# **NIWA TRIP REPORT**

# **Cylc + Rose Training & Discussions**

# Bureau of Meteorology, Melbourne, Australia

# http://www.bom.gov.au/assets-140/img/bom/bom_logo_clr.gif13-17 October 2014

# ***Hilary Oliver, NIWA***

During the week of 13-17 October 2014 I visited the Bureau of Meteorology in Melbourne at the invitation of Michael Naughton of CAWCR (*Centre for Australian Weather and Climate Research*) and Jim Fraser of BNOC (*Bureau National Operations Centre*). The principal reasons for the trip were to provide training on the new Rose and cylc technical infrastructures for the UM and related systems, and to help inform BNOC’s upcoming decisions on choice of operational metascheduler (researchers now use cylc and prefer it, but operations have used SMS for a long time and might prefer to stick with it or to transition to EcFlow, which is reputedly a rewrite of SMS).

Several of the discussions during the week were focused on whether or not cylc can do certain things that BNOC staff consider to be important in SMS. I have attempted to summarize these issues below because they impact on how existing SMS-based systems could be migrated to cylc. *Otherwise this report necessarily omits much of the detail that arose during what was a very full week of discussions. In particular it does not cover functionality that the two metaschedulers have in common, major differences in the suite definition format and consequences thereof, or any of the features of cylc that are more or less entirely missing in SMS (and EcFlow) such as automatic cycle interleaving, adaptive scheduling, and the universal date-time cycling that can handle all climate workflows (etc.) under exactly the same framework as NWP suites - for that please refer to my presentation material, which has been uploaded to the NIWA and CAWCR wikis.*

On the first day, Monday, I met with a number of Bureau staff, worked on some local suite issues with Yi Xiao (e.g. correct polling for tasks submitted to the SGE batch scheduler) and presented a general seminar on use of Rose and cylc at NIWA. To sum up the presentation, cylc has provided NIWA with an efficient modern framework for construction and control of complex distributed workflows; and Rose has given us the means to manage the complexity of these systems very effectively. As a result we now have self-contained, self-deploying, version controlled suites that can easily be configured, shared, collaboratively developed, and cleanly transitioned from research into operations.

Tuesday was dedicated to the ACCESS workshop on technical infrastructure, with a focus on job scheduling for NWP and climate suites. In the morning I gave an overview of Rose and cylc, more technically oriented than Monday's seminar, including comparisons with pre Rose-era UM infrastructure (e.g. UMUI), and finished up with code for minimal *Hello World* cylc suites and Rose apps in an attempt to show the fundamentally simple and intuitive nature of these technologies - which might not be immediately apparent if you just dive in at the deep end. Other participants presented on topics such use of SMS in operations at BNOC; prototyping of inter-operable cylc/SMS suites (which worked as proof of concept but was not recommended as the way forward); and use of Rose and cylc in Met Office coupled model seasonal forecasting. At the end of the day I presented a selection of advanced cylc topics.

On Wednesday morning I attended the BNOC daily weather briefing, followed by the first official meeting to discuss cylc’s operational capabilities. Participants included Jim Fraser, Joan Fernon, Ivor Blockley, Wenming Lu, Yi Xiao, Asri Sulaiman, Robin Bowen, Arnold Mavromatis, and Milton Woods. This was followed in the afternoon by a wider discussion on suite design issues, which involved other CAWCR staff too and focused more on the Rose suite development model as well as how the shared *env files* used in local SMS suites might translate to cylc (see below).

One item of concern was the stability and reliability of cylc in comparison with SMS. My thoughts on this are that SMS might reasonably be expected to be more stable than cylc because it is much older and has not been actively developed for some time, whereas cylc is still undergoing quite rapid development. That said, cylc has proven to be very reliable at NIWA (and at that Met Office as far as I'm aware) and we actively guard against breaking existing functionality with a formal code review process, in public on Github, and an increasingly large automated test battery that actually runs suites (i.e. not just source code unit tests). Every new feature or bug fix that goes into cylc is now accompanied by new tests to ensure that the associated functionality does not get broken in the future.

Another item of interest was cylc's single-daemon-per-suite model vs the SMS uber-daemon that runs all suites. SMS users seemed to view this aspect of cylc's design as a negative because it results in cylc having separate control GUIs for each suite rather than a single one for all suites (although we do have the *cylc gsummary* GUI that presents summary states of all of a user’s suites on multiple hosts, and from which suite control GUIs can be launched with a mouse-click). Cylc’s model does have a number of distinct advantages though. For example, no central server administration is required (and note that cylc suite daemons take care of themselves - the running suite is the daemon); it takes system-level problems such as hardware failure to bring down all of your cylc suites at once – because they are independent; and large multi-suite systems can be upgraded (e.g. to new cylc versions) incrementally, one suite at a time. I noted that there is currently no read-only access to running suites owned by others, but this restriction will be lifted when we replace the current all-or-nothing authentication mechanism.

Somewhat relatedly, the question of scaling with suite size (number of tasks) was raised, perhaps because my description of NIWA’s cylc suites gave the impression that they were a lot smaller than BNOC's systems. During the discussion it became apparent that SMS users tend to think in terms of “number of jobs submitted per day” (17,000 at BNOC) whereas in the cylc world, which is not solely focused on NWP cycling timescales, to get the number of job submitted in some period you have to multiply “the number of tasks in a suite" by the number of cycles executed (e.g. a 250 task suite in which all tasks complete four cycles results in 1000 job submissions). By this measure NIWA's systems are not insignificant – a quick calculation suggests we routinely exceeded 10,000 jobs per day during recent testing of an 1800-task ensemble suite. UK Met Office operational systems are considerably larger again, of course. It is possible that cylc's more sophisticated scheduling algorithm has higher overhead than SMS, but this matters less when you don't run your entire system in one suite daemon, and in any case we will definitely be ensuring that cylc scales well to handle the increasing requirements of the Met Office in coming years. The recent cylc-6 release, for example, has proven to be 6-7 times more efficient, by CPU usage, than the 5.x versions currently still in operation at the Met Office.

