PLAS
Platformed Task

Andrea Detti: andrea.detti@uniroma2.it
Luca Petrucci: luca.petrucci@uniroma2.it
Ludovico Funari: ludovico.funari@uniroma2.it
PLAS – Platformed Tasks

- GÉANT Cloud Flow (GCF) provides researchers with the ability to run workflows composed of analysis tasks exploiting GÉANT cloud facilities.
- The PLAS project aims to extend GCF services with platformed-tasks.
PLAS Goal

Enable researchers to execute single tasks not only within Containers, but also within overlay platforms installed on-demand during the task deployment phase.

- Parallel computation
The Task Execution Service (TES) API is a standardized schema and API for describing and executing batch execution tasks. A task defines a set of input files, a set of containers and commands to run, a set of output files.
TESK - TES on Kubernetes

An implementation of a task execution engine based on the TES standard running on Kubernetes.

https://github.com/elixir-cloud-aai/TESK
TESK Lifecycle

1. TESK-API consumes TES request and translates them to K8s job
TESK Lifecycle

1. TESK-API consumes TES request and translates them to K8s job
2. Taskmaster creates *filer* pod which creates a *PVC* to mount as scratch. All files are downloaded into the locations specified in the TES request

https://github.com/elixir-cloud-aai/TESK
1. TESK-API consumes TES request and translates them to K8s job
2. Taskmaster creates *filer* pod which creates a *PVC* to mount as scratch. All files are downloaded into the locations specified in the TES request
3. After the *filer* has finished, the taskmaster execute the *executor* as pod
1. TESK-API consumes TES request and translates them to K8s job
2. Taskmaster creates *filer* pod which creates a *PVC* to mount as scratch. All files are downloaded into the locations specified in the TES request
3. After the *filer* has finished, the taskmaster execute the *executor* as pod
4. When the executor is terminated, the *filer* is called once more to process the outputs and push them to remote locations from the PVC
TESK - Extension

The figure shows our extended architecture of TESK to support Platformed Task

https://github.com/elixir-cloud-aai/TESK
1. TESK-API consumes TES request and translates them to K8s job.
2. Taskmaster creates *filer* pod which creates a *PVC* to mount as scratch. All files are downloaded into the locations specified in the TES request.
3a. After the *filer* has finished, the taskmaster deploys the *platform* with *Helm*.

https://github.com/elixir-cloud-aai/TESK
1. TESK-API consumes TES request and translates them to K8s job
2. Taskmaster creates *filer* pod which creates a *PVC* to mount as scratch. All files are downloaded into the locations specified in the TES request
3a. After the *filer* has finished, the taskmaster deploys the platform with helm
3b. After the *platform* is ready, the taskmaster runs the *executor* as pod
TESK Extended Lifecycle

1. TESK-API consumes TES request and translates them to K8s job
2. Taskmaster creates *filer* pod which creates a *PVC* to mount as scratch. All files are downloaded into the locations specified in the TES request
3a. After the *filer* has finished, the taskmaster deploys the platform with helm
3b. After the *platform* is ready, the taskmaster runs the *executor* as pod
4. The *filer* saves the outputs
**TESK - cwl-TES**

*cwl-tes* component is used to submit tasks to a TES server: it translates the *task* from CWL to JSON

[Diagram of TESK process]

[GitHub link to TESK project](https://github.com/elixir-cloud-aai/TESK)
CWL – Task

hashsplitter-sha.cwl.yml

```yaml
1  cwlVersion: v1.0
2  class: CommandLineTool
3    hints:
4      - class: DockerRequirement
5        dockerPull: kubler/openssl
6    inputs:
7      - id: input
8        type: File
9        doc: "original content"
10       inputBinding:
11          position: 1
12    outputs:
13      - id: output
14        type: stdout
15        stdout: sha
16        baseCommand: ["openssl", "dgst"]
17        arguments: ["-sha1"]
```

input-file.json

```json
1  {
2    "input": {
3      "class": "File",
4      "location": "ftp://10.0.0.10/files/input.txt"
5    }
6  }
```
Extension Challenges
Extend the cwl-tes component to add the support to new Common Workflow Language (CWL) HelmRequirement class.

```json
{
  "cwlVersion": "v1.0",
  "class": "CommandLineTool",
  "requirements": {
    "class": "HelmRequirement",
    "chartRepo": "https://link_to_horovod_chart_repo/charts",
    "chartVersion": "3.0.0",
    "chartName": "horovod",
    "executorImage": "horovod:latest"
  }
}
```

**Note:** Supported chart repository link and his specific executor image is needed.
Extend the TESK-API component to forward changes and generate the taskmaster pod with additional volume and extra necessary details.
Improve the capability of the taskmaster to deploy the platform using Helm, in addition to the creation of input output and executor.
TESK - Extension Challenge (IV)

- Knowledge of the *horovod* platform itself before the implementation with CWL workflow
- Developing the specific *horovod* executor image
- Integrate all with the extended components

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>task-a963dd6a--1-8zhrq</td>
<td>1/1</td>
<td>Running</td>
</tr>
<tr>
<td>task-a963dd6a-ex-00--1-57c9w</td>
<td>0/1</td>
<td>Completed</td>
</tr>
<tr>
<td>task-a963dd6a-inputs-filer--1-hmmqd</td>
<td>0/1</td>
<td>Completed</td>
</tr>
<tr>
<td>task-a963dd6a-outputs-filer--1-phhc6</td>
<td>0/1</td>
<td>Completed</td>
</tr>
<tr>
<td>task-a963dd6a-platform-horovod-0</td>
<td>1/1</td>
<td>Terminating</td>
</tr>
<tr>
<td>task-a963dd6a-platform-horovod-1</td>
<td>1/1</td>
<td>Terminating</td>
</tr>
<tr>
<td>tesk-api-d4f8579c8-jjkv5</td>
<td>1/1</td>
<td>Running</td>
</tr>
</tbody>
</table>
PLAS - Repositories

Make a *github* chart repository with the compatible charts (at the moment *horovod* and *spark*) and the corresponding executor.
Repositories

We have a GitHub repository for each module:

- cwl-TES
- TESK-API
- Taskmaster
- Platform helm charts:
  - horovod
  - spark

We also have developed the Docker Images of:

- TESK-API
- Taskmaster
- horovod
- spark
Conclusion and Future works

- We are able to perform a platformed task
- We have implement the horovod and spark use cases
- Platform security issues need to be evaluated
DEMO