Team 5: Integration
CS5604: Information Storage and Retrieval

Advised by Dr. Edward Fox
Blacksburg, VA 24061
Dec. 06, 2022

Aaron Travasso | Anmol Shukla | Harish Babu | Pallavi Sisodiya | Yuze Li
Subject Matter Expert: Dhanush Dinesh
Agenda

1. Workflow automation
2. Workflow CRUD interface
3. Developer Operations
4. Future work
Containerization of bunch of services
- For developers, hard to maintain dependencies
- For end-users, hard to know what they need

Workflow automation helps end-users by abstracting/hiding complex low-level services.
- For developers, each service in workflow is containerized, easy to maintain
- For end-users, they only see the predefined workflow, don’t care about what’s inside the black box
Design

1. Developers generate grammar: reasoner reads postgres table and generate cfl file.
2. Users generate DAG with environment variables, get the workflow ID.
3. User execute DAG based on the workflow ID.
Implementation

- Context-Free Grammar (CFG)
  - Recursively generate patterns
  - Represents goals and services

- Reasoner APIs
  - Generate grammar (CFG file)
  - Generate workflow
    - Required input: `<goal_id>`
    - Optional input: `<environment (ETD_ID)>`
    - Output: `<workflow_id>`
  - Execute workflow
    - Required input: `<workflow_id>`
    - Optional input: `<service_id(s)>`
    - Output: `<log_url>`, `<status_key>`
  - Check workflow status
    - Required input: `<status_key>`
    - Output: `<status_metadata>`
    - G3->S1S2 epsilon

\[
S \rightarrow G_1 \\
G_3 \rightarrow S_2 G_2 \\
G_2 \rightarrow S_1 G_1 \\
G_1 \rightarrow \epsilon
\]
Implementation

- Workflows implemented
  - Indexing
  - Object detection
Implementation

- Workflows implemented
  - Segmentation, summarization, classification
Demonstration

Team-4’s segmentation

- File Team5INTdemo.mp4 - Team 4's Segmentation Demo Video
How do I integrate this?

- Helper code to generate reasoner [6]
  - generateWorkflow()
  - runWorkflow()
  - getWorkflowStatus()
- List of container registries [2]
Workflow CRUD Interface
Workflow Automation
Workflow automation definition page

In workflow automation, we need to visualize in terms of services, inputs and outputs. Each service is a unit of computation, which takes some inputs and generates outputs. For example, if one service is a file that extracts URLs from a text file, it is a file written in Python, Java etc. That does this computation. But, we still need to provide an input to this unit, and the inputs and outputs come in.

Let's not think of goals in the traditional sense here, but as what we can pass as inputs or outputs.

<table>
<thead>
<tr>
<th>goal_name</th>
<th>goal_description</th>
<th>goal_format</th>
<th>file_location</th>
<th>environment_variable</th>
<th>owned_by</th>
</tr>
</thead>
<tbody>
<tr>
<td>clean_and_parse_output</td>
<td>CLEAN PDF</td>
<td>&lt;Directory&gt;</td>
<td>/mnt/data/team5/cleaned_chapters</td>
<td>CLEANED_ENV</td>
<td>INT</td>
</tr>
<tr>
<td>classification_output</td>
<td>CLASSIFY_CHAP</td>
<td>&lt;Directory&gt;</td>
<td>/mnt/data/team5/classified_chapters</td>
<td>CLASSIFICATION_ENV</td>
<td>INT</td>
</tr>
<tr>
<td>fetch_etd_output</td>
<td>Stores fetched ETD for segmentation, summ</td>
<td>&lt;Directory&gt;</td>
<td>/mnt/data/team5/segmented</td>
<td>ORIGINAL_DATASET</td>
<td>INT</td>
</tr>
</tbody>
</table>
When we think of workflow automation, we need to visualize in terms of services, inputs and outputs. Each service is a unit of computation, which takes some inputs and generates outputs. For example, if we have a service that extracts URLs from a text file, the service is a file written in Python, Java, etc. that does this computation. But, we still need to provide an input to this unit of computation. This is where the inputs and the outputs come in.

