EPJ Web of Conferences **245**, 06041 (2020) https://doi.org/10.1051/epjconf/202024506041
CHEP 2019

Hybrid analysis pipelines in the REANA reproducible analysis platform

Diego Rodríguez[1], *Rokas* Mačiulaitis[1], *Jan* Okraska[1], and *Tibor* Šimko[1,*]

[1]CERN, Geneva, Switzerland

Abstract. We introduce the feasibility of running hybrid analysis pipelines in the REANA reproducible analysis platform. The REANA platform allows researchers to specify declarative computational workflow steps describing the analysis process and to execute analysis workload on remote containerised compute clouds. We have designed an abstract job controller component permitting to execute different parts of the analysis workflow on different compute backends, such as HTCondor, Kubernetes and SLURM. We have prototyped the designed solution including the job execution, job monitoring, and input/output file staging mechanism between the various compute backends. We have tested the prototype using several particle physics model analyses. The present work introduces support for hybrid analysis workflows in the REANA reproducible analysis platform and paves the way towards studying underlying performance advantages and challenges associated with hybrid analysis patterns in complex particle physics data analyses.

1 Introduction

REANA is a reproducible analysis platform that allows researchers to run containerised analysis workflows on remote compute clouds [1]. It was developed within the wider context of CERN analysis preservation and reuse framework [2]. The researcher describes (i) the input data, (ii) the analysis code, (iii) the computing environment and depending libraries, and (iv) the computational workflow steps used to perform the analysis. The REANA platform will take care of instantiating thusly containerised analysis on supported compute backends. See Figure 1.

The main compute backend for analysis jobs in the REANA platform is the Kubernetes cloud. We have recently added support for dispatching high-throughput jobs to the HTCondor platform [5]. Starting from the job abstractions designed for the HTCondor compute backend, this paper describes recent developments allowing to send high-performance jobs to the Slurm platform. This enables to run hybrid analysis workflows where some part of the analysis pipeline are run on Kubernetes, some on HTCondor, and some on Slurm compute backends.

The developed integration with the Slurm compute backend is described in Section 2. The hybrid analysis pipeline concept together with a simple analysis example is presented in Section 3. The advantages and disadvantages of hybrid declarative analysis approach are briefly discussed in Section 4.

*e-mail: tibor.simko@cern.ch

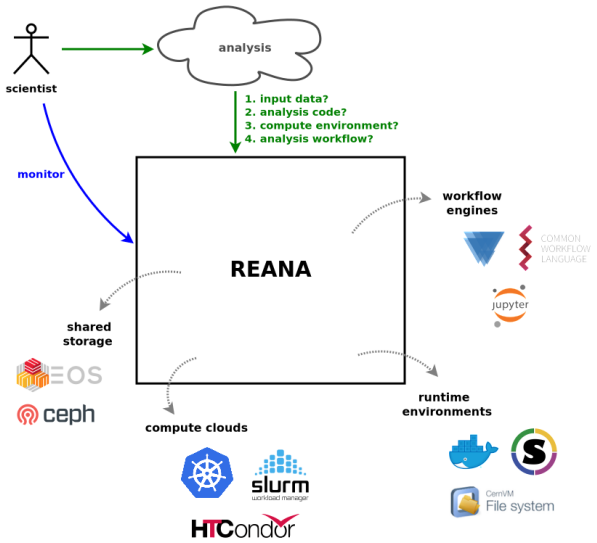

Figure 1. A schematic representation of REANA platform taking user inputs, code, environment and workflow defintion for executing analysis pipeline using various workflow systems (such as CWL [3] and Yadage [4]) on various compute backends (such as Kubernetes, HTCondor, Slurm).

2 Slurm compute backend

REANA platform consist of several components with separated concerns and responsibilities in instantiating and executing user workflows as a set of orchestrated runtime computing jobs. `REANA-Job-Controller` is responsible for executing and managing individual jobs. It takes inputs, container image and commands to run, prepares the payload and dispatches it to the Kubernetes backend for execution. The job controller uses REST API technology for communication with other components.

The current `REANA-Job-Controller` REST API was redesigned to ease integration of external compute backends for job execution [5]. The designed solution allowed to extend REANA job execution capabilities with external compute backends such as HTCondor for high-throughput computing.

The redesigned `REANA-Job-Controller` abstraction concerns job submission and execution, job status monitoring and the input/output data transfer between REANA platform and remote compute backends. Each supported backend can easily customise common interface. Figure 2 shows the abstract job manager class and its three implementations.

