The Distant Reader - Tool for Reading

Eric Lease Morgan
University of Notre Dame
Notre Dame, Indiana
emorgan@nd.edu

Eroma Abeysinghe
Science Gateways Research Center
Pervasive Technology Institute
Indiana University
Bloomington, Indiana
eabeysin@iu.edu

Sudhkar Pamidighantam
Science Gateways Research Center
Pervasive Technology Institute
Indiana University
Bloomington, Indiana
pamidigs@iu.edu

Eric Coulter
Science Gateways Research Center
Pervasive Technology Institute
Indiana University
Bloomington, Indiana
jecoulte@iu.edu

Suresh Marru
Science Gateways Research Center
Pervasive Technology Institute
Indiana University
Bloomington, Indiana
smarru@iu.edu

Marlon Pierce
Science Gateways Research Center
Pervasive Technology Institute
Indiana University
Bloomington, Indiana
marpierc@iu.edu

ABSTRACT

The Distant Reader science gateway can be used to automatically create and analyze text corpora at a scale of thousands of user-supplied documents. These processing steps are deployed on a dynamic virtual cluster deployed on XSEDE's Jetstream academic cloud computing resource and are accessed through a Web interface. The science gateway uses Apache Airavata middleware to manage the interactions between the Web interface and the virtual clusters. The gateway leverages the Science Gateway Platform as a service (SciGaP) infrastructure at Indiana University, which provides user authentication, authorization, and identity management as well as access to the Distant Reader tools. The Distant Reader is designed to assist in the process of using & understanding corpora – reading.

CCS CONCEPTS

• **Information systems → Content analysis and feature selection**; *Search interfaces*; • **Computer systems organization → Client-server architectures**.

KEYWORDS

Text Analysis, Apache Airavata, SciGaP, Distant Reader, URL, Study Carrels, Library, Science Gateways, Education, XSEDE, Cloud Computing, Virtual Clusters

ACM Reference Format:
Eric Lease Morgan, Eroma Abeysinghe, Sudhkar Pamidighantam, Eric Coulter, Suresh Marru, and Marlon Pierce. 2019. The Distant Reader - Tool for Reading. In *Practice and Experience in Advanced Research Computing (PEARC '19), July 28-August 1, 2019, Chicago, IL, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3332186.3333260

PEARC '19, July 28-August 1, 2019, Chicago, IL, USA

ACM ISBN 978-1-4503-7227-5/19/07…$15.00
https://doi.org/10.1145/3332186.3333260

1 INTRODUCTION

Close reading is the traditional practice of carefully analyzing a passage of text. The Distant Reader is a tool for reading a large body of literature, providing "close reading" at scale. Given a URL, a file of URLs, or a set of files, the Distant Reader harvests/caches the given content, transforms it into plain text, applies sets of natural language and text mining processes, reduces the whole to machine-readable formats and a relational database, summarizes what it learned, reduces the whole into a "study carrel", and returns a link to the resulting zip file. The "study carrel" can then be downloaded and used to supplement a student's/scholar's or a reader's traditional reading process. In order to share the tool with a broader audience of researchers and scholars, the Principal Investigator (PI) reached out to XSEDE's Extended Collaborative Support Services (ECSS) [8] for gateway services. After the initial interviews with ECSS managers, the PI met with SciGaP gateway platform with Apache Airavata middleware for gateway requirements.

2 NEED FOR DISTANT READER

From a librarian's perspective, the problem to solve nowadays is not about search nor discovery. Instead, the problem is one of use and understanding: "How do I make sense of all the good stuff I find?" Such is the problem the Distant Reader is intended to solve. Given dozens of scholarly journal articles, a long scientific report complete with links to cited documents, one or more books (such as the complete works of Charles Dickens), a blog, or simply a few hundred URLs of personal interest, the Distant Reader attempts to organize content into a coherent whole, analyze it, package up the analysis, and deliver the results for a more systematic reading experience.

3 THE DISTANT READER APPLICATION

3.1 Distant Reader Architecture

The Distant Reader tools's execution environment is made up of a single, relatively small head node and a dozen compute nodes. We dynamically provide a cluster on XSEDE's Jetstream [7], using SLURM as a scheduler and resource manager. After the head node receives a request from the scheduler, it uses OpenStack to provision

one or more compute nodes for executing the request. Each compute node is comprised of 10 cores, a shared file system, and 10 GB of RAM. Everything is run on top of (Centos) Linux.

