A Workflow Management Engine in CORD

Illyoung Choi
iychoi@email.arizona.edu
University of Arizona
A Workflow?

This differentiates a workflow from a general program

Is a series of tasks in order to accomplish a repeatable business objective with details on when, how and what work is to be done.

Task = (When ⇒ How ⇒ What)
    = (Event ⇒ Action ⇒ Result)
Current Workflows in CORD

- Implemented as a **XOS Synchronizer**
  - **Data-models** (data)
  - **Model-policies** (model-event handlers)
  - **Event-steps** (external-event handlers)
- Event handlers define **when, how, and what**
- A container per workflow
  - Scalability & Isolation

**Execution Flow of AT&T Workflow**
- Two execution paths
- inter-workflow calls (dot arrows)
Difficulties & Limitations

• Development
  • Inconsistent technologies & interfaces for event-handling
  • Seemingly fragmented codes
  • Manual workflow state management
  • Possible race conditions & loops

• Management
  • Difficult to understand execution flows & relations
  • Difficult to monitor workflow state
Execution Flow At First Glance
Event-steps vs. Model-policies

Periodic check for updates (5 sec)

```python
class ONUEventStep(EventStep):
    topics = ['onu.events']
    technology = 'kafka'

    def __init__(self, *args, **kwargs):
        super(ONUEventStep, self).__init__(*args, **kwargs)

    def process_event(self, event):
        value = json.loads(event.value)
        ...
```

```python
class AttWorkflowDriverServiceInstancePolicy(Policy):
    model_name = 'AttWorkflowDriverServiceInstance'

    def handle_create(self, self, si):
        ...

    def handle_update(self, self, si):
        ...

    def handle_delete(self, self, si):
        ...
```
Possibly...

Race Conditions

Loops
The Pilot Engine

- Based on **Apache Airflow**
  - An open-source workflow management platform by Airbnb

- Development
  - Simple & consistent event-handler interfaces
  - Execution flow is clearly described
  - State management
  - Can avoid race conditions using Pools (like mutex)
  - Can find loops via graphs

- Management
  - Visualize state, flows and relations of workflows
  - Scalable (using kubernetes/celery)
Design of the Pilot Engine

- Probe (producer) Interface
- Workflow (subscriber) Interface
- Workflow Controller
- Probes
- Workflow #1
- Workflow #2
- Airflow
- Airflow Extensions
Workflow Controller

• Bridge the gap between **CORD** and **Airflow**
  • **OLTP** (Online Transactional Processing)
    vs. **OLAP** (Online Analytical Processing)
  • OLTP ⇒ Short transactions (e.g., CRUD)
  • OLAP ⇒ Periodical batch processing (e.g., Hadoop analysis)
  • **Run a transaction as a workflow instance**

• Workflow management
  • Launch new workflow instances
  • Monitor state of workflow instances

• Event routing
  • Route events to workflow instances
Code Changes

```
onu_event_sensor = CORDEventSensor(
    task_id='onu_event_sensor',
    topic='onu.events',
    key_field='serialNumber',
    controller_conn_id='local_cord_controller',
    poke_interval=5,
    dag=dag_att
)
onu_event_handler = CORDModelOperator(
    task_id='onu_event_handler',
    python_callable=on_onu_event,
    data_models=['ATTServiceInstanceModel'],
    cord_event_sensor_task_id='onu_event_sensor',
    dag=dag_att
)
onu_event_sensor >> onu_event_handler
```

```
When & how

<table>
<thead>
<tr>
<th>When</th>
<th>How</th>
<th>What</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Step</td>
<td>Model Policy</td>
<td>Data Model</td>
</tr>
<tr>
<td>OR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event Sensor</td>
<td>Model Operator</td>
<td>Data Model</td>
</tr>
</tbody>
</table>

Explicit flow definition
```
Design Changes

XOS Synchronizer

The Pilot Engine
Monitoring
Limitations

• Performance
  • Slow polling-based event detection*
  • Slow task scheduling*
• Scalability
  • Single point of failure
  • Unscalable Airflow UI (Web admin)*
• Usability
  • Annoying workflow registration
  • Annoying restriction in programming
    (e.g., a single python file per a workflow)

Related to Airflow’s target market, “an orchestrator for **OLAP**”

**OLAP**
(Online Analytical Processing): periodic, long-running batch jobs
Future Workflow Management Engine in CORD

• **Best assets of the pilot engine**
  • Simple & consistent event handlers
  • Simple state management & flow control
  • Task scheduling using Pools to avoid race conditions
  • Visualizations for monitoring

• **Supplementary features to the pilot engine**
  • Automated loop detection & race condition
  • Fast event detection (event-driven or short polling period)
  • Fast task-scheduling
  • High availability & Scalability
  • A container per workflow (Like XOS Synchronizer)
  • Simple workflow registration at runtime
  • Workflow code packaging for deployment
Best of Both Worlds

- **XOS Synchronizer**
  - **Separation** of concerns

- **The Pilot Engine**
  - **Explicit** and **connected** concerns
Thank You