Abstract

These days workflows becoming more popular because of new techniques like distributed cloud based systems. This project is about integration of different sources of provenance data, generated by scientific workflows. This data can later be used to optimize the workflow. During the project a framework model for integration has been created. The outcome of the project is that the usage of an integrator hub as service is useful to gain integration of workflows logs. An example of this has been given in a Proof of Concept.
Acknowledgements

I would like to thank my supervisor Dr. Zhiming Zhao for his help understanding the subject matter. In addition, I would like to thank Spiros Koulouzis providing the support with the research environment and related services associated with the project.
Contents

1 Introduction 4

2 Research question 4

3 Background Information 4
  3.1 Complexity of workflows 4
  3.2 Types of workflows 4

4 Methodology 5
  4.1 Reference workflow 5
  4.2 Traceability of tasks 5
  4.3 Proposed solution 6

5 Experiment 6
  5.1 Results 7

6 Conclusion 8
  6.1 Discussion 8
  6.2 Future work 8

7 Appendix 9
  7.1 Integration server example 9
  7.2 Dump of the example workflow 9
1 Introduction

These days workflows becoming more popular because of new techniques like distributed cloud based systems.[1] Workflow management is there to control the flow of information. Where possible it uses automated scripted sources to make decisions.[2] Workflow management aims to create an abstract of more than one autonomous systems and their functions without having to touch or request modification. Workflow management relays on underlying services it’s application programming interface (API) which are seen as Input and Output. The service is a black box for the Workflow management system.

2 Research question

The main question for this research is:

How can different sources of provenance data generated by scientific workflows be integrated to allow analysis?

The research question can be divided into multiple sub-questions:

1. What main types of workflow exist?
2. What kind of system logs are available according to specific use cases?
3. How does the workflow system relate to system logs?

3 Background Information

This section covers background information about workflows which is relevant for the topic. This section provides an answer for sub-question 1.

3.1 Complexity of workflows

The complexity of workflows has increased due to the use of distributed cloud based systems. The factors that need to be taken into account are;

- Amount of distributed autonomous systems and their variety
- Availability or logs
  (Application, Syslog, Docker, PROV)
- Geographical and logical location
  (Local, Remote, Cloud, Containerized, ...)

The combination of these factors has increased complexity. Problems that may occur are e.g. different log formats, summarized and abstraction of information, IP addresses by name pacing, accessibility of log files and possible business and privacy policies based on different jurisdictions. These are aspects that have to be taken into account when integrating data provenance based on log files.

3.2 Types of workflows

There are many different types of workflows. These types are important to take into account if you want to reconnect a timeline. When we look at the aspect of ordering the tasks within a workflow, the following types can be defined;

Linear The simplest form of a workflow is the linear workflow where the steps are executed in a fixed sequence shown in figure 1. It is relatively easy to retrace a timeline afterwards because you do not have to take into account the circumstances such as skipped parts, repetition or iteration.

Figure 1: Schematic representation of a linear workflow
Recursive  In a recursive workflow, particular straps can be executed more than once. At some point in during the execution, there might be a call back to an earlier function in the flow. This could be a 'foreach element in list'-function. An example of this is shown in figure 2. Knowledge of this type is required because it could be relevant for log integration. The workflow execution can be found at multiple times in the log file of the remote service.

Parallel  In a parallel workflow, tasks are executed in parallel. This means that the output of one service is being processed in two or more new services, which are being executed at the same moment in time. Later in time there can be a merge again as shown in figure 3 where workflow service(WS) 2 and 3 are merged into WS 4.

Decision based  In a decision-based workflow, the actual execution of the steps is based on information processed during the execution of the workflow. This can be internal or external information factors. Based on this some steps can be for example excluded from the execution as shown in figure 4 where workflow service(WS) 3.1 has been skipped.

Hybrid  An hybrid form with a combination of the above mentioned.

4 Methodology

In this section, a proposed solution is discussed, in response to sub-question 3.

