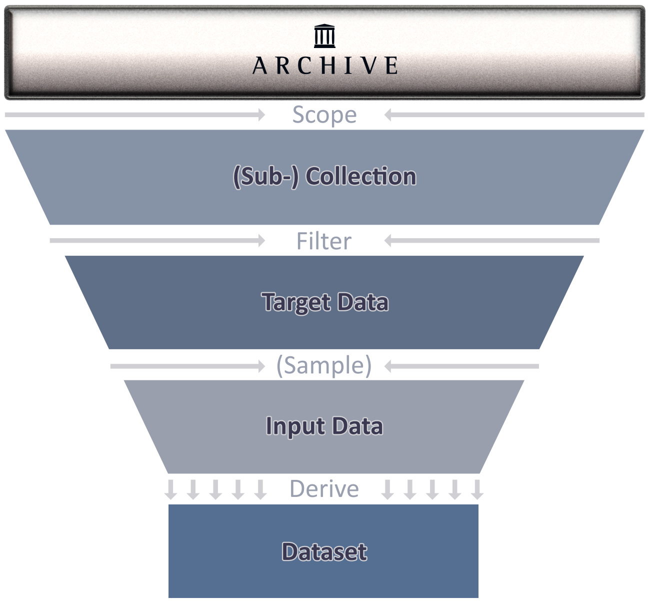


Figure 2: Big Data derivation.

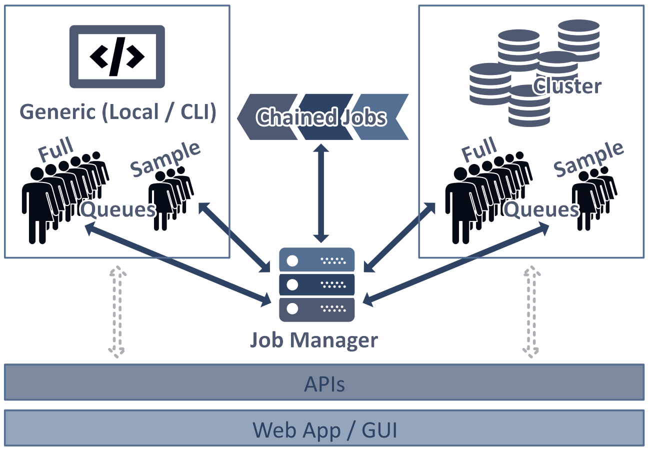


Figure 3: Concurrent job architecture.

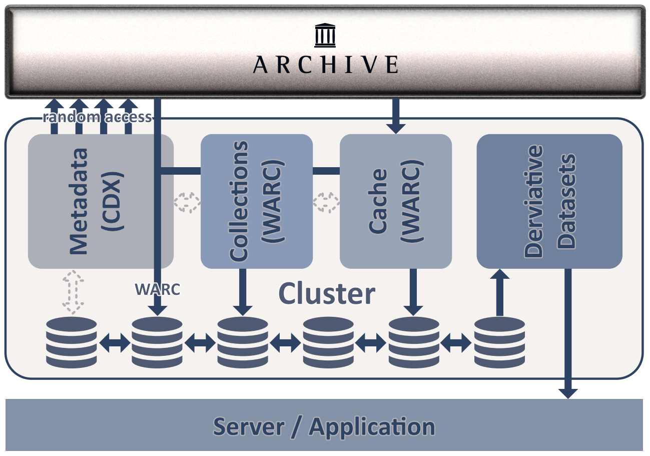


Figure 4: Distributed data workflows.

3.4 Distributed

In summarizing the previous sections (**A**rchive, **B**ig Data, **C**oncurrent), we see that these concepts are based on the idea of a distributed architecture, whether for storage, scheduling or processing. Hence, distributed design is a fundamental principle of every library or archival processing infrastructure and should be an objective in all aspects of our work. Distributed data storage and inherent parallel, distributed processing is driven by the question of how to split up and distribute data, processes, and tasks.

Today's de-facto standard to work with computer clusters are the various Apache projects around Hadoop, such as YARN, MapReduce, Spark, and related tools. Storage is commonly powered by Hadoop's Distributed Filesystem (HDFS). HDFS organizes the way it stores files across multiple machines largely by itself, with files split into blocks, which are stored replicated on different machines and racks. This way, data locality can be exploited by running the code to process a file on the storage node that stores large portions of that file. In a library and archive setting, the same approach works well for all collections stored outside the digital library or archive, such as custom collections loaded into the platform, as well as cached collections copied from the long-preservation archival backend system onto the cluster to be processed.

Additionally, derivative datasets extracted or generated from the raw input data in a derivation job are immediately written to the distributed filesystem and automatically split into blocks as described above. To access such output files for in-browser preview and visualization or to be downloaded by users, they have to be streamed from distributed storage, with all blocks of that file being concatenated on-the-fly through a central endpoint.

For sub-collections or filtered sub-sets, an efficient approach to distributing and processing web archive data and other archival datasets have been shown by ArchiveSpark [6] [7]. Instead of parallelizing datasets in a distributed setting through files or blocks, metadata records can be used, which, in the case of web archives, are the crawl index records (CDX). This, as well as the above described data flows, is illustrated in Figure 4.