In this paper we have developed the Slurm compute backend integration. The typical job submission to Slurm HPC cluster is done from the head node. To allow communication between REANA and Slurm clusters, we have designed an `ssh`-based communication mechanism which establishes a connection used for job input/output transfer via SFTP, job submission, and job monitoring. The Singleton design pattern helps to ensure that only one connection per workflow is established in order to economise resources. The `ssh` connection to the Slurm head node is authenticated in a passwordless manner by making use of the Kerberos technology from user-provided secrets.

The Slurm cluster that was used for testing REANA↔Slurm extension does not support Docker due to security concerns. In HPC centres, Singularity container technology is more widely used. Singularity provides a mechanism to convert Docker images and run containers

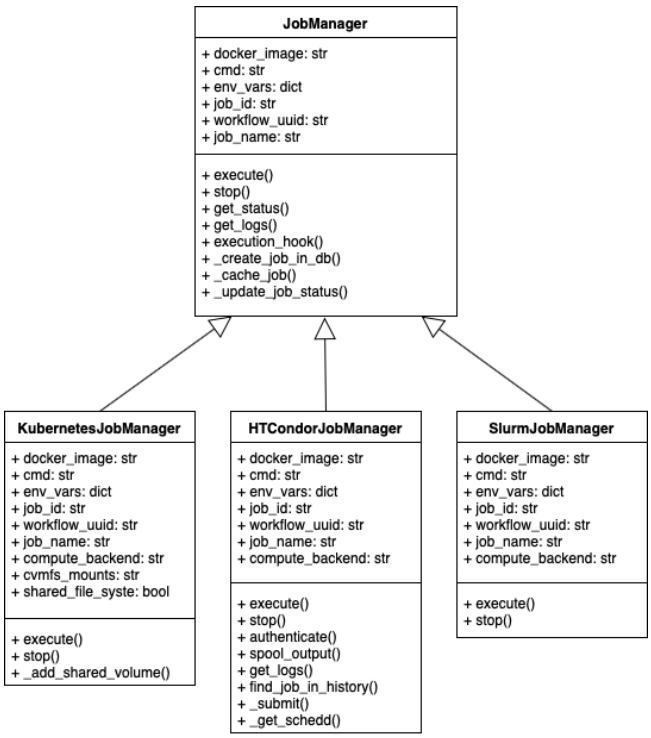

Figure 2. The abstract `REANA-Job-Controller` component interface designed to allow job execution and monitoring for multiple supported compute backends. [5]

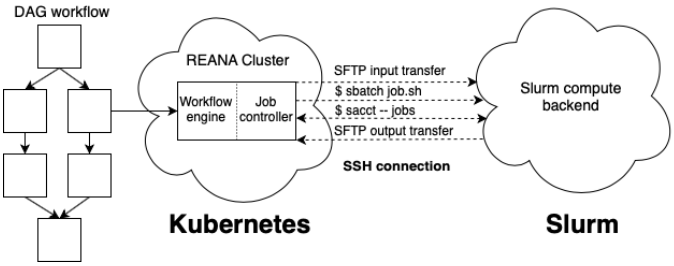

Figure 3. The typical interaction between REANA platform (running on Kubernetes) and Slurm compute backend (if a job is dispatched there).

in that way. The payload that is being sent to the Slurm cluster was therefore wrapped to generate Singularity execution command.

The typical communication scenario between the REANA cluster running on Kubernetes and the Slurm job execution backend cluster is presented in Figure 3. A code example implementing job Slurm job submission, illustrating implementation of the `execute()` method of the abstract `REANA-Job-Controller` interface, is illustrated in Figure 4.

