Selectively instrumenting the code with DR_HOOK

Philippe Marguinaud, 22 June 2010

The whole ARPEGE/AROME code (including the XRD, and SURFEX) has been instrumented with dr_hook so that it be possible to assess the model performances on several platforms without relying on proprietary instrumentation software.

We found that running AROME on the NEC with dr_hook enabled yielded unreasonable overhead (more than 80%; this is the assessment of dr_hook itself).

Therefore, re-compiled the code with gmkpack using the GMK_DR_HOOK_ALL option. This option allows to select which files will be instrumented at run-time, using a namelist.

This namelist can be created automatically from existing dr_hook profiles using the following scripts:

- mkdh2f.pl; scans the code to establish the mapping between dr_hook tags and source code files; it should be run from the pack directory being considered and generates a file named `dh2f.pl'
- mknamhook.pl; takes a dr_hook profile, a threshold in milliseconds, and the name of file in which the namelist containing the source files not to be instrumented will be written. Routines whose elapsed an CPU time are below this threshold will not be evaluated by dr_hook.

Note that it is not always possible to remove a file from dr_hook instrumentation because a single source file may contain several routines with different profiles; a conservative approach is used : a source file is kept if at least one of its routine has either an average elapsed or CPU time above the threshold passed as argument. The list of source file names which should have been removed but could not is printed by mknamhook.pl.

After filtering files with mknamhook.pl, the overhead was reduced to about 8%; it is worth mentioning that mknamhook.pl showed that some conflicts remained (some source files contain several subroutines with different profiles, that is with times above and below the threshold). An overhead of 8% can however be considered as acceptable.

Below is the approach we recommend for using dr_hook:

- compile your pack with the "export GMK_DR_HOOK_ALL=1" option
- run the software once with dr_hook enabled on all routines
- filter the routines whose CPU or elapsed time is too small
- run the software again; the dr_hook overhead should be dramatically reduced

One issue is to find the good threshold to pass as an argument to mknamhook.pl. Here is how we decided to use 3 ms on the NEC; for this purpose, we used the following program:

```
program main

use yomhook, only : lhook, dr_hook
```

```
implicit none
integer :: i
real*8 :: zhook_handle
if (lhook) call dr_hook('main', 0, zhook_handle)
do i = 1, 100000
  call sub
enddo
if (lhook) call dr_hook('main', 1, zhook_handle)
contains
subroutine sub
real*8 :: zhook_handle
if (lhook) call dr_hook('sub',0,zhook_handle)
if (lhook) call dr_hook('sub',1,zhook_handle)
end subroutine
end program
```

On the NEC, running this program program with dr_hook took about 32s, and without dr_hook, it took 3s; the overhead of dr_hook is therefore 0.3 ms per routine. In order to have on overhead smaller then 