3.5 Efficient

Efficiency is paramount when it comes to data processing. High memory consumption occupies resources, reduces parallelism and leads to failures if limits are reached. Therefore, it is essential to keep memory consumption as low as possible to provide a reliable and robust process. Records can vary considerably in size in a library or archive environment, given their heterogeneous character (as elaborated in **B**ig Data). At the same time, they can be very large: input is not usually structured data but rather raw digital objects, such as books, images, or in the case of web archives, web resources (which can be pages, images, videos, and others). Loading and processing these fully in memory easily results in memory overruns. These must be avoided. As it is often not evident before the data is read how much remains, there are a few strategies that can be adopted to prevent memory issues and ensure an efficient and stable system:

- Use available headers and/or metadata records (see the ArchiveSpark approach above, under **D**istributed) to pre-validate data prior to access. For example, one can check content length against type. Raw webpages without embeds usually have sizes measured in kilobytes. If they are over a given threshold, such as one megabyte, the system can consider them invalid and discard.

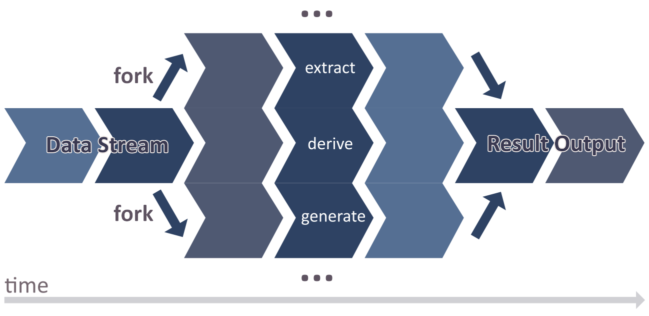


Figure 5: Efficient, parallel stream processing.

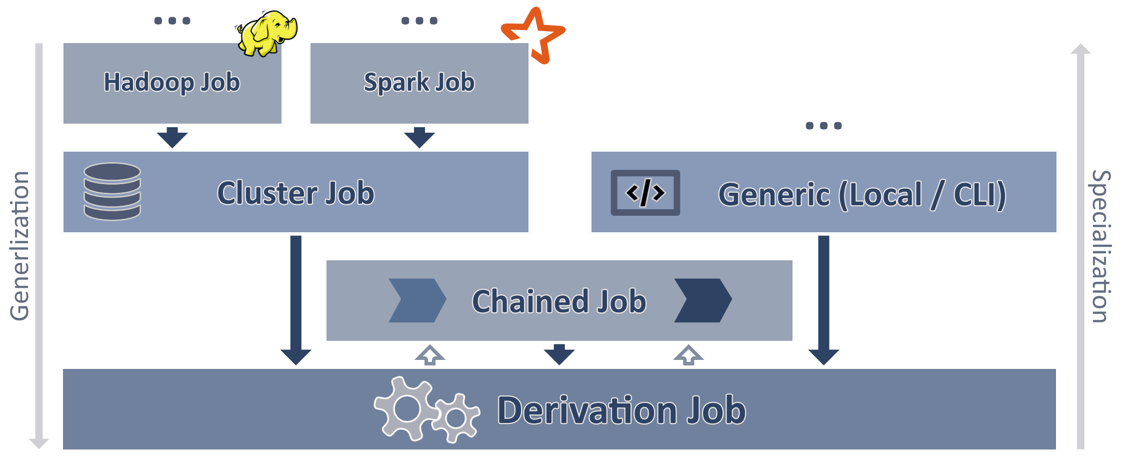


Figure 6: Flexible abstraction layers.

- Skip over invalid and irrelevant/filtered records, rather than reading them with no operation. This prevents avoidable reads and filling up memory buffers.
- Stream input data and processes on the fly, while keeping only what is absolutely necessary in memory.
- Fork streams to process them in parallel with multiple derivation tools in one run, as depicted in Figure 5.
- If necessary, retain derivatives as the relevant essence of records. These are only a fraction of the size of full records, as suggested above under **B**ig Data.
- As a last resort, streams should be bounded when reading single primitive values, such as strings, to not have them overflow memory, e.g., because of malformed records.
- Write derived results to disks as early as possible and release them from memory if no longer needed.

3.6 Flexible

Future-proof systems should be modular and flexible to allow for switching outdated tools, to plug in novel technology, and to adapt new data and derivation types. They should be easily extensible without restricting themselves too much. This can be achieved through decoupled abstraction layers with clear responsibilities and well-defined interfaces. At the same time, it is important for module and layer interfaces to be lean, reduced to a minimum, and as generic as possible. Crucially, it is unknown what the future will bring, though it should fit into the same system's specifications.

To re-use code and modules or create more specific interfaces for tools that share functionality, it helps to introduce additional, higher-level layers and develop tools against these, while other, more distinct tools can be developed against less-specific, lower-level layers. An example of this is found in the different job types (see **C**oncurrent) as well as de-facto standard technology for working with computer clusters as listed above (see **D**istributed). All supported job types should the same at their core to be manageable by the central job manager and support basic operations, such as queueing, running, and retrieving results. However, all jobs that use the same technology (e.g., Apache Spark) may support more specific operations such as launching a new "Spark Context," which is Spark's runtime environment. Hence, shared Spark-specific code can live in a high-level abstraction layer on top of the generic job interface without limiting the system's functionality by making it tailored to Spark jobs only.

Furthermore, as described under **C**oncurrent, chained jobs should be jobs in and of themselves. Hence, they should implement the same interface and provide another abstraction layer with the same interface to plug-in downstream jobs as children. Figure 6 depicts a reference design for such an architecture. However, this basic structure is subject to change, evolving with future developments in an agile process.

