



VUSE Summer Research 2021

High Performance CPS Co-Simulations



Udit Malik

Pls: Dr. Janos Sztipanovits
Dr. Yogesh Barve



VANDERBILT



UNIVERSITY

ISIS
Institute for Software
Integrated Systems

Introduction

- **Cyber-physical systems (CPS):** deeply intertwined physical and software components
- **Co-simulations:** enable better understanding of complex scenarios arising from CPS

Problems:

- Co-simulations are compute-intensive and execute slowly in traditional development and server environments
- Performance is affected by available platform and resources
- Cloud computing provides elastic resources for scaling simulations across large clusters

Proposal:

- Argo Workflows!
- Leveraging Kubernetes Cloud Deployment
- Offers high-performance execution of the co-simulations
- Easy integration of highly parallel jobs, and running CI/CD pipelines

This summer, I built: A full-stack web service/REST API from scratch, that automates the workflow submission process from end to end for the user!

Core technologies: AngularJS (frontend), Java with Spring Boot (backend)

Docker (containerization), Kubernetes (container management)

PostgreSQL (database management), Argo Workflows (workflow

management)

Other: MinIO (S3 object storage), Postman (API endpoint testing)

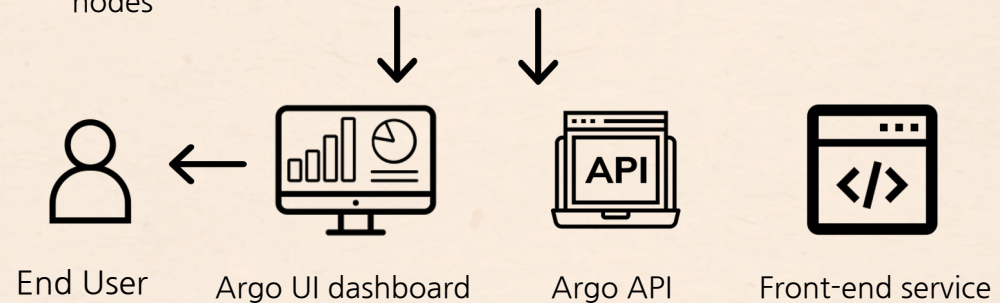
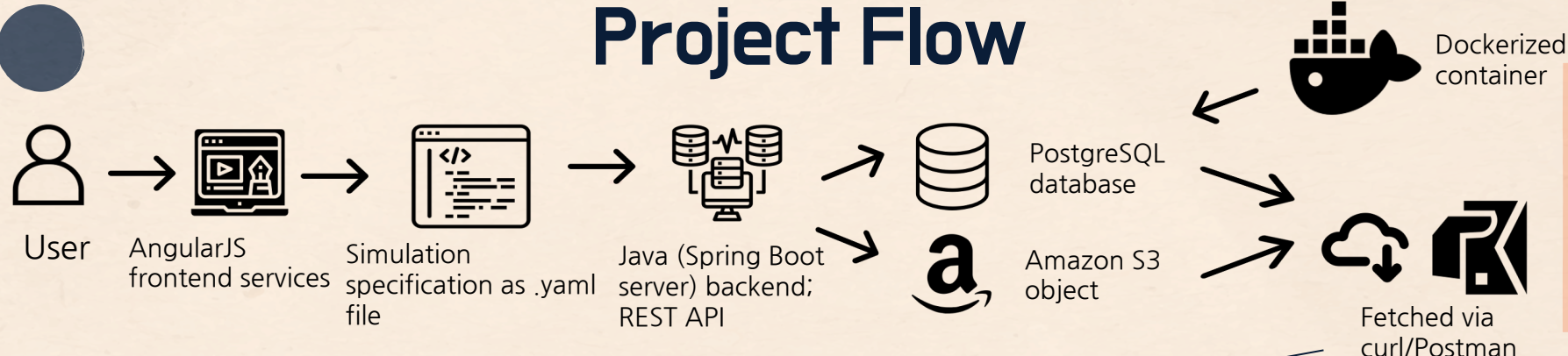
What exactly is a simulation/workflow?

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: coinflip-recursive-
spec:
  entrypoint: coinflip
  templates:
  - name: coinflip
    steps:
    - - name: flip-coin
      |   template: flip-coin
    - - name: heads
      |   template: heads
      |   when: "{{steps.flip-coin.outputs.result}} == heads"
    - - name: tails
      |   template: coinflip
      |   when: "{{steps.flip-coin.outputs.result}} == tails"

  - name: flip-coin
    script:
      image: python:alpine3.6
      command: [python]
      source: |
        import random
        result = "heads" if random.randint(0,1) == 0 else "tails"
        print(result)

  - name: heads
    container:
      image: alpine:3.6
      command: [sh, -c]
      args: ["echo \"it was heads\""]
```

Project Flow



Sample Demo

The image shows a Visual Studio Code editor window with a workflow definition in the Explorer and a SimpleScreenRecorder window in the foreground. The workflow definition is as follows:

```
example-workflows > | coinflip.yml | spec | [templates > | 2 | ] name
1 apVersion: argoproj.io/v1alpha1
2 kind: workflow
3 metadata:
4   generateName: coinflip-recursive-
5 spec:
6   entrypoint: coinflip
7   templates:
8     - name: coinflip
9       steps:
10        - name: flip-coin
11          template: flip-coin
12        - name: heads
13          templates: heads
14          when: "{{steps.flip-coin.outputs.result}} == heads"
15        - name: tails
16          templates: coinflip
17          when: "{{steps.flip-coin.outputs.result}} == tails"
18
19     - name: flip-coin
20       script:
21         image: python:alpine3.6
22         command: [python]
23         sources:
24         - import random
25           result = "heads" if random.randint(0,1) == 0 else "tails"
26           print(result)
27
28     - name: heads
29       container:
30         image: alpine:3.6
31         command: [sh, -c]
32         args: ["echo \"It was heads!\""]
33
```

The SimpleScreenRecorder window shows the following information:

- Recording: Pause recording
- Schedule: (inactive) Activate schedule Edit schedule
- Enable recording hotkey Enable sound notifications
- Hotkey: Ctrl + Shift + Alt + Super + R
- Information: Total time: 0:00:01, FPS in: 31.91, FPS out: 0:00, Size in: 1920x1080, Size out: 1920x1080, File name: simp...r.mp4, File size: 333 KiB, Bit rate: 0 bit/s
- Preview: Preview frame rate: 30, Note: Previewing requires extra CPU time (especially at high frame rates).
- Log: [X11InputSink] Detecting screen configuration..., [X11InputSink] Screen 0: x1 = 0, y1 = 0, x2 = 1920, y2 = 1080, [PageRecord-StartInput] Started input., [X11Input-InputThread] Input thread started.

Recursive coin flip
example run

Lessons, Challenges, and Successes



IDEATION

Understanding workflow
Conceptualization
Brainstorming

Prioritizing tasks
Feedback

TIME MANAGEMENT



ASSESSMENT

Testing
Experimentation

New languages
New frameworks
Integration

PROGRAMMING