A goal is an output or an input. Please don't think of goals in the traditional sense here, but as what we can pass as inputs or outputs.

<table>
<thead>
<tr>
<th>Item</th>
<th>goals</th>
<th>services</th>
<th>reasoner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lid</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>goal_name</td>
<td>goal_description</td>
<td>goal_format</td>
<td>file_location</td>
</tr>
<tr>
<td>73</td>
<td>clean_and_parse_output</td>
<td>CLEAN_PDF</td>
<td>/mnt/data/team5/cleaned_chapters</td>
</tr>
<tr>
<td>90</td>
<td>classification_output</td>
<td>CLASSIFY_CHAP</td>
<td>/mnt/data/team5/classified_chapters</td>
</tr>
<tr>
<td>71</td>
<td>fetch_etd_output</td>
<td>Stores fetched ETD for segmentation, summ, class, etc.</td>
<td>/mnt/data/team5/segmentation_input</td>
</tr>
<tr>
<td>112</td>
<td>segmentation_output</td>
<td>A Place to store the segmented chapters of an</td>
<td>/mnt/data/team5/segmented_chapters</td>
</tr>
</tbody>
</table>
When we think of workflow automation, we need to visualize it in terms of computation, which takes some inputs and generates outputs. For example, if we have a service that extracts URLs from a text file, the service will have an input (the text file) and an output (the list of URLs).

A goal is an output or an input. Please don’t think of goals in the traditional sense of a goal as something you want to achieve. Goals are the building blocks of workflow automation. They can be inputs to other goals, or outputs from other goals.

Here is a table of goals:

<table>
<thead>
<tr>
<th>Item</th>
<th>goal_id</th>
<th>goal_name</th>
<th>goal_description</th>
<th>goal_format</th>
<th>file_location</th>
<th>environment_variable</th>
<th>owned_by</th>
</tr>
</thead>
<tbody>
<tr>
<td>73</td>
<td>73</td>
<td>clean_parse_output</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>90</td>
<td>classification_output</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>71</td>
<td>fetch_etd_output</td>
<td>Stores fetched etd input</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A Place to store the segmented chapters of an unstructured dataset:

/mnt/data/team5/segmented_chapters
03
Developer Operations
TEAM-1
- PostgreSQL
- Python
- Virtuoso

TEAM-2
- Jupyter Notebook
- Elasticsearch

TEAM-3
- PyTorch with CUDA

TEAM-4
- PyTorch & other dependencies - Segmentation
- PyTorch & other dependencies - Summaries
- PyTorch & other dependencies - Classification
Cluster statistics

- 45 Deployments
- 40 Ingresses
- 39 Pods
- 180 Services
- 13 Secrets
- Volumes
  - CEPH\(^1\): etd-cephfs-cs5604-pv:

1. Although the total capacity is 19TB, due to it being a shared space, it was recommended to use no more than 4TB.
CONTINUOUS INTEGRATION AND CONTINUOUS DEPLOYMENT
CI/CD has been implemented for Teams 1, 2, 3 and frontend.
- GitLab Runner is hosted on containers.cs.vt.edu
- All Docker Images are located in code.vt.edu Container Registry.
Frontend CI/CD Analytics

**Overall statistics**

- Total: 224 pipelines
- Successful: 191 pipelines
- Failed: 12 pipelines
- Success ratio: 94.09%

![Pipeline durations for the last 30 commits](image)
04
Future Work
• Reasoner Optimization
  ○ Saving/loading intermediate data. For example, if two workflows overlap in some steps, the succeeding workflow can directly read the intermediate output produced by the former workflow, instead of executing the workflow end-to-end.

1st time executing:

2nd time executing:
System Performance Trade-off

- Observation: Image provisioning & container spawning up take constant time, service execution time varies.
- Optimization: Merging several insignificant/lightweight services into one monolithic service.
- Trade-off: Decoupled services VS. user response time.
Future work

Gitlab issue board for issues for future teams [3]
References

THANK YOU!

Questions?