4 IMPLEMENTING ABCDEF WITH ARCH

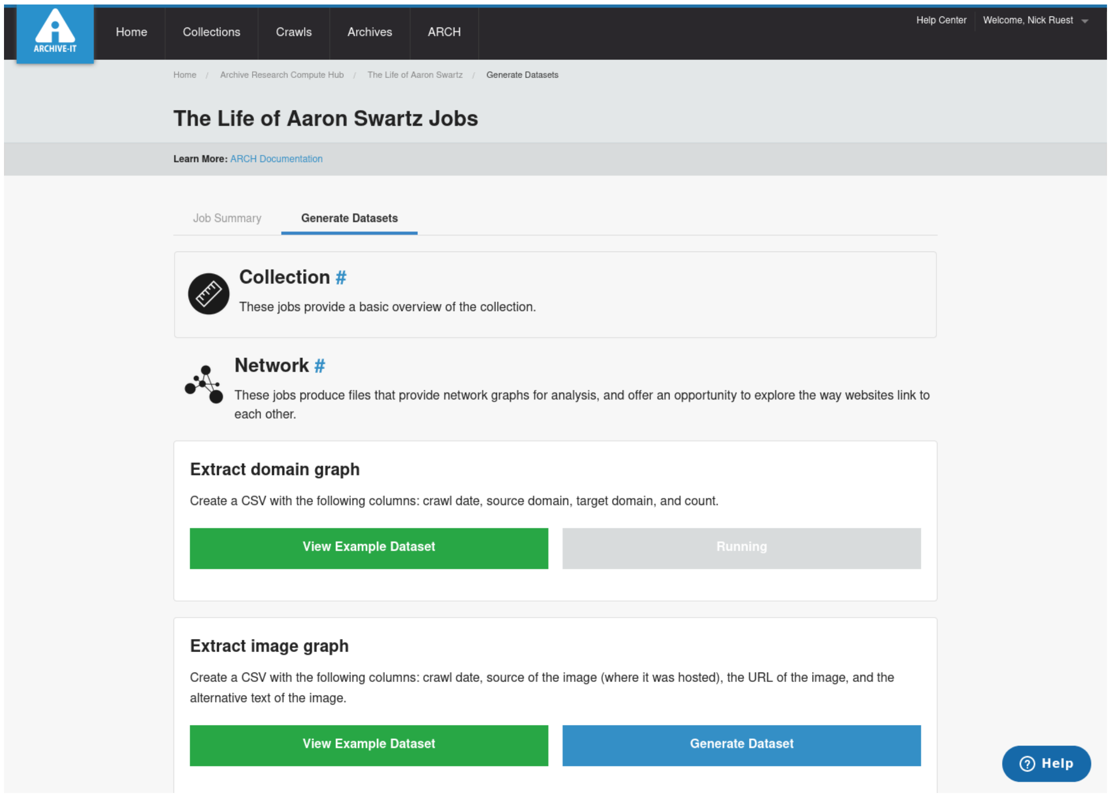
ARCH represents our attempt to incorporate the ABCDEF principles. Inspired conceptually by the earlier Archives Unleashed Cloud, ARCH has been redesigned from scratch to meet the ABCDEFs.

In this section, we present our interface and its broader context within Archive-It. ARCH allows users to take their Archive-It collections - or other collections they have been granted access to - and create their own derivative datasets. They do so by selecting their collection, navigating a list of available derivatives datasets to generate (seen in Figure 7), and then generating those derivatives datasets for either in-browser limited exploration or downloading them for work on local infrastructure.

4.1 Archive-It Research Services

ARCH integrates with the Internet Archive's existing research services. In 2015, Archive-It launched Archive-It Research Services (ARS), a service to provide Archive-It institutions with the ability to generate several derivative datasets from their web archive collections in Archive-It. ARS helped establish the groundwork for providing data-driven access services.

ARS featured three datasets available to users: WAT, WANE, and LGA datasets. WATs are effectively WARC files with the "payload" (i.e. the content) stripped out but metadata intact. WANE files are named entities (people, places, and organizations). LGA files are link graph files that provide information on which records link to other records within a collection. Users can click a button in



The Life of Aaron Swartz Jobs

Learn More: ARCH Documentation

Collection #

These jobs provide a basic overview of the collection.

Network #

These jobs produce files that provide network graphs for analysis, and offer an opportunity to explore the way websites link to each other.

Extract domain graph

Create a CSV with the following columns: crawl date, source domain, target domain, and count.

Extract image graph

Create a CSV with the following columns: crawl date, source of the image (where it was hosted), the URL of the image, and the alternative text of the image.

Figure 7: The "generate datasets" page in the ARCH interface.

the Archive-It interface to generate these datasets, which would prompt an Archive-It web archivist to process the request. This is a labour-intensive approach. While a good first step, a more robust, self-service and at-scale opportunity that translated files into more standardized derivative datasets was necessary.

4.2 Current Integration

As of December 2021, ARCH has both feature parity with the earlier Archives Unleashed Cloud, and also additional functionality to generate several additional datasets. As functionality from the earlier Cloud was ported, all features were redesigned and reimplemented with the ABCDEFs in mind. By reimplementing features with this architecture, we addressed known issues, fixed existing bugs, and more importantly, implemented an approach that scales to meet our needs.

ARCH now runs on an infrastructure that is physically connected to Archive-It servers and computing infrastructure, mitigating the need to copy data before processing. As not all Archive-It data is kept in its dedicated computing cluster, ARCH is connected to the Internet Archive's long-term storage system (the "Petabox") to fetch missing data. In addition, we implemented a smart caching mechanism to avoid re-fetches for consecutive access to the same data. Cognizant of researcher needs beyond Archive-It collections, we also support custom collections which can be located on ARCH's own cluster.

