Efficiency and use of multiple processors

Multi-process

Metview itself is not multi-threaded, but it is multi-process, meaning that certain tasks will automatically run in parallel. Metview consists of several *modules*, each of which is a separate executable. Whether running from the user interface or from the Macro language, Metview will schedule the execution of these modules so that they run in parallel unless one module needs to wait for the result from another. Each module can run several instances of itself in parallel, configured by the environment variable `MARS_MAXFORKS` (default=6).

For example, a *Cross Section Data* icon might have two separate *Mars Retrieval* icons as data input (e.g. Temperature on model levels, and LNSP). When the *Cross Section Data* icon is executed, Metview will run both Mars retrievals in parallel, and the *Cross Section Data* module will wait until both are complete before it runs (because it depends on their results).

This is particularly useful to know when writing a macro. For instance, Mars (or other potentially long-running) calls should appear towards the start of the macro so that they can run whilst the rest of the macro is being executed. Macro will only wait for a module at the point at which its result is actually used. For example:

```
data = retrieve(....)
d = 2016-03-30      # retrieve() is running asynchronously
s = 'Hello world'   # retrieve() is running asynchronously
# .. other lines of code, not depending on 'data'

derived = data - 273.15  # Macro will wait here for the retrieve() command to finish
```

Note that only calls to other modules are asynchronous in this way - the functions internal to the Macro language (see List of Operators and Functions) are part of the Macro module and are therefore not farmed out to the other modules. As a rule of thumb, Macro function calls which relate to an icon (e.g. *Cross Section Data*, *Observation Filter*) will be called asynchronously.

The currently-running modules can be seen using Metview's Process Monitor - see Exploring Metview.

Threading

Although Metview's own internals do not use threading, Metview relies on several libraries for some of its work. Magics is used for plotting, and employs multiple threads when plotting contours.

Temporary files

To avoid the use of temporary files in large Macro computations, see Vectors.