The `REANA-Job-Controller` component was thusly enriched to provide full support for running payloads on Slurm computed backend using Singularity container technology. The

```
@JobManager.execution_hook
def execute(self):
    """Execute / submit a job with Slurm."""
    self.cmd = self._encode_cmd(' '.join(self.cmd))
    initialize_krb5_token(workflow_uuid=self.workflow_uuid)
    self.slurm_connection = SSHClient(
        hostname=SlurmJobManagerCERN.SLURM_HEADNODE_HOSTNAME,
        port=SlurmJobManagerCERN.SLURM_HEADNODE_PORT,
    )
    self._transfer_inputs()
    self._dump_job_file()
    self._dump_job_submission_file()
    stdout = self.slurm_connection.exec_command(
        'cd {} && sbatch --parsable {}'.format(
            SlurmJobManagerCERN.SLURM_WORKSAPCE_PATH,
            self.job_description_file))
    backend_job_id = stdout.rstrip()
    return backend_job_id
```

Figure 4. The Slurm job submission code example consists of encoding executable command to ensure escaping of special shell characters, Kerberos token initialization, establishing an SSH connection, transferring inputs, generating job submission files, and submission of a job. Implements `execute()` method of the abstract `JobManager` class presented in Figure 2.

development was tested using CERN Slurm cluster. The following section will provide one concrete example of a hybrid workflow.

3 Hybrid analysis pipelines

The integration of REANA with the external HTCondor and Slurm compute backend platforms opens up a possibility for users to execute their analysis in *hybrid* manner, with different parts of computational workflow tree being executed on different compute clouds.

Let us use the CMS open data Higgs-to-four-lepton analysis example [6] for illustration. The analysis workflow is presented in Figure 5 and is composed of three steps. The initial two steps are parallel, processing collision data and simulated data, and are followed by the third final step which merges the results and produces the final plot. The different steps could be dispatched to different compute backends depending on different factors assessed by the user or by the system: a certain compute backend may be more suitable for a certain task, e.g. HTCondor for high-throuput calculations and Slurm for high-performance calculations; or various users may have various lower or higher priorities for certain compute backends.

Let us look at how this computational workflow, consisting of three steps, is expressed in REANA using the CWL workflow definition language. Figure 6 displays the core part of the workflow. As can be seen, the researcher needs to specify, in a declarative way, each step of the workflow with its inputs, outputs, and whether some steps depend on other steps. Note the use of the *compute_backend* hint which allows the user to specify the desired compute backend for each step of the workflow.

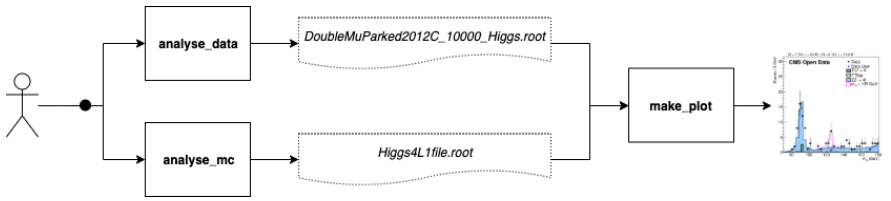

Figure 5. A Higgs-to-four-lepton physics analysis example using CMS open data. [6]. Note two parallel steps, analysing collision and simulated datasets; the results are then combined to make final plot.

```
steps:
  analyse_data:
    run: analyse_data.cwl
    hints:
      reana:
        compute_backend: slurmcern
    out: [DoubleMuParked2012C_10000_Higgs.root]
  analyse_mc:
    run: analyse_mc.cwl
    hints:
      reana:
        compute_backend: htcondorcern
    out: [Higgs4L1file.root]
  make_plot:
    run: make_plot.cwl
    hints:
      reana:
        compute_backend: kubernetes
    in:
      DoubleMuParked2012C_10000_Higgs: >
        analyse_data/DoubleMuParked2012C_10000_Higgs.root
      Higgs4L1file: >
        analyse_mc/Higgs4L1file.root
    out: [mass4l_combine_userlvl3.pdf]
```

Figure 6. The workflow definition corresponding to the Higgs-to-four-lepton example analysis from Figure 5. Note the *compute_backend* hint provided by the user as to where each stop of the analysis is to be run. Each workflow step runs on different compute backend in this example. The REANA platform takes care of orchestrating jobs and moving inputs/outputs between platforms, as needed.

4 Declarative vs imperative analysis paradigm

The interconnection and orchestration of jobs running over different batch systems is a difficult task which is abstracted by the REANA platform, giving the user the choice of expressing 'what' the user wants to do, rather than requiring the user to write low-level glue code 'how' to do it. As we have seen in Figure 6, the choice of the compute backend was specified on one line. The declarative approach to data analyses is not only simplifying analysis code, as

there is no need to write specific orchestration details for each backend, but it also makes it easier to switch from one compute platform to another.