Given the sensitive nature of web archival collections, we have implemented a user and permissions system. There are two authentication providers: Archive-It user accounts and dedicated ARCH users. For Archive-It users, we rely on Archive-It's internal permissions process. We have also implemented a permission control access that allows ARCH and Archive-It users to cross-access additional Archive-It collections (pending permission from the data collector) and ARCH custom collections.

The core of ARCH is its job processing module. This supports different dataset generation jobs based on a generic interface to start jobs, monitor their status, and explore the ensuing output (seen in Figure 7). We currently provide implementations for Spark jobs and general-purpose command-line instructions, which can be chained for pre- and post-processing pipelines.

Finally, ARCH offers type-specific visualizations for its sixteen job types in four different categories along with output-specific previews in the browser. An example of this for the domain frequency dataset can be seen in Figure 8.

To control jobs and enable the downloading of files via different tools (browser-based downloads for smaller files, command line for larger ones), we provide multiple APIs and authentication methods. While the actual implementation details are beyond the scope of this paper, ARCH is a native Scala application built using Scalatra[3]. The underlying toolkit is based on the Archives Unleashed Toolkit (previously known as Warcbase) as well as the Internet

[3]https://scalatra.org/

Archive's Sparkling library[4]. Jobs and queues are controlled via APIs, enabling Spark jobs to be chained with post-processing jobs, as well as separate queues for example/full jobs, Spark operations, and post-processing.

4.3 User Interface

ARCH's interface consists of four levels. These guide users to interact with their collections by generating datasets for analysis and engaging with in-browser features. The goal of ARCH is to provide an efficient, streamlined workflow without burdening users with computing requirements or actions.

The first level is the **main collections** page. All of a user's Archive-It collections are presented in a table (Figure 9), accompanied by information about the most recent analysis conducted and other collection-based metadata. Each collection title provides an access point for conducting analysis. The second is a **job summary** page, where users can generate, download, and monitor derivative datasets. An overview of the collection identifies basic metadata about the collection, including collection size and whether it is a public or private collection. The second main feature of this space provides tables that summarize "Jobs in Process" - the stage and queue of any current jobs being run - and a "Completed Jobs" table identifying all datasets previously generated, noting an accompanying date/time stamp (Figure 10).

The third level is the **generation of datasets** (Figure 7). As a core feature of ARCH, users can generate sixteen different datasets for scholarly exploration. These datasets are categorized into four main themes of analysis (Table 1).

Finally, the last level are the **derivative dataset** pages themselves. For each dataset generated, users can access an overview page of the dataset, which provides metadata (file name, file size, results count, and date completed), download options, a preview of up to 100 lines, and the option to re-run any job. An example of this can be seen in Figure 8. Where possible, in-browser visualization and charts present a summary of the data. For instance, the *extract web graph* dataset page offers an interactive network graph that users can explore using simple functionalities like zooming in and out on modes and clusters and exporting a high-resolution image. These datasets are intended be downloaded and further explored with other analytical tools and methods.

[4]https://github.com/internetarchive/Sparkling

Dataset Category	Description
Collection	Offers an overview of a collection by looking at simple statistical counts.
Network	Produces files that provide network graphs for analysis and offer an opportunity to explore the way websites link to each other.
Text	Allows the user to explore text components of a web archive, including extracted "plain text" HTML, CSS, and other web elements.
File formats	Provides files that contain information on certain types of binary files found within a web archive.

Table 1: ARCH Datasets

4.4 User Evaluation

The design process for ARCH involved a variety of interconnected stages, from designing wireframes to building infrastructure to connecting backend processes to the user interface. User experience (UX) evaluations were essential for measuring and understanding the needs of researchers. As such, the team conducted iterative and multi-staged user testing and surveying to assess user needs and experience. By engaging with Archive-It power users and Archives Unleashed Cloud alumni in five closed user testing rounds, our team gathered feedback and initial impressions of ARCH. Testing was primarily conducted through surveys, which collected qualitative and quantitative data to determine user satisfaction and experience.

Findings from the survey were translated into actionable tickets to provide action-based tasks for development cycles. We were able to implement the majority of action items, with some needing further planning and only a few that fell outside of our scope of work.

