TRAIL: Audit Trails for Enhanced Reproducibility and Observability of Research Computing

1st Jake Rosenberg
Web and Mobile Applications
Texas Advanced Computing Center
Austin, TX
jrosenberg@tacc.utexas.edu

1st Rich Cardone
Cloud and Interactive Computing
Texas Advanced Computing Center
Austin, TX

1st Gilbert Curbelo
Cloud and Interactive Computing
Texas Advanced Computing Center
Austin, TX
gcurbelo@tacc.utexas.edu

1st Vanessa Gonzalez
Web and Mobile Applications
Texas Advanced Computing Center
Austin, TX
vgonzalez@tacc.utexas.edu

1st Steve Black
Cloud and Interactive Computing
Texas Advanced Computing Center
Austin, TX
scblack@tacc.utexas.edu

2nd Sal Tijerina
Web and Mobile Applications
Texas Advanced Computing Center
Austin, TX
stijerina@tacc.utexas.edu

2nd Erik Rivera-Sanchez
Web and Mobile Applications
Texas Advanced Computing Center
Austin, TX
erik.riverasanchez@austin.utexas.edu

3rd Maytal Dahan
Advanced Computing Interfaces
Texas Advanced Computing Center
Austin, TX
maytal@tacc.utexas.edu

4th Dan Stanzione
Director's Office
Texas Advanced Computing Center
Austin, TX
dan@tacc.utexas.edu

***Abstract*—As research projects grow more complex and researchers use a mix of tools - command-line scripts, science gateways, and Jupyter notebooks - it becomes increasingly difficult to track exactly how a final result was produced. Each tool often keeps its own logs, making it hard to reconstruct the full sequence of computational steps. This lack of end-to-end visibility poses a serious challenge for scientific reproducibility. Yet advanced computing remains a critical part of nearly every field of academic research, and researchers continue to rely on a wide range of interfaces to run their scientific software. To address this challenge, the Advanced Computing Interfaces group at the Texas Advanced Computing Center (TACC) created a system that collates logs from multiple sources - science gateways, Jupyter notebooks, and the Tapis platform - into one unified "audit trail." The TACC Research Audit and Integration of Logs (TRAIL) system allows researchers and staff to follow the complete path a dataset or file took: from the moment it was first uploaded to TACC, through every step of computation, to the final result. This kind of tracking helps ensure scientific results can be reproduced and gives advanced computing services better insight into how data and resources are being used.**

***Index Terms*—High performance computing, Replicability, Software engineering, Database systems**

I. Introduction

In response to the increasing prevalence of advanced computing in academic research, a number of tools have arisen to simplify access to computational resources. Jupyter notebooks provide an interactive platform to edit and execute code and create a record of the computational workflow. Science gateways enable collaborative data management and provide a graphical interface for submitting jobs on high-performance computing systems [3]. At the Texas Advanced Computing Center (TACC), these utilities connect users with HPC resources using the Tapis platform. Tapis provides a unified API through which GUI applications can communicate with HPC systems by making HTTP requests or invoking a Python SDK [1].

The breadth of options enabled by Tapis gives researchers the flexibility to define a computational workflow that suits the needs of their specific project, but it presents challenges as well. If multiple interactive tools were involved in the course of a computational project, or if multiple users collaborated asynchronously, it can become difficult to accurately reconstruct the exact sequence of operations that produced a given result. Different tools log their activity using different schemas, and these logs are not guaranteed to persist beyond the lifetime of the research project. A third party attempting to reproduce the analysis would therefore rely on the original authors having followed best practices and kept detailed records of their computational workflow [4]. This is far from guaranteed; a survey of researchers identified non-replicable or poorly specified data analysis workflows as a key contributing factor to the reproducibility crisis in science [2]. Moreover, as providers of advanced computing services, it is in our interest to deepen our understanding of user behavior in order to meet the needs of researchers in an evolving computational landscape.

To address the issue of reproducibility and observability of computational workflows, we introduce the TACC Research Audit and Integration of Logs (TRAIL) facility. TRAIL is a methodology for generating audit trails, defined here as chronological sequences of the actions taken by users and tools in the course of a computational project. The system is intended to be granular at the level of individual files, and is meant to answer questions of the form:

- Given the output of a computational job, what were its inputs and which other computations were those inputs involved in?
- How did a given file move through the data center after being uploaded, and which users interacted with it using which tools?
- Which files and computational resources did an individual user interact with during a given session in a science gateway?