4.1 Reference workflow

As reference we used a workflow consisting of 3 components shown in figure 5. This workflow consists of different types of infrastructure from local, remote and distributed. The reference workflow has the following steps:

1. Apache Taverna calls D4Science and includes the URI of uvalight as datasource.
2. D4Science calls drip.uvalight.
3. Drip.uvalight pulls the data from this database and replies the request.
4. D4Science processes the data from drip.uvalight and replies the request to Apache Taverna.

4.2 Traceability of tasks

In the workflow as discussed in the previous section, it is hard to correlate and trace individual tasks in the workflow. The workflow manager is only partially able to meet this requirement since the tasks are running remote.

For example the steps 2 and 3 of the reference workflow(figure5) are not going through the manager and are thus hidden for workflow managing software like Apache Taverna.
Within the log files of the successive systems it is hard to relate the events of the task to the specific workflow execution. This is because workflows can be executed in parallel, scheduled on other systems.

4.3 Proposed solution

As a solution to the problem of traceability, we propose that it is possible to add an extra service with an integrator role. Figure 6, shows the proposed workflow that has been modified with this idea in mind.

The proposed workflow has the following steps:

1. **Apache Taverna** calls **Integrator** with the FLOWLABEL.
2. **Integrator** registers the initiation and replies with a timestamp.
3. **Apache Taverna** calls **D4Science** and includes the URI of uvalight as datasource. The URI includes FLOWLABEL.
4. **D4science** calls **drip.uvalight**.
5. **Drip.uvalight** pulls the data from this database and replies the request.
6. **D4science** processes the data from drip.uvalight and replies the request to **Apache Taverna**.
7. **Apache Taverna** calls **Integrator** with the FLOWLABEL and the reply from D4Science(step 6).
8. **Integrator** registers the flowlabel, and phrases the logfile URI from the reply. Then the **Integrator** calls **D4Science** for the logfile.
9. **D4science** replies the request with the logfile.
10. **Integrator** calls the logfile from drip.uvalight with the flowlabel.
11. **drip.uvalight** replies the request with the logfile. Optionally filtered on the flowlabel for privacy.

5 Experiment

To test the operation of the proposed solution of the previous section an experiment in the form of a proof of concept (PoC) has been used. This proof of concept and its results are described in this chapter.

In this solution there a flowlabel has been attached to the individual tasks within the workflow. This flowlabel makes it possible to relate events to a specific workflow execution.

In figure 7 is an Apache Taverna workflow based on the reference workflow of section 4.1. The variable "input_file_name" contains an uniform resource identifier (URI) to an UvA-Light VM running a database.
Figure 8 is an Apache Taverna workflow based on the proposed workflow described in section 4.3. This workflow has an integrator, namely BLAAUWGEERS.SU. This integrator registers the execution of the workflow at the moment of initiation. It also generates a timestamp which will be added to the flowlabel.

5.1 Results

The URL template of within Taverna will be look like:

```
http://dataminer-prototypes.d4science.org/wps/WebProcessingService?request=Execute&service=WPS&Version=1.0.0&gcube-token={TOKEN}&lang=en-US&Identifier={Identifier}&DataInputs=input_file_name={input_file_name}&flowlabel
```

And when we inspect the log files (figure 9 & 10) the workflow traffic is identifiable.

![Logfile from Intergrator server](image-url)

![Logfile from Uvalight Tomcat server](image-url)
6 Conclusion

The usage of an integrator hub as service is useful to gain integration of workflows logs. Different sources of provenance data generated by can be integrated by this.

There are different kind of workflows. They are Linear, Recursive, Parallel, Decision based and Hybrid.

The factors that need to be taken into account are: Amount of systems and their variety, Availability of logs and Location like Local, Remote, Cloud. The combination of these factors has increased complexity. Problems can be limited detailed logs, summarization and abstraction.

It is recommended to add a flowelabel and timestamp to the API calls in the Workflow Manager. This flowlabel can be used later to relate the workflow and the logs. The timestamp can be used to overcome time differences.

It is the role of the integrator to collect the logfiles other Workflows Services and to enrich them with relevant data based on the flowlabel.