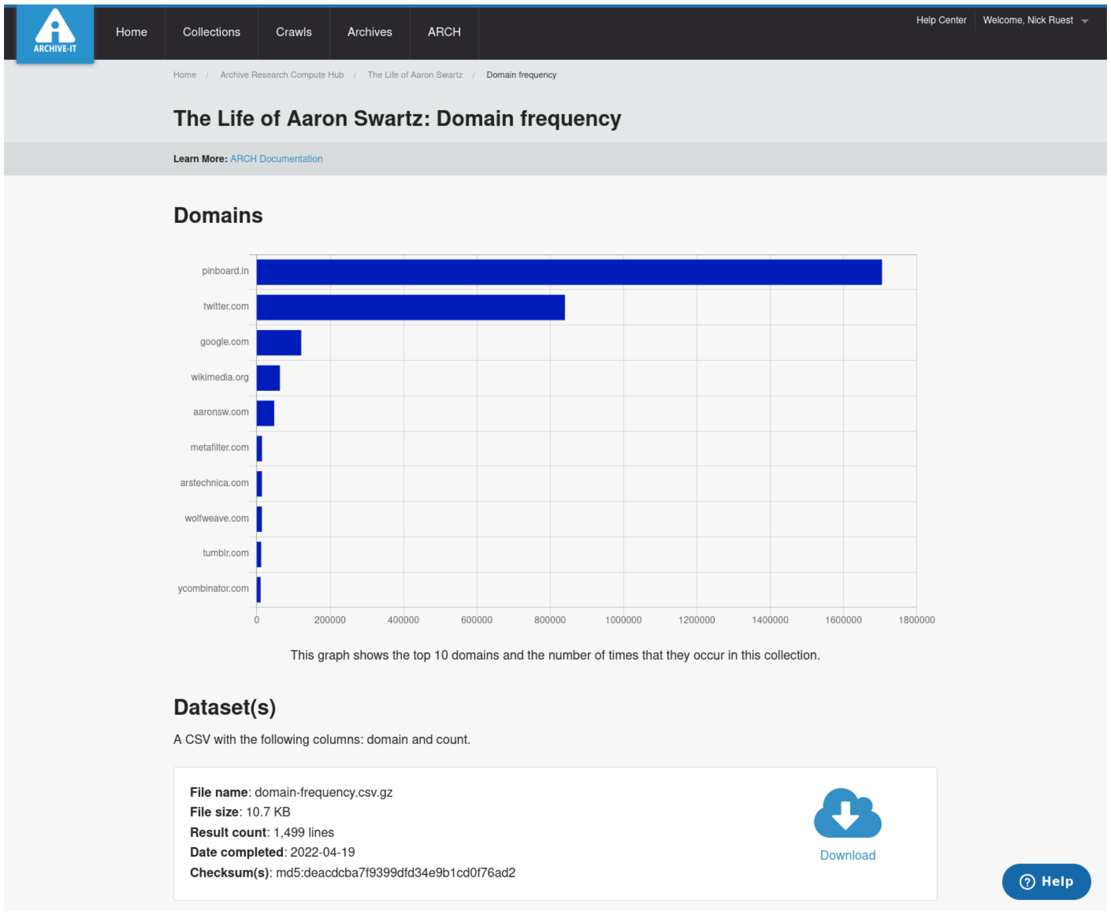
As a multi-stage UX testing process, each subsequent round of testing served as another opportunity to review and refine impressions of prior development and enhancements — improving our accuracy and capacity to match user needs at each stage. Our final rounds of testing concluded in early 2022. This final process served two purposes. First, we expanded testing to include a larger group (approximately 100 participants) to serve as a stress test. As this was our largest testing group to date, this offered an opportunity to verify ARCH's robustness, capacity, and efficiency while noting any bottlenecks or areas for improvement. Second, we conducted focused interviews with a small group of researchers who have extensively used ARCH since August 2021. These researchers were ideal for understanding the real-life application and use cases of the web archives research journey.

5 REFLECTIONS

In implementing ARCH, we focused on six key principles - discussed extensively above - that have driven design choices and been verified through user studies. As a quantitative evaluation is difficult for such objectives, we assess ARCH's level of compliance to these in the following qualitative evaluation.

5.1 Current State

ARCH has been designed as an integrated component of Archive-It. Although it has been implemented as a separate, independent app, it follows Archive-It's design guidelines and workflows, and natively supports Archive-It's user accounts and collections by deeply integrating its APIs. Data access from ARCH to these files makes use of this replication. Its computing cluster runs within the Internet Archive's local network and is physically connected to both PetaBox and Archive-It's cluster nodes. ARCH's access strategy tries to first access files through the foreign cluster's HDFS (Hadoop Distributed File System) over the network. If a file is not present, it exploits the direct access to HDFS and re-fetches a file to its own HDFS, where it is cached medium term, to be available for consecutive processing jobs using the same data. This close integration of ARCH with the Internet Archive and Archive-It fulfills the first and most basic principle of the ABCDEFs: it's backed by a digital **A**rchive.



The Life of Aaron Swartz: Domain frequency

Learn More: ARCH Documentation

Domains

This graph shows the top 10 domains and the number of times that they occur in this collection.

Dataset(s)

A CSV with the following columns: domain and count.

File name: domain-frequency.csv.gz
File size: 10.7 KB
Result count: 1,499 lines
Date completed: 2022-04-19
Checksum(s): md5:deacdcba7f9399dfd34e9b1cd0f76ad2

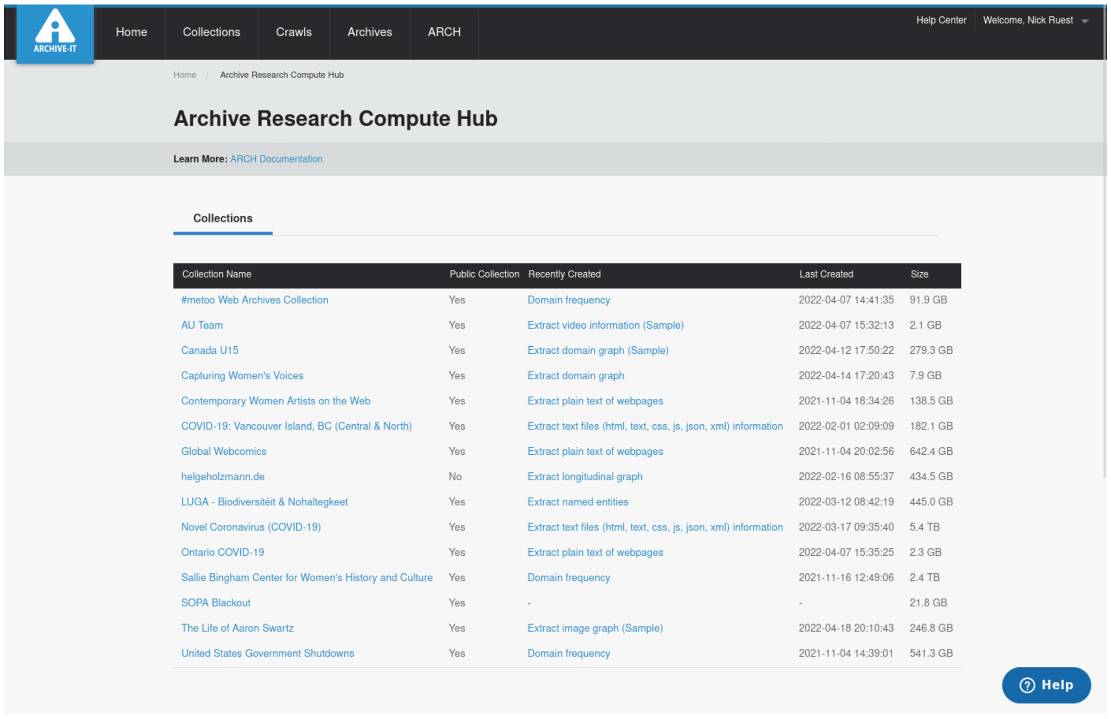
Figure 8: One of ARCH's Dataset Results Pages.