TRAIL is currently implemented at TACC and collates logs across 10 science gateways and 2 deployed instances of Jupyterhub, all of which invoke the Tapis platform to connect users with TACC systems. It consists of an ETL pipeline for ingesting logs from compliant applications, enforcing a consistent schema, and storing the transformed records in a persistent database for subsequent analysis. The ultimate goal of this work is to trace the outputs of computational research back through the workflows that created them, and to communicate these workflows to users so that they can better understand and replicate the processes involved. In this paper we describe the TRAIL architecture, the ETL pipeline and database schemas involved, and some considerations to address the challenges of keeping audit trails complete and valid through workflows that involve multiple entry points to computational resources.

II. Architecture and Implementation

The main components of the TRAIL auditing facility are:

- **Log Writers** - The I/O subsystems in TRAIL-enabled applications that write text records to their logs, which are then extracted by the Extractor and inserted into the Audit DB.
- **Extractor** - A program that extracts audit-relevant records from the logs of each TRAIL-enabled application.
- **Audit Database** - A SQL database into which all audit records are inserted and available for analysis.

Log writers are not part of the dedicated TRAIL infrastructure; rather, they are implemented on a per-application basis. Applications that use the Tapis platform to manage data and submit jobs are considered TRAIL-compliant if their logs contain enough information to populate the schema in **Table 1**. The information collected by TRAIL-compliant log writers enables both file provenance tracking and user session tracing. Administrators and users can reconstruct the life cycle of files — including creation, modification, movement, and deletion — and view actions taken within the portal, such as uploads, downloads, and job submissions. Each audit record includes a **tracking ID** for cross-system correlation and a **guid** for deduplication, ensuring reliable and comprehensive traceability. Operations which copy or transfer files record a **source** and **target** so that file provenance can be traced back through the full history of operations on those files. These audit trails support security auditing, operational monitoring, user accountability, and research reproducibility by making it possible to verify the origin and history of user activities and data.

The Extractor is a dockerized Python script which runs periodically to process logs from compliant applications. Each application supported by TRAIL has an associated module within the Extractor which takes the entries from each Log Writer and processes them to conform to the schema of the audit database. The TRAIL methodology is not prescriptive about the structure or formatting of the logs produced by each enabled application. Rather, as long as the logs produced by the Log Writer comprise a superset of the required fields in the audit database, the application authors can provide a specialized Extractor module for processing that applications's logs. This enables audit logging to be extended to any application that uses Tapis to interact with TACC's computing resources. Each Extractor module implemented so far at TACC pulls its records from the Splunk monitoring platform, which ingests syslog records from every host deployed at the center.

After logs are processed by the Extractor, they are inserted into the Audit Database. This is a PostgreSQL database hosted on Rodeo, TACC's internal VMware installation. Based on initial capacity testing, we estimated that supporting our current suite of science gateways and JupyterHub instances would involve ingesting about 50,000 logs per day, or 60 MB. At current usage levels we expect the database to grow by approximately 25 GB/year. For the initial rollout, we provided 100GB of dedicated disk space for the Audit DB, which can be upgraded or scaled through table partitioning as our needs evolve. We note that the Audit DB is intended as a permanent store of user activity and file system events. A dedicated database is required because such persistence guarantees are not provided by our Splunk cluster, which ingests gigabytes of system logs per day and must occasionally purge old records.

III. Discussion and Logistical Considerations

A. Constructing Audit Trails for Individual Files:

Files and user activities are tracked using a **Tracking ID** attached to every action. Each user session within a TRAIL-enabled science gateway is assigned a tracking ID, most commonly by hashing the session ID generated by the Django web framework. Tracking IDs associate a client (or sender) with an operation carried out by a server (or receiver). In the science gateways/Tapis context, senders include tracking IDs in the headers of HTTP requests they send to receiving services. Receivers include these tracking IDs in any audit records they generate while servicing the requests. Receivers may also save tracking IDs in their own databases. Tracking IDs allow manual and automated procedures to trace all tracked operations that originate from a single sender instance, whether that instance is a portal session or a job. These traces start at a root sender and form trees of arbitrary depth of

