Improving latency and scalability of the user runtime job log collecting and exposure in the REANA reproducible analysis platform

IRIS-HEP project proposal

Jelizaveta Lemeševa

REANA is “a reproducible analysis system allowing scientists to run containerized data analysis pipelines on remote computer clouds”[1]. It allows researchers to express their computational analyses using declarative workflow languages (CWL, Snakemake, Yadage) that are then executed on a target compute backend. The researchers can easily execute and rerun the analysis with different input data, parameters or code.

Scientists interact with REANA via a command-line client or via the web interface. The client allows to send requests to connected REANA cluster for their execution. The REANA cluster consists of several micro-services that accept client requests and schedule analysis workflows and jobs for the execution on the supported compute backends (Kubernetes, HTCondor, Slurm). The REANA cluster schedules user workflows and jobs for execution by sending requests to compute backend’s API. The compute backend executes the job and notifies the REANA cluster about its completion. The REANA cluster then retrieves the logs of the jobs from the compute backend and stores them in a PostgreSQL database for persistency. The client retrieves these logs from the REANA cluster and displays them to the user.

In the current setup, users can access workflow and job logs only after the job execution has finished. When a workflow consists of many jobs, the researcher can therefore consult logs of already finished steps, but not of the step that is currently being executed.

The goal of this IRIS-HEP project proposal is to enhance the REANA job logging system with a possibility to capture logs of executing processes “live”, so that the user would have access to the running job logs even during job execution. Certain jobs can run for several hours so it would often be beneficial to peek into the execution process without much latency delays.

The technological solution to this problem could consist of introducing a sidecar container that would be deployed alongside the main job, reading the logs in real time and pushing them to the database. An alternative approach could be to deploy daemonsets that would listen to all user workload on a given cluster node and dispatch the results to a new log collector service, living outside of the database, for example in an OpenSearch. This may make the log collection more

[1] https://docs.reana.io/

scalable at the price of introducing a new infrastructure component and making necessary changes to the REANA application to expose thusly collected logs back to the clients in the traditional way. Further evaluation is needed to decide which approach to take.

In either case, it is desired that the logs are exposed to the user over the command-line client and the web interface with very low latency.

Objectives:

- Design a solution to efficiently capture live workflow logs and job logs whilst they are being executed.
- Analyse the pros and cons of the various technical approaches to capturing Kubernetes pod logs with low latency.
- Implement the chosen solution by amending the necessary REANA micro-service components.
- Update REANA client and web interface to display live log streams to the user.
- Run benchmarks to assess the performance of the developed solution.

The project will use an agile and iterative development model with the following tentative time plan:

Task	Weeks
Introduction to REANA infrastructure	1
Rapid prototyping of technical approaches to capturing main job logs from the Kubernetes pods	2
Design of the selected live logging system	3-4
Backend implementation	5-8
Client implementation	9-10
Performance benchmarks	11-12