Looking at our **B**ig Data definition, a crucial principle is the derivation of data from difficult-to-handle big collections. This is the main purpose of ARCH, as described under *Current State* and guided by its user interface (see *User Interface*). Users start with one of their full collections, a sub-collection of the Internet Archive's full web archive, also known as Wayback Machine, and select a dataset, before ARCH derives the desired information from the raw archival records to a smaller, derived, well-structured dataset. In addition, ARCH runs derivation jobs on a smaller sample of a given collection. For this process, a conditional algorithm was developed to efficiently identify a sample, in which each specified condition has to be met at least once.

A typical example of such conditions is data types. For derivation jobs that aim to extract multiple data types of records, each of these types should be included in the sample at least once in order to have at least one example for every type in the preview. ARCH reads a limited, pre-defined number of records from a dynamic set of partitions of the distributed input dataset, in parallel, checking every read record against the conditions and logging fulfilled ones. If not all of them have been met, the set of partitions to be included grows by fixed factor and another round of parallel probing starts. Finally, from all checked partitions, the minimum set to fulfill all or most conditions gets picked, and the candidate records from these are selected as a sample in a highly efficient process.

In combination with job architecture, the central job manager has been implemented according to the design presented through the **C**oncurrent principle. ARCH has been planned and developed as a multi-tenant system with concurrency in mind from the beginning. It currently features four job queues in total, two for the currently supported job types, Spark and generic jobs, and two of each of those for example and full runs. All of the currently integrated jobs are based on Spark, which is used for the main derivation tasks on our cluster in a distributed, parallel setting. Generic jobs are simpler and can run any code or command-line instruction. They are commonly used for post-processing the resulting dataset, packaging them and making them ready for download by users. The two job types are combined in chained jobs, which are registered in the central job manager and presented as one job to the user, but get queued and executed consecutively.

As mentioned, we make heavy use of Spark as our main driver, running on a Hadoop cluster, to handle the typically large web archive collections we have to deal with. **D**istributed design is key in our caching and short-term storage infrastructure as well as ARCH's parallel jobs to gain the efficiency our users expect. The



Archive Research Compute Hub

Learn More: ARCH Documentation

Collections

Collection Name	Public Collection	Recently Created	Last Created	Size
#metoo Web Archives Collection	Yes	Domain frequency	2022-04-07 14:41:35	91.9 GB
AU Team	Yes	Extract video information (Sample)	2022-04-07 15:32:13	2.1 GB
Canada U15	Yes	Extract domain graph (Sample)	2022-04-12 17:50:22	279.3 GB
Capturing Women's Voices	Yes	Extract domain graph	2022-04-14 17:20:43	7.9 GB
Contemporary Women Artists on the Web	Yes	Extract plain text of webpages	2021-11-04 18:34:26	138.5 GB
COVID-19: Vancouver Island, BC (Central & North)	Yes	Extract text files (html, text, css, js, json, xml) information	2022-02-01 02:09:09	182.1 GB
Global Webcomics	Yes	Extract plain text of webpages	2021-11-04 20:02:56	642.4 GB
helgeholzmann.de	No	Extract longitudinal graph	2022-02-16 08:55:37	434.5 GB
LUGA - Biodiversitéit & Nohaltegkeet	Yes	Extract named entities	2022-03-12 08:42:19	445.0 GB
Novel Coronavirus (COVID-19)	Yes	Extract text files (html, text, css, js, json, xml) information	2022-03-17 09:35:40	5.4 TB
Ontario COVID-19	Yes	Extract plain text of webpages	2022-04-07 15:35:25	2.3 GB
Sallie Bingham Center for Women's History and Culture	Yes	Domain frequency	2021-11-16 12:49:06	2.4 TB
SOPA Blackout	Yes	-	-	21.8 GB
The Life of Aaron Swartz	Yes	Extract image graph (Sample)	2022-04-18 20:10:43	246.8 GB
United States Government Shutdowns	Yes	Domain frequency	2021-11-04 14:39:01	541.3 GB

Figure 9: ARCH main collections page.

previously described sampling algorithm has been specifically designed for such distributed datasets. Without these ingredients, we would not be able to process data at the scale we currently do. At the same time, they complicate the development and require thorough planning, as debugging such a system, running on multiple machines, can quickly become very complex. This has been an area of engineering and implementation focus. Aware of the extensive memory consumption of the underlying code base and related issues, which caused numerous failed jobs in the previous Cloud platform - issues that remained unresolved as we started the work on ARCH.