TABLE I
FIELDS FOR FILE OPERATIONS IN THE TRAIL DATABASE SCHEMA

Database Field	**Description**	**Notes**
id	DB Sequence Id	Automatically generated by postgres.
uuid	Unique Id	Generated by Extractor
guid	Globally Unique Id	Generated by Log Writer and used for deduplication
timestamp	Log timestamp	
jwtTenant	Tapis requester tenant	
jwtUser	Tapis requester username	
oboTenant	Tapis on-behalf-of tenant	
oboUser	Tapis on-behalf-of username	
action	Action requested copy, move, transfer, upload, mkdir, delete, chmod, chown, chgrp, setfacl	Only copy, move and transfer involve a source and target. Other operations only have a target.
targetSystemId	The Tapis system ID of the affected file.	
targetSystemType	The type of the target system (POSIX, S3, Globus, IRODS, etc).	
targetHost	The target host, which currently may be a DNS name, IP address, or Globus ID.	
targetPath	Absolute paths only.	
sourceSystemId	The Tapis system ID of the original file, if one exists.	Used only on actions with both source and target files: copy, move and transfer
sourceSystemType	The type of the source system (POSIX, S3, Globus, IRODS, etc),	
sourceHost	The source host, which currently may be a DNS name, IP address, or Globus ID.	
sourcePath	Absolute paths only.	
data	Metadata in the form of a JSON object.	Optional additional information associated with the event.
trackingId	Used to identify all records associated with something to track: A portal session, job or file transfer.	Job uuid for a job, Session id for a portal session, or Transfer id for a file transfer.
parentTrackingId	Used to indicate trackingId is part of something else that is being tracked: A portal session or a job.	Session id if job is part of a portal session, or Job uuid if file transfer is part of a job.

senders connected to receivers. To avoid cycles, each sender creates a unique tracking ID for each sending context (i.e., each gateway session, batch job, etc.) from which it makes requests.

As a concrete example, suppose `user1` starts a portal session which is assigned tracking ID `portal.123`. When `user1` runs their first job, the Tapis Jobs service receives ID `portal.123` and saves that in its job record. The job then creates its own unique tracking ID, `job.456`. Every call the job makes to the Files API includes `job.456` in the request header and in the audit record it writes to its audit log. When the Files API receives the request, it records ID `job.456` in its audit log and in its database (for asynchronous transfers only). If `user1` runs a second job, that job will be assigned a new tracking ID, `job.789`, which will again be propagated through subsequent file operations. The tree of all file operations associated with portal session `portal.123` can be constructed by querying the Audit Database's file provenance table for all records with tracking ID `portal.123`. This query returns `job.456` and `jjob.789`. A second query can then be issued to find all file operations with tracking IDs `job.456` and `job.789`. By following these tracking IDs to the point at which input files were initially uploaded, the full sequence of file operations and computational steps can be reconstructed.

B. Determining Canonical Hosts and Paths for Files

In order to track provenance for a file path, there must be a way to uniquely identify the file. This is challenging, because the information available in an audit record does not uniquely identify a file. Knowing the host and absolute path to the file is not enough. The host may be an IP address or a DNS name that routes to the same physical or virtual location. Also, if the file is on a shared file system or if there are symbolic links, the absolute path may differ by host, depending on the mount points for the hosts and the symbolic links. This problem is addressed by maintaining two sets of auxiliary information. One set contains a mapping of host names and IP addresses to a canonical host. A second set contains mapping information allowing us to determine a unique canonical host/path combination given a canonical host and absolute path.

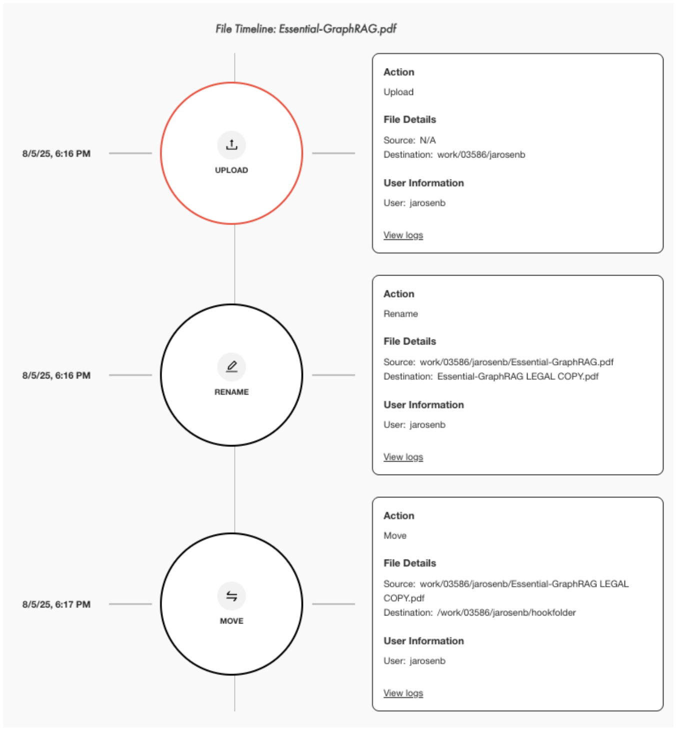
The concept of a canonical host is important in order to uniquely identify a host when the host attribute from a Tapis system can be an IP address or one of many host names that resolve to the same physical host via DNS. As a concrete example, the records "`login1.frontera.tacc.utexas.edu`" and "`29.114.63.98`" both resolve to login nodes on the Frontera supercomputer. Within the TRAIL extractor, these records are therefore both assigned a canonical host of "`frontera.tacc.utexas.edu`".

