Containerized Workflow Scheduling

Research Project 1
Project #71

Isaac Klop
July 5, 2018

Supervisor: dr. Z. Zhao
University of Amsterdam
Introduction - Workflows

- Nodes represent tasks
- Edges represent dependencies

**Figure 1:** Example workflow
Introduction - Workflow Management Systems

- Used to manage/execute workflows
- Automation
- Failure recovery
- Map tasks to resources
- Examples:
  - Pegasus [1]
  - Taverna [2]
Introduction - Tasks as Containers

- OS-level Virtualization
- Lightweight
- Stand-alone

Figure 2: Example of binaries packaged with their dependencies in a container [3]
Introduction - Container Orchestration

- Containers at scale
- Cluster of multiple nodes
- Automates scheduling, deployment and management of containers
- Examples:
  - Docker Swarm [4]
  - Kubernetes [5]

**Figure 3:** Example of a cluster with 3 worker nodes.
Problem statement - Combining Workflows and Container Scheduling

- Find node for container
- Queue is FIFO
- Context of task is lost
- No dependencies
- Ordering/Dependencies on higher level
How can we order the execution of a containerized workflow on a container scheduler?
• Argo - Container-native workflow engine for Kubernetes [6]
• Apache Airflow - Plugin for Kubernetes (in development) [7]
• Makeflow on Mesos by Zheng et al. [8]
Method

1. Design a workflow with a critical path
2. Run workflow on container schedulers
   - Two container scheduling algorithms: Docker Swarm and Kubernetes
   - Two workflow scheduling algorithms: Critical path and Batch
3. Measure total execution time
Method - The Workflow

Figure 4: Test Workflow indicating processing times and resources.
Method

- Infinite resources: $5+20+90+5=120$ seconds
- Constrained resources:
  - Swarm: 5 nodes $\times$ 1 GB RAM
  - Kubernetes: 4 nodes $\times$ 1 GB RAM
- Assuming no overhead:
- Depending on the ordering of tasks

**Table 1**: Lowest/Highest possible total execution times assuming no overhead

<table>
<thead>
<tr>
<th>Scheduler</th>
<th>Lowest</th>
<th>Highest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swarm</td>
<td>120s</td>
<td>160s</td>
</tr>
<tr>
<td>Kubernetes</td>
<td>130s</td>
<td>180s</td>
</tr>
</tbody>
</table>
Method - Order the Execution

- Submit containers in order
  - Scheduler queue is not FIFO
  - Seemingly random
- Kubernetes:
  - Priority flag
- Swarm:
  - No priority flag
  - Hold back part of tasks
Figure 5: Average execution time of the Workflow on Swarm
Figure 6: Average execution time of the Workflow on Kubernetes
Conclusion

- Scheduling queue is not FIFO
- Execution time is erratic
- Critical path slightly lower execution times
Discussion

- Container schedulers lack features
- Kubernetes priority flag does pre-emption
- Interface between Workflow Management System and Container Scheduler
  - Monitoring
  - Active re-ordering
- More scheduling algorithms
Questions?


