Events as a Basis for Workflow Scheduling

David Marchant
Premable

- Part of the MUMMERING Project
- MUltiscale, Multimodal, and Multidimensional imaging for EngineeRING
- Marie Skłodowska-Curie Innovative Training Network (Grant Agreement no. 765604)
Scientific Workflow Management Systems

- Utilise DAGs or logical equivalent
- Very strong predictive system
- But also very limited adaptability
Top-down vs Bottom-up

Traditional, Static, Top-Down
- User definitions
- Controller
- DAG
- Step
- Step
- Step
- Job
- Job
- Job
- Job
- Job
- Job
- Resources

New, Dynamic, Bottom-Up
- DAG
- Step
- Step
- Step
- Job
- Job
- Job
- Job
- Job
- Job
- Resources
- User definitions
- Trigger
- Controller
- Trigger
- Controller
- Trigger
- Controller
- User definitions

VS
Managing Event Oriented Workflows

Rules-based system for isolated job scheduling

Composed of Patterns and Recipes

Workflows as an emergent property

Workflow structure can be altered by adding, cancelling or modifying jobs or monitoring structures
mig_meow

- Python library for building MEOW objects
- Users define *Recipes* (the code to run) ... or *Patterns* (The conditions when to run)


together these form a *Rule* (Scheduling in response to events)

```python
input_file: infile
input_paths:
  - initial_data/*
output:
  outfile: '{VGRID}/int_1/{FILENAME}'
parameterize_over: {}
recipes:
  - append_text
variables:
  extra: This line is overridden
```
Where can it be used?

• Minimum intrusion Grid

• Integrated into cloud-based Jupyter hub instances

• Also available as a stand alone package in any Python3 environment via mig_meow library

• Provides Workflow_Runner
Using the WorkFlow Widget to Predict
Using the Monitor Widget to Track

```python
In [7]: meow.create_monitor_widget(vgrid='test_vgrid')
```

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Status</th>
<th>Created at</th>
<th>+</th>
<th>-</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>40_1_27_2020___14_59_48_mgrid.test0</td>
<td>FAILED</td>
<td>2020-01-27 14:53:48</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>39_1_27_2020___14_44_48_mgrid.test0</td>
<td>FAILED</td>
<td>2020-01-27 14:44:48</td>
<td>+</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>38_1_27_2020___14_10_58_mgrid.test0</td>
<td>FAILED</td>
<td>2020-01-27 14:16:58</td>
<td>+</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>37_1_27_2020___13_41_49_mgrid.test0</td>
<td>FAILED</td>
<td>2020-01-27 13:41:49</td>
<td>+</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>36_1_27_2020___13_35_18_mgrid.test0</td>
<td>FAILED</td>
<td>2020-01-27 13:35:18</td>
<td>+</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>35_1_27_2020___13_16_46_mgrid.test0</td>
<td>FINISHED</td>
<td>2020-01-27 13:16:46</td>
<td>+</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>34_1_27_2020___13_9_43_mgrid.test0</td>
<td>FINISHED</td>
<td>2020-01-27 13:09:43</td>
<td>+</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>33_1_27_2020___13_6_27_mgrid.test0</td>
<td>FINISHED</td>
<td>2020-01-27 13:06:27</td>
<td>+</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>32_1_27_2020___12_50_12_mgrid.test0</td>
<td>FINISHED</td>
<td>2020-01-27 12:50:12</td>
<td>+</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>31_1_27_2020___12_43_36_mgrid.test0</td>
<td>FINISHED</td>
<td>2020-01-27 12:43:36</td>
<td>+</td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>
Using the Report Widget to Reflect
Workflow Runner Structure

- Designed to mimic MiG, albeit in a reduced capacity as we don’t have a whole grid to manage

- Originally intended illustration for users who lacked access to MiG so inherits structure warts and all
Notes on the timing tests

- All tests run repeated 10 times with mean taken as final result
- Run for 10 to 500 jobs
- Run both on a Laptop and ‘Threadripper’ desktop for scientific processing

- Threadripper:
  - 16 cores (32 hyperthreads)
  - Threadripper 1950X processor at 3.4GHz
  - 112 GB RAM
- Laptop:
  - 4 cores (8 hyperthreads)
  - i7-8550U at 1.8GHz
  - 8 GB RAM

- As tests are not explicitly parallelised (e.g only ever one scheduler), main performance difference is as desktop vs laptop
Slurm as a Baseline

- Not a fully fledged SWMS, but MEOW is currently light on non-scheduling features
- Widely adopted, both by researchers directly and within SWMSs. If we can get similar performance then MEOW is acceptable
- Main difference is Slurm scheduling is not ongoing
Slurm as a Baseline

![Slurm scheduling overheads on the Threadripper](image)

- sbatch
- sbatch_sequential

Number of jobs scheduled vs Time taken (seconds)
Five Benchmarks

- Each test will schedule n jobs
1) Single Pattern Single File Parallel (SPSFP)
2) Multiple Patterns Single Files (MPSF)
3) Single Pattern Multiple Files (SPMF)
4) Multiple Patterns Multiple Files (MPMF)
5) Single Pattern Single File Sequential (SPSFS)
MEOW on the WorkflowRunner

![WorkflowRunner scheduling overheads on the Threadripper](image)

**mig_meow WorkflowRunner scheduling overheads on the Threadripper**

- **SPSFS**
- **SPSF**
- **sbatch**
- **MPSF**
- **MPMF**
- **sbatch_sequential**
- **SPMF**

**Time taken (seconds)**

**Number of jobs scheduled**

- 0
- 10
- 20
- 30
- 40
- 50
- 60
- 70
- 80
- 90
- 125
- 150
- 175
- 200
- 250
- 300
- 400
- 500

**14/11/22 SC22 | Dallas, TX | hpc accelerates.**
MEOW on the WorkflowRunner

![Graph showing mig_meow WorkflowRunner scheduling overheads on the Threadripper](image)

- **Time taken (seconds)**
- **Number of jobs scheduled**

Legend:
- SPSFS
- SPSFP
- sbatch
- sbatch_sequential

SC22 | Dallas, TX | hpc accelerates.
Scalability on the WorkflowRunner

![Graph showing scalability analysis](image)
Key Timings Takeaways

**The Good**
- WorkflowRunner generally beats Slurm. ~2.5x speedup in non-MPMF
- WorkflowRunner can be used as is for scheduling
- Per-job scheduling time is vanishingly small in any significant processing

**The Bad**
- Sequential Testing is always much slower

**The Ugly**
- MPMF is showing signs of quadratic growth
Concluding Remarks

- Events are plausible alternative to DAGs as a basis for workflow scheduling, but DAGs are still useful
- Available now as part of the MiG, or as stand-alone implementation in mig_meow
- Demonstrates acceptable scalability
- Well suited to extremely dynamic, distributed or heterogeneous systems (with work)
- More work needed to make managing event based analysis prettier and more useable