In order to appreciate this point, let us briefly have a look at how the REANA ↔ Slurm integration works. Firstly, the REANA system will launch jobs on the underlying compute backends in the name of the user. This requires to handle user authentication (using e.g. Kerberos) with possible reauthentications for longer running jobs. Secondly, the REANA system needs to translate job specifications for each backend, using their own language. In the example of Figure 5, the *analyse_data* step will run in HTCondor, *analyse_mc* in Slurm and *make_plot* in Kubernetes, each using different job submission protocols. Thirdly, each compute backend may have their own file storage systems. REANA needs to take care of staging in the input files and staging out the output files generated by each step. Last but not least, REANA also abstracts the task of status reporting, providing updates about job progress in a uniform manner for all three backends. If the user would like to use some computing platform without proper abstractions, all these tasks would have to be managed directly by the user.

The separation of "declarative" analysis programming from "imperative" orchestration of computing tasks allows researchers to focus fully on physics and leave mundane computing tasks to the workflow platform. The declarative programming approach thusly facilitates the understanding of the research pipeline and the portability of its various steps to different compute backends.

The hybrid analysis approach allows to take full advantage of different compute backends that are at the user's disposal. However, the penalty inherent in possible file transfer between compute backends is to be carefully considered; staging-in and staging-out of large files in between compute farms may unnecessarily penalise the user. The declarative analysis approach allows to rapidly test the pros and cons and the hybrid overhead for each particular analysis scenario.

5 Conclusions

We have enriched the REANA reproducible analysis platform with a possibility to run hybrid analysis pipelines. We have added support for dispatching jobs to the Slurm compute backend system that is widely used in HPC clusters. The present work enables users to run their analysis workflows on containerised Kubernetes clouds, on high-throughput HTCondor compute platform, and high-performance Slurm compute backends. The user can select a different desired compute backend for each task by providing a compute backend hint declaration to the workflow language. The workflow hints are then automatically handled by the REANA workflow orchestration system to dispatch jobs to various backends and to stage-in and stage-out the job inputs and outputs. The abstraction of the job orchestration allows to separate high-level physics analysis code from low-level compute platform orchestration code. We have described one example of a hybrid analysis workflow using the CMS open data Higgs-to-four-lepton analysis example. A detailed performance comparison of hybrid approach for complex physics analyses will be part of a future work.

References

[1] T. Šimko, L. Heinrich, H. Hirvonsalo, D. Kousidis, D. Rodríguez, "REANA: A system for reusable research data analyses", EPJ Web of Conferences 214, 06034 (2019), https://doi.org/10.1051/epjconf/201921406034

[2] X. Chen, S. Dallmeier-Tiessen, R. Dasler, S. Feger, P. Fokianos, J. .B. Gonzalez, H. Hirvonsalo, D. Kousidis, A. Lavasa, S. Mele, D. Rodríguez, T. Šimko, T. Smith, A. Trisovic, A. Trzcinska, I. Tsanaktsidis, M. Zimmermann, K. Cranmer, L. Heinrich, G. Watts, M. Hildreth, L. Lloret Iglesias, K. Lassila-Perini, S. Neubert, "Open is not enough", Nature Physics **15** 113–118 (2019). https://www.nature.com/articles/s41567-018-0342-2

[3] P. Amstutz, M. R. Crusoe, N. Tijanić (editors), B. Chapman, J. Chilton, M. Heuer, A. Kartashov, D. Leehr, H. Ménager, M. Nedeljkovich, M. Scales, S. Soiland-Reyes, L. Stojanovic, "Common Workflow Language, v1.0" Specification, Common Workflow Language working group, https://doi.org/10.6084/m9.figshare.3115156.v2

[4] K. Cranmer, L. Heinrich, "Yadage and Packtivity – analysis preservation using parametrized workflows", arXiv:1706.01878

[5] R. Maciulaitis, P. Brenner, S. Hampton, M. Hildreth, K. Hurtado Anapama, I. Johnson, C. Kankel, J. Okraska, D. Rodriguez, T. Šimko, "Support for HTCondor high-throughput computing workflows in the REANA reusable analysis platform", 15th IEEE eScience 2019 conference, San Diego, United States, 24–27 September 2019. CERN-IT-2019-004. http://cds.cern.ch/record/2696223

[6] N. Z. Jomhari, A. Geiser, A. A. Bin Anuar, "Higgs-to-four-lepton analysis example using 2011-2012 data", CERN Open Data Portal, 2017. DOI:10.7483/OPENDATA.CMS.JKB8.RR42