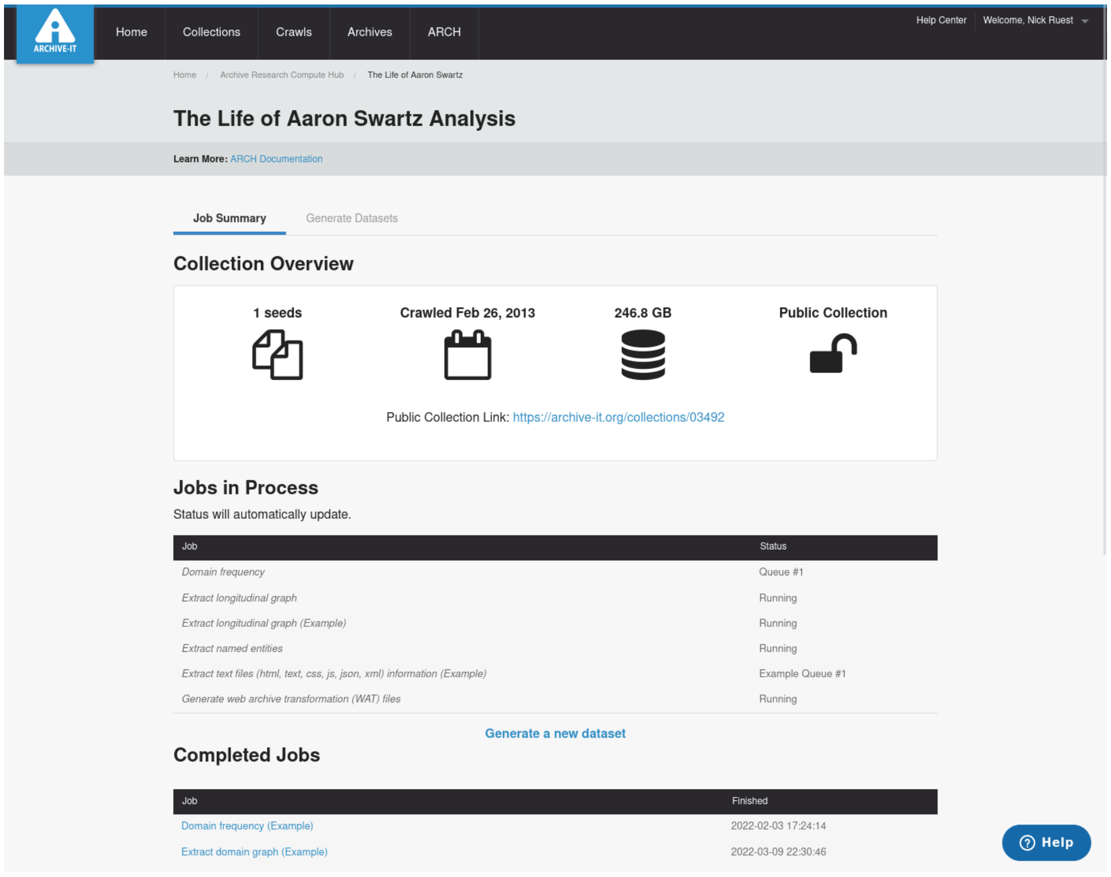
Efficiency is difficult to measure given the very different preconditions and hardware setup of the earlier Cloud and the current ARCH. However, we reduced the legacy issues around high memory consumption of the Cloud during our redesign. Finally, we've reduced the memory consumption of ARCH to a fraction, roughly by a factor of 10, while keeping a high level of efficiency and stability. As demonstrated through extensive testing and large user studies, our jobs are robust. Furthermore, new efficiency of memory-related issues were relatively straightforward to fix by applying the strategies identified above under the *Efficiency* principle.

Finally, **f**lexible design is a high priority in all of our architecture. We have employed an agile development process in which we regularly reviewed code, permanently refactored existing code to combine duplicate codes into additional abstraction layers, introduced shared modules, and streamlined unnecessary complexity. Particularly for derivation jobs and corresponding data inputs, we provide lean interfaces to plug in new types directly compatible with ARCH's job management system, including queuing, visualization and user access. As an example, all jobs based on the old Archives Unleashed Cloud and Toolkit, which are very similar, derive from the same superclass with a specialized interface and shared code specifically for these jobs. However, this layer sits on top of the Spark job layer, which implements the generic job interface. So, new jobs can plug-in either depending on their underlying technology or even introduce a new type, which, by implementing a very lean interface, will be manageable by the ARCH's job management system.

5.2 Outlook and Next Steps

In this outlook, we reflect on those features and improvements that directly address the six key aspects this paper presents. While the customization and configuration of ARCH have not yet been a focus, it will become a priority in the next phase of this project as it touches on all six ABCDEF principles. The most awaited features under this theme are the customization of collections by specifying filters to derive sub-collections and the customization and configuration of derivation jobs through parameterization, for instance, the definition of tags for HTML extraction.

As we enter the final stage of user testing, we will launch ARCH as a public Minimum Viable Product (MVP) to Archive-It subscribers. Final development activities will prioritize user-defined queries, which will ultimately allow users to generate smaller and more manageable subsets within a collection. Many Archive-It collections are large: for instance, some of the earlier referenced researchers are using the International Internet Preservation Consortium's global Coronavirus web archive. While an exciting collection,



The Life of Aaron Swartz Analysis

Learn More: ARCH Documentation

Collection Overview

1 seeds | Crawled Feb 26, 2013 | 246.8 GB | Public Collection

Public Collection Link: https://archive-it.org/collections/03492

Jobs in Process

Status will automatically update.