A second issue is how to uniquely identify a file path given the canonical host name and absolute path on the host. If two hosts have shared storage mounted at different paths or hosts have symbolic links, then the absolute path to the same file may differ by host.To address this, we maintain a mapping of shared paths to canonical hosts and the corresponding paths on the hosts. For a given shared path, only one of the entries must be marked as the reference entry. A reference entry is used to determine which host is the unique canonical host to use for the final unique host+path identifying the file path.

For instance, suppose that the host/path combination "`data.tacc.utexas.edu:/data`" is listed as the reference entry for a data directory. If the NFS mount "`frontera.tacc.utexas.edu:/mounted-data`" is recorded as a mount point for this directory within the Extractor, then its canonical host/path will be overwritten to "`data.tacc.utexas.edu:/data`" when recording file operations in the database. This allows us to uniquely identify files even when they are interacted with at different mount points, reducing the likelihood that the file provenance trail will be broken in the course of a user's computational workflow.

IV. Summary and Future Work

The TACC Research Audit and Integration of Logs (TRAIL) system is an audit trail scheme implemented at TACC, with the goal of enhancing the reproducibility of computational workflows. It was developed in response to the increasing diversity of tools used by researchers to access computational resources. In a multi-modal workflow, reconstructing the exact sequence of operations poses a challenge because a given workflow might involve multiple utilities with different logging behavior. TRAIL addresses this challenge by providing a compliance standard for applications to meet, at which point their logs can be processed and collated with other utilities in the computational ecosystem. For applications built on the Tapis platform, the movement of data can be traced through the file system, from initial upload to final result. The next step for this project will be to build upon the TRAIL database to provide tangible value for researchers. The ultimate goal is for users to be able to select a file in a science gateway UI and retrieve its complete provenance trail. This trail could be provided as part of a published work, in order to facilitate replication of the computational results. A prototype TRAIL UI is currently in development, and a representative audit trial is given in **Figure 1**. Additionally, in the context of a computing facility, audit trails can provide value by enabling analysis of user behavior, helping identify opportunities to expand services or streamline operations.


File Timeline: Essential-GraphRAG.pdf

8/5/25, 6:16 PM
UPLOAD

Action
Upload
File Details
Source: N/A
Destination: work/03586/jarosenb
User Information
User: jarosenb
View logs

8/5/25, 6:16 PM
RENAME

Action
Rename
File Details
Source: work/03586/jarosenb/Essential-GraphRAG.pdf
Destination: Essential-GraphRAG LEGAL COPY.pdf
User Information
User: jarosenb
View logs

8/5/25, 6:17 PM
MOVE

Action
Move
File Details
Source: work/03586/jarosenb/Essential-GraphRAG LEGAL COPY.pdf
Destination: /work/03586/jarosenb/hookfolder
User Information
User: jarosenb
View logs

Fig. 1. TRAL user interface showing the timeline for a representative file, with sequential upload, rename, and move operations represented

References

[1] Tapis: An API Platform for Reproducible, Distributed Computational Research. In *Advances in Intelligent Systems and Computing*, pages 878–900. Springer International Publishing, Cham, 2021. ISSN: 2194-5357, 2194-5365.

[2] Monya Baker. 1,500 scientists lift the lid on reproducibility. *Nature*, 533(7604):452–454, May 2016. Publisher: Springer Science and Business Media LLC.

[3] Sandra Gesing, Steven Brandt, Shannon Bradley, Mark Potkewitz, Kerk Kee, Noreen Whysel, Mark Perri, Sean Cleveland, Annelie Rugg, and Jack Smith. A Vision for Science Gateways: Bridging the Gap and Broadening the Outreach. In *Practice and Experience in Advanced Research Computing*, pages 1–8, Boston MA USA, July 2021. ACM.

[4] Geir Kjetil Sandve, Anton Nekrutenko, James Taylor, and Eivind Hovig. Ten Simple Rules for Reproducible Computational Research. *PLoS Computational Biology*, 9(10):e1003285, October 2013. Publisher: Public Library of Science (PLoS).