Different sources of data generated by workflows can be integrated with an integrator hub. The collection of log files can be used to create a timeline on which analysis can be performed. The accessibility of log data is important for completeness of the picture.

6.1 Discussion

Log data might be influenced by policies, containerization, time-zones, namespace translation. The effect of this can be reduced by adding a flowlabel and timestamp to the requests. It is important to use a variable that is not yet used by the application. Some integrator functions might break when components of the workflow change.

6.2 Future work

The workflow from the experiment should be tested with different kind of Workflow managers then Apache Taverna. However, it should not make a difference for the result. The current example integrator is created for D4Science and Uvalight to prove the concept. There is improvement on the integrator needed to support more services.

Future research is needed to the integrator more resilient to workflow changes and in a distribution strategy for the integrator in order to limit the single point of failure.

References


7 Appendix

7.1 Integration server example

In the figure below an example of an Integration-server as integrator.

```php
<?php
require_once ("XML2Array.function.php") ;

if (!isset($_POST["f"])) {
    header("HTTP/1.1 405 Method Not Allowed");
echo "Error: Post Expected";
exit();
}

$fname["xml"] = "post." . $_GET["flowlabel"] . "." . time() . ".xml";
$fname["log"] = "post." . $_GET["flowlabel"] . "." . time() . ".log";
$fname["data"] = "post." . $_GET["flowlabel"] . "." . time() . ".data";

$fdata = file_get_contents(’php://input’);
file_put_contents($fname["xml"], $fdata);

#get the file
$flowout = file_get_contents($fname["xml"]);
#Replace the illegal character before parsing to the simple object.
$flowbuffer = str_replace(": ", "−", $flowout);
#Convert the xml−string value to an array via a simple xml object.
$flowxml = simplexml_load_string($flowbuffer);
$flowarray = array($flowxml->getName() => $flowarray);

#adapt the wps data.
$flowwps = $flowarray["wps−ExecuteResponse"]['wps−ProcessOutputs']["wps−Output"]['wps−Data']["wps−ComplexData"]['ogr−FeatureCollection']
[gml−featureMember]["ogr−Result"];

#grep the location of d4s
$flowdfs["data"] = str_replace("http−", "http: ", $flowwps[0]["d4science−Data "]);
#Log of the computation
$flowdfs["log"] = str_replace("http−", "http: ", $flowwps["d4science−Data"])

file_put_contents($fname["log"], fopen($flowdfs["log"], 'r'));
file_put_contents($fname["data"], fopen($flowdfs["data"], 'r'));
?>
```

7.2 Dump of the example workflow

In the figure below a wireshark capture can be found with the flowlabel

`ablaauwgeers@desktop−30:~$ tshark −r capt1.pcapng −Y "http"
821 4.466783271 145.100.102.105 81.4.107.121 HTTP 245 GET /rpa/
flowlabel.php?flowlabel=0011T
HTTP/1.1
823 4.474268876 81.4.107.121 145.100.102.105 HTTP 266 HTTP/1.1 200 OK
(text/html)
830 4.476631015 145.100.102.105 81.4.107.121 HTTP 769 GET /wps/WebProcessingService?request=Execute&service=WPS&Version=1.0.0&gcube
token=591e9609%2D02df%2D453d%2Bff2%2D272ef0ac91ac%2D843339462&lang=en−US&Identifer=org%2Egcube%2Edataanalysis%2Ewps%2Estatisticalmanager%2Esynchserver%2Emappedclasses%2Etransducerers%2EARGO%5FFEBRUARY&DataInputs
=input_file_name=http%3A%2F%2Fdrip%2Evlan400%2Euvalight%2Enet%3A8081%2Fcue%2Frest%2Fargo%2Fget%3Fgeospatial%5Flat%5Fmin%3D38%2E000%26geospatial%5Flat%5Fmax%3D38%2E200%26geospatial%5Flon%5Fmin%3D147%2E000%26geospatial%5Flon%5Fmax%3D147%2E100%26flowlabel
%5D0011T HTTP/1.1`