Job	Status
Domain frequency	Queue #1
Extract longitudinal graph	Running
Extract longitudinal graph (Example)	Running
Extract named entities	Running
Extract text files (html, text, css, js, json, xml) information (Example)	Example Queue #1
Generate web archive transformation (WAT) files	Running

Completed Jobs

Job	Finished
Domain frequency (Example)	2022-02-03 17:24:14
Extract domain graph (Example)	2022-03-09 22:30:46

Figure 10: ARCH job summary page.

at 5 TB it is on the upper side of a collection and means generated derivatives, such as the full-text, can still be in the hundreds of gigabytes.

Accordingly, we are developing a system that will enable users to filter these large collections into usable subsets. For example, users might want to filter collections on facets including date, keyword, or domain. The large collection might then be broken down into a "collection of all webpages containing the word 'Canada'," "a collection of webpages from March 2020," or "all of the webpages from `https://www.who.int`."

These features are under development and will be released in the near future. They will help to tailor the archival datasets to the user's needs (**A**rchive) and make datasets more specific, less arbitrary, with more relevant data for downstream application and hence, facilitate manageability of the dataset (**B**ig Data). While existing jobs have been implemented alongside ARCH job manager, which ensures fair, concurrent processing, introducing customized jobs and tailored datasets to this interface will be one of the biggest challenges (**C**oncurrent and **F**lexbile). In terms of data distribution for custom sub-collections, it is crucial that candidate records are efficiently selected and filtered from their source datasets. As sub-collections may be very small compared to the original collection, we want to avoid full scans of those by skipping irrelevant records. This is where metadata records and random-access will come into play to achieve efficiency in working with archival subsets, as sketched above (**D**istributed and **E**fficient).

6 CONCLUSION

We have presented ARCH, the Archives Research Compute Hub, along with six key principles (ABCDEF) for library or archive-powered computing infrastructures that have fundamentally driven our development and design. The evaluation has shown to what extent the system follows the guidelines we have formulated as part of these six objectives. Finally, it has been shown that ARCH successfully reaches those self-set requirements, making it a reference implementation for such a platform in the area of web archiving.

This will help us further meet the principles we've established as the ABCDEF of Digital Libraries at Scale. In particular, we continue our "Big Data" approach by making derivatives even more usable and tailored to our users. Similarly, we increase the "efficiency" by allowing people to work with more efficient and bounded datasets.

ACKNOWLEDGEMENTS

This research was supported by the Andrew W. Mellon Foundation's Public Knowledge program, the Social Sciences and Humanities Research Council of Canada, as well as Start Smart Labs, the University of Waterloo, and York University.

REFERENCES

[1] Internet Archive. 2021. Archive-It! https://archive-it.org/

[2] Internet Archive. 2021. heritrix3. https://github.com/internetarchive/heritrix3

[3] Niels Brügger. 2018. *The Archived Web: Doing History in the Digital Age.* MIT Press.

[4] International Internet Preservation Consortium. 2021. The WARC Format. https://iipc.github.io/warc-specifications/specifications/warc-format/warc-1.1/

[5] Jens Dittrich and Jorge-Arnulfo Quiané-Ruiz. 2012. Efficient big data processing in Hadoop MapReduce. *Proceedings of the VLDB Endowment* 5, 12 (2012), 2014–2015.

[6] Helge Holzmann, Vinay Goel, and Avishek Anand. 2016. ArchiveSpark: Efficient Web Archive Access, Extraction and Derivation. In *Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries*. ACM, Newark New Jersey USA, 83–92. https://doi.org/10.1145/2910896.2910902

[7] Helge Holzmann, Vinay Goel, and Emily Novak Gustainis. 2017. Universal distant reading through metadata proxies with archivespark. In *2017 IEEE International Conference on Big Data (Big Data)*. 459–464. https://doi.org/10.1109/BigData.2017.8257958

[8] Andrew Jackson, Jimmy Lin, Ian Milligan, and Nick Ruest. 2016. Desiderata for Exploratory Search Interfaces to Web Archives in Support of Scholarly Activities. (2016). https://doi.org/10.1145/2910896.2910912 Accepted: 2016-05-09T11:57:37Z.

[9] Katherine Kim, Wayne Graham, Paige Walker, National Digital Stewardship Alliance (NDSA), Carol Kussmann, and Aliya Reich. 2018. 2017 Web Archiving in the United States - A 2017 Survey. (Oct. 2018). https://doi.org/10.17605/OSF.IO/3QH6N Publisher: OSF.

[10] The Royal Danish Library. 2021. SolrWayback. https://github.com/netarchivesuite/solrwayback

[11] Ian Milligan. 2019. *History in the Age of Abundance?: How the Web Is Transforming Historical Research.*

[12] Nick Ruest, Samantha Fritz, Ryan Deschamps, Jimmy Lin, and Ian Milligan. 2021. From archive to analysis: accessing web archives at scale through a cloud-based interface. *International Journal of Digital Humanities* 2, 1 (Nov. 2021), 5–24. https://doi.org/10.1007/s42803-020-00029-6

[13] Nick Ruest, Jimmy Lin, Ian Milligan, and Samantha Fritz. 2020. The Archives Unleashed Project: Technology, Process, and Community to Improve Scholarly Access to Web Archives. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020 (JCDL '20)*. Association for Computing Machinery, New York, NY, USA, 157–166. https://doi.org/10.1145/3383583.3398513