Building robust and reproducible workflows with Cylc and Rose

Dr Stuart Whitehouse
Data Workflows Team
Met Office
Contents

• Introduction to Cylc and Rose

• Robustness features

• Reproducibility

• Summary
Introduction to Cylc and Rose
What is Cylc?

- Cylc is a workflow engine for running suites of inter-dependent jobs
- Cylc can automatically:
  - Submit tasks across computer systems and resource managers.
  - Recover from failures.
  - Repeat workflows.
- Scales to very large, complex workflows
- Cycling suites:
  - Integer, real time, simulated time, externally driven
A Simple Workflow
Simple definition

```
[scheduling]
[[dependencies]]
  graph = """
    a => c => d
    a => b
  """
```
A Repeating Workflow
A Cycling Workflow
Cycling definition

```python
[scheduling]
[[dependencies]]
[[[P1]]]  # Repeat every cycle
graph = ""
   a => c => d
   a => b

# inter-cycle dependencies
a[-P1] => a
  c[-P1] => b

"""
```
Date/time cycling

Initial and final dates

Initial cycle point = 20191014
Final cycle point = 20191016

Times to run

$ a \Rightarrow b \& c$
$ a[-PT6H] \Rightarrow a$

$b$ and $c$ depend on $a$

$a$ depends on $a$ from the previous cycle

$x$ depends on $c$, but only for T06 and T18
Supports many batch systems

- at
- background
- loadleveler
- lsf
- moab
- pbs
- sge
- slurm
Scaling

• Proven to work at the Met Office with:
  • Thousands of recurrences
  • Tens of thousands of tasks
  • Hundreds of thousands of dependencies

• Met Office workflows are often huge!
Met Office Usage

- > 500 consistently since June
- a general upward trend (ignoring noise)
Major Sites using Cylc
What is Rose?

- Rose is a toolkit for writing, editing and running application configurations.
- Rose also contains other optional tools for:
  - Version control.
  - Suite discovery and management.
  - Validating and transforming Rose configurations.
  - Interfacing with Cylc.
Robustness
Robustness

• How do you ensure that your suite continues to run without manual intervention?
  • E.g. at the weekend, while you’re at a conference, on holiday, etc.

• Job fail for a number of reasons
  • With larger and more complex systems there’s more chance of a outage
    • E.g. HPC, archive, network problems, etc
    • There are also bugs in code, system errors, etc.

• These can cause errors either in job submission or job execution
Retries

- Cylc can automatically resubmit jobs in the event of failure
- Failures in job submission and job execution can be handled separately
- You can customise the intervals and number of retries

```
[runtime]
[[atmos]]
[[[job]]]
    submission retry delays = 3*PT15M
    execution retry delays = 2*PT10M, PT1H
```
Changing behaviour on retry

```
[runtime]

[[atmos]]

script = ""

test ${CYLC_TASK_TRY_NUMBER} -eq 3

# do something different

""

[[[job]]]

execution retry delays = PT0S, 2*PT10M
```

Use case:
Changing the timestep when the model fails to converge
Coping with Failure

• Modify the graph depending on whether a task passed or failed

```python
graph = ""
make_cake_mixture => bake_cake
bake_cake => sell_cake
bake_cake:fail => eat_cake
"""
```
Changing workflow based on task output

```
[scheduling]
  [[dependencies]]
  graph = """" arch:archive_success => housekeep
           arch:archive_fail => other_archive"

[runtime]
[[arch]]
  script = """
    test ${CYLC_TASK_TRY_NUMBER} -eq 3
    # do something different
  """

[[job]]
  execution retry delays = PT0S, PT10M
[[[outputs]]]
  archive_success = Archiving succeeded
  archive_fail = Archiving failed
```
For supported batch systems, polling can query the batch system to determine the job state.

This can be triggered by the user through the GUI...

...or automatically when a job times out.

Timeouts terminate a job (etc) when Cylc has not heard from the job by an expected time:

```plaintext
[[[job]]]
execution time limit = PT1H
```
Events and e-mails

- Define a set of events you care about
- If these occur, the system will e-mail you
- Useful if you need to monitor lots of suites, for example:

```plaintext
[[[events]]]
  mail events = submission failed, submission retry, failed, retry
```
As well as the features in Cylc for runtime robustness, Rose includes validation of the settings in an application.

For example, ensuring that you’re using compatible options

This is defined by metadata provided by the original developer and tested by the GUI

• Settings which conflict with the metadata are highlighted

More complex validation can be done using macros

• (using Rose’s Python API)
Reproducibility
What do we mean by “Reproducible”?

- We run the same tasks in the same order?
  - On a different platform (portability)

- We get bit-comparable results?

- We get scientifically identical results?

- We get the same runtime every time (e.g. no variation in wallclock time)

- We get the same bug every time (e.g. detecting race conditions)
Reproducing workflows

- Cylc’s suite configuration file is a text file, as we’ve seen
  - No special tools needed so it’s easy to edit

- Rose’s application configuration file is also a text file

- Version control these files to reproduce the workflow later
  - The whole workflow and configurations inside can be reproduced later

- This doesn’t protect from hardware/OS changes!
Reproducing results

• Getting bit-comparable results is very difficult
  • Eventually you’ll be forced to upgrade platforms/compilers which typically change results
  • Often a trade-off between performance/optimisation and getting the same results

• Scientifically identical is more achievable
  • But depends on a scientific judgement as to what is appropriate
  • How good is “good enough”? 
  • And this differs depending on the quantity and the application
Conclusion
Summary

• Cylc and Rose provide a way of creating and reproducing a workflow

• They include lots of functionality to assist in robustness, including:
  • Retries
  • Polling, manual and automatic
  • Failure triggers
  • Events handling
  • Metadata validation
Where to find out more

• Cylc and Rose are Open Source software
  • Cylc: https://github.com/cylc/cylc-flow
  • Rose: https://github.com/metomi/rose

• Support is available
Questions?

For more information please contact

🌐 www.metoffice.gov.uk
✉️ stuart.whitehouse@metoffice.gov.uk
📞 +44 (0) 330 135 2491