Another point that generated a lot of discussion was use of shared *env scripts* to avoid repetition in SMS suites. This evidently derives from SCS suite construction techniques (for historical reasons some SCS-based UM suites are still launched by the wider SMS system at BNOC). In my view cylc has better ways of achieving the same thing. Within a suite, multiple inheritance can be used to give every task exactly the configuration it needs, from environment and scripting to job submission and hosting, with no repetition or redundancy at all. Between suites we would typically pull common script files (etc.) in from external version control repositories at install time. Cylc does also support literal inlining of include-files at any point in a suite definition, however, which is more directly analogous to the use of shared scripts in SMS suites.

SMS *edit runs* were also a hot topic: SMS suite operators can easily re-trigger a task after making one-off changes to its settings via the GUI. In a running cylc suite, task settings can be overridden with a suite *reload*, although that requires modifying the suite definition; or with the *cylc broadcast* command, although that has wider use and does not currently have a simpler interface specifically for one-off use on single tasks. See below for more on this topic.

On Thursday I visited the BNOC *Information Systems and Services Division* to talk about cylc from a software design and systems management perspective. Arnold Mavromatis had previously installed cylc on his own box. He commented that it seemed well designed, particularly cylc-6’s spawning of all external processes - job submission, event handlers, task poll, and task kill - via a multi-processing pool (contrasted with the single-threaded nature of SMS which can occasionally result in the whole system locking up while it waits on a hung file). We also talked about the importance of supporting multiple cylc versions at once so that multi-suite systems can be upgraded incrementally, suite by suite (and running tasks need access to the same cylc version as their parent suite). As it happens cylc's native version wrapping system closely parallels BNOC's own software installation versioning infrastructure, so no problems were evident there.

Later on Thursday I met briefly with several Bureau staff who are using another (non-SMS) scheduler for processing satellite data. For comparison I demonstrated a cylc-6 integer cycling suite that can trigger parallel workflows to process incoming datasets. It was suggested, however, that this kind of application would benefit from having file arrival events actively trigger the suite workflow rather than, as cylc does it, tasks that poll for the incoming files. To date we have deliberately avoided supporting filesystem event triggering in cylc because *inotify* (for example) is not portable. However, Joan Fernon had the excellent idea that we could do this generically in cylc by using dummy tasks as external event proxies that wait for incoming *message triggers* (these are normally sent by running tasks to report events such as completion of an output file prior the end of a model run). As a result of Joan’s suggestion I have posted a proposal to support event triggering in cylc by exactly this method, but without requiring the workaround of deliberately using a dummy task as the receiver: <https://github.com/cylc/cylc/issues/1188>.

Another issue raised was whether cylc’s automatic task error handling – implemented at the top level in task job scripts - is sufficient to catch all errors in the Korn Shell scripts executed under SMS at the Bureau. Signal trapping is not inherited by subshells, so these scripts are apparently modified on the fly to add it to them. Cylc job scripts don’t rely on signal trapping in subshells though. Rather, the trap at the top level just assumes that subshells abort on error in the standard way, i.e. with non-zero exit status. I contend that this is the best that can be done by a generic metascheduler that does not impose restrictions on what can be run as a task. If a script exits with success status on error, that is technically a bug that should be fixed in the script rather than complicating the suite or the scheduler with workarounds (it is often sufficient to simply put ***set –eu*** at the top of scripts). That said, on-the-fly modification could be implemented for specific scripts that can take it, if need be, using include-files for inlined scripting, or the Jinja2 template processor, or external filter scripts (because cylc tasks can run anything you like).

The ACCESS Technical Infrastructure Working Group meeting was held on Friday morning. Suite design was discussed some more, along with how to do collaborative branch-and-merge development properly under Rose and FCM, and whether or not the rosie+FCM flat repository layout and branch naming conventions really need be used. In my opinion these conventions should be adhered to in order to avoid breaking functionality that (as far as we know) might depend on them. In any case, as a restriction of freedom this is of little consequence: in a collaborative environment it is very helpful to have a well-thought-out standard structure that everyone uses, and searchable meta-data provides a more powerful way to organise your suites than user-defined directory trees etc.

On Friday afternoon I demonstrated two ways of doing an SMS-style *edit run* in a running cylc suite, in a final meeting with BNOC staff. The *cylc broadcast* command can override any task runtime settings for one or more tasks or families at one or more future cycle points. Single-task edit run is a subset of this functionality; we do not yet have a simplified user interface specifically for that purpose, but the functionality exists and the result can be achieved easily enough via the command line. I also demonstrated a more direct analogue of the SMS edit run by manually editing and resubmitting one of the job scripts generated by cylc to encapsulate a task. It would not take much to hook either of these approaches into the suite control GUI, but to ensure that we follow up on it I have amended the following cylc repository ticket: [https://github.com/cylc/cylc/issues/91#issuecomment-59848840](https://github.com/cylc/cylc/issues/91%23issuecomment-59848840).

Last but not least, I would like to express my thanks to colleagues at the Bureau for a great week of stimulating discussions, and for being such generous hosts!