The processing can be divided into three stages: 1) corpus creation, 2) content analysis, and 3) results reduction. Corpus creation is the most complicated task. The Reader (Distant Reader Software) takes the input and locally caches the associated content in its original form. Given a single URL, the Reader will cache the content at the other end of the URL, loop through the content to find additional URLs, and cache that content as well. Given a file where each line in the file is a URL, the Reader will retrieve the associated content and cache it locally. In this case, the Reader will not crawl the content for additional URLs. Given one or more files from the student's/researcher's file system, the Reader will then use that as the cache.

In the case of the single URL, the student/researcher may point to something like the root of an open access journal article or issue. The student might point the Reader to a Wikipedia article. They might point it to a blog, an aggregation of blog postings, or a mailing list archives. The single URL is the simplest type of input, but it is also the most messy in that it will indiscriminately cache anything linked from the original document.

The file of URLs is intended to cache something like sets of individual journal articles, sets of electronic books, maybe Google search results, or multiple aggregations of blog postings. The file of URLs is very scalable since the Reader can easily accommodate hundreds (if not thousands) of URLs in a single request. Processing of such a request does require hours of processing but is much faster and more thorough than the work of any individual person. On the other hand, the creation of a file of URLs is (ironically) difficult for many people to accomplish.

Files shared from a student's/researcher's file system is the most targeted type of input, but the Reader is designed for and most useful when at least a few documents are analyzed. Individually selecting more than a few documents from one's file system can get tedious. A solution to this problem has yet to be articulated.

The second processing stage – content analysis – is embarrassingly parallel. It is during this phase of processing that each item in the cache is transformed into plain text and indexed in a number of ways. Next, a number of different computer technologies and languages are employed. Tika is used for the transformation to plain text. Bash is used to loop through files on the file system, call either Perl or Python scripts, and distribute the calls across the CPUs. Output is usually in the form of plain text (tab delimited files or SQL statements). Some scripts are implemented as servers listening on local ports for input. Other scripts take entire files as input and use various natural language modules (Spacy, Gensim, etc.) to do analysis. Since very little of this processing relies on the output of other processing, the content analysis stage is highly parallelizable. During this stage of the process the load on CPUs on the compute nodes is all but maximized.

The final processing stage – reduction – is simplistic. In this stage all of the previously created index files and SQL statements are first distilled into the normalized tables of a relational (SQLite) database. The reduction stage also creates a semantic (word2vec) index, and applies sets of SQL queries to the database thus outputting a sort of summary of the database's content. Finally, the whole collection (original data, transformed text files, tab delimited files, database, semantic index, and summary file) are compressed into a single zip file – a "study carrel", described below. A link to the study carrel is then provided as output to the student/researcher.

3.2 Study Carrel

The tangible output of the Distant Reader process is called a "study carrel". In reality, it a set of files, each being a different type of index against each harvested/cached file. There are indices for part-of-speech, named-entities, readability scores, document lengths, summary statements, email addresses, URLs, internet domains, keywords, etc. All of these indices are amalgamated into a single SQLite database. The indexes (in the form of tab delimited files), the database, the originally harvested files, and the harvested files transformed into plain text are all zipped into a single compressed file.

The student/researcher is expected to download the study carrel, unzip it, and uses the files found there for further investigation. For example, the tab delimited files can be imported in something like OpenRefine, faceted, and queried to answer questions such as "What are the things under discussion in the corpus?", "What actions take place in the corpus?", or "Who and what places are mentioned in the corpus?". It is then possible to illustrate the answers with word clouds. Topic modeling can be done against the plain text versions of the corpus, and themes can be drawn out and enumerated. If the Reader classifies sets of documents with one or more keywords, themes, etc., then the topic models can pivoted to literally illustrate themes within the classifications. Short, simple grammars can then be articulated by the Reader, such as noun-verb-noun, and these grammars can be transformed into SQL queries, and finally sentences matching the queries can be pulled from the corpus.

All of the activities above are akin to "distant reading", but more traditional reading or "close reading" has just as much benefit, albeit not as scalable. To accommodate close reading, the student can query the study carrel, identify things of interest, open the plain text versions of the corpus in a concordance, and then do intelligent and targeted searching of the corpus, and if that is not enough, the cached documents in their original form can be read online or printed for even more analysis.

4 BUILDING A DISTANT READER GATEWAY

The gateway starts with assigning a domain and a SciGaP basic theme and is deployed in the multi-tenanted SciGaP gateway hosting system at Indiana University [6]. This includes integration with the basic services Apache Airavata [4] provides such as student/researcher authentication and authorization technology [2], data and applications APIs, experiment orchestration and application and compute resource configuration.

Once the gateway is deployed, it is configured with specific compute and storage resources (in this case the Jetstream elastic virtual cluster described previously) and a local storage device. Apache Airavata communicates with these resources via SSH keys generated specifically through the gateway Credential Store [3]. The compute resource specification is inherited from the SciGaP compute resource profiles.

Gateway specific applications are registered using a three component system that consists of a 'module' that identifies the application, a 'deployment' that defines the access to the application on the compute resource and an 'interface' that provides user facing interaction enabler to define input parameters and other data that drives the experiment and its execution. The gateway provides experiment launching and monitoring panels (pages) and provides links for outputs generated at the end of the successful execution or error information in the case of execution failures. The gateway provides administration API interfaces through which the management and configurations can be achieved.

As with all SciGaP-hosted gateways, Distant Reader uses the Apache Airavata's identity management system, which is based on the open source Keycloak software. Students/Researchers can authenticate to the gateway with their existing institute logins through CILogon [1]. Once the students/researchers create their accounts, the gateway PI can grant access to use the Distant Reader tool.

Gateway users (Students/Researchers) can create "experiments" in the gateway in order to launch Distant Reader executions on Jetstream and view and download results from the gateway interface. Apart from downloading data, they can also share their data with other gateway users through Apache Airavata sharing services [5]. Figure 1 displays the basic layout of Distant Reader gateway.

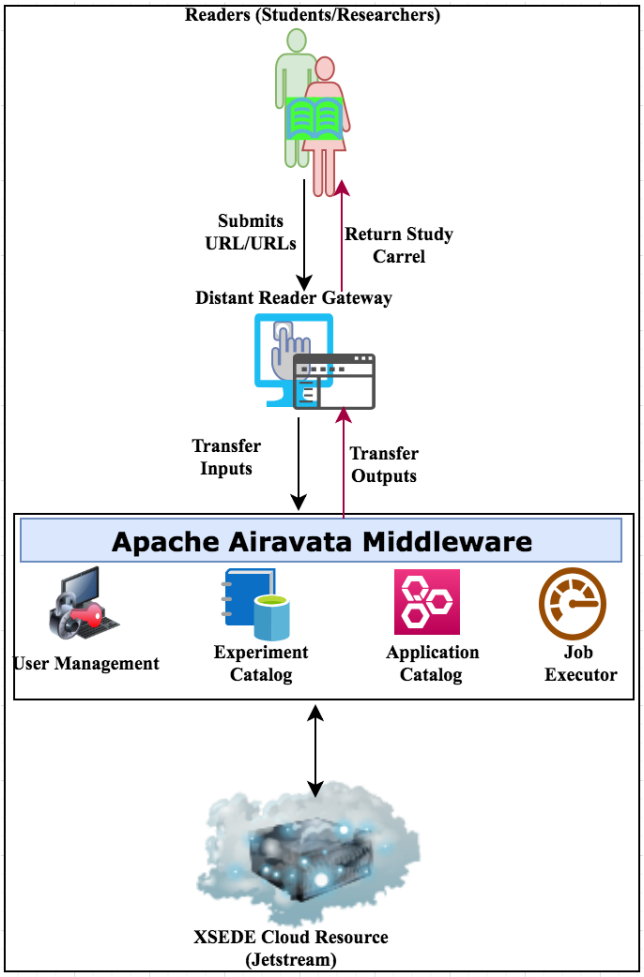

Figure 1: The Distant Reader Gateway with Apache Airavata

5 OUTREACH

Focus group interviews were held at the University of Notre Dame regarding the process of "distant reading", and it was through these interviews that the Distant Reader was conceived. In May of 2019 the PI will attend SGCI's Science Gateway Bootcamp, and during that week long event ideas and plans regarding outreach activities will be articulated and outlined. Other outreach activities will include hands-on workshops facilitated in libraries, mailing list announcements, and hopefully the integration of the Distant Reader into teaching/learning activities across the Notre Dame campus.

6 FUTURE WORK

There are a number of ways the Distant Reader could be improved. For example, the Reader's input could be expanded to include single, zip files of archived documents. Such would make it easier for a student/researcher to have a greater amount of their content analyzed. Since a student's/researcher's scholarly interests are rather focused, the study carrels created by each student/researcher are probably similar in subject matter. And since each study carrel is similarly laid out, it would be possible and meaningful to combine multiple study carrels into single, larger study carrels. Similarly, study carrels could be made public or shared between different student/researcher. More importantly, interfaces ought to be created making the process of using study carrels more efficient. Pre-staging data from the URLs would be another options to reduce the overall time to results and additional analysis visualization would also be useful.

7 CONCLUSIONS

The Distant Reader science gateway functions. It successfully creates large corpora, analyzes the corpora, and saves the results in both human-readable and computer-readable formats, and it does all of this work at scale. In order to make the Distant Reader truly successful, a number of things needs to be developed. First of all, scripts need to be written which interact with the Distant Reader's output, thus making the output more accessible. Second, Web interfaces against study carrels need to be implemented to make things even more accessible. Third, the user base needs to be increased. We sincerely believe the Distant Reader can be used to increase the use & understanding of large corpora. These improvements will take the Distant Reader gateway from its current basic state to one that can be effectively used by a larger community.

ACKNOWLEDGMENTS

This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1548562. Development of the Apache Airavata used to develop the science gateway is supported by NSF award #1339774.

REFERENCES

[1] Jim Basney, Terry Fleury, and Jeff Gaynor. 2014. CILogon: A federated X. 509 certification authority for cyberinfrastructure logon. *Concurrency and Computation: Practice and Experience* 26, 13 (2014), 2225–2239.

[2] Marcus A Christie, Anuj Bhandar, Supun Nakandala, Suresh Marru, Eroma Abeysinghe, Sudhakar Pamidighantam, and Marlon E Pierce. 2017. Using Keycloak for Gateway Authentication and Authorization. (2017).

[3] Thejaka Amila Kanewala, Suresh Marru, Jim Basney, and Marlon Pierce. 2014. A credential store for multi-tenant science gateways. In *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*. IEEE, 445–454.

[4] Suresh Marru, Lahiru Gunathilake, Chathura Herath, Patanachai Tangchaisin, Marlon Pierce, Chris Mattmann, Raminder Singh, Thilina Gunarathne, Eran Chinthaka, Ross Gardler, et al. 2011. Apache airavata: a framework for distributed applications and computational workflows. In *Proceedings of the 2011 ACM workshop on Gateway computing environments*. ACM, 21–28.

[5] Supun Nakandala, Suresh Marru, Marlon Piece, Sudhakar Pamidighantam, Kenneth Yoshimoto, Terri Schwartz, Subhashini Sivagnanam, Amit Majumdar, and Mark A Miller. 2017. Apache Airavata Sharing Service: A Tool for Enabling User Collaboration in Science Gateways. In *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact*. ACM, 20.

[6] Marlon Pierce, Suresh Marru, Eroma Abeysinghe, Sudhakar Pamidighantam, Marcus Christie, and Dimuthu Wannipurage. 2018. Supporting Science Gateways Using Apache Airavata and SciGaP Services. In *Proceedings of the Practice and Experience on Advanced Research Computing*. ACM, 99.

[7] Craig A Stewart, Timothy M Cockerill, Ian Foster, David Hancock, Nirav Merchant, Edwin Skidmore, Daniel Stanzione, James Taylor, Steven Tuecke, George Turner, et al. 2015. Jetstream: a self-provisioned, scalable science and engineering cloud environment. In *Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure*. ACM, 29.

[8] Nancy Wilkins-Diehr, Sergiu Sanielevici, Jay Alameda, John Cazes, Lonnie Crosby, Marlon Pierce, and Ralph Roskies. 2015. An overview of the XSEDE extended collaborative support program. In *International Conference on Supercomputing in Mexico*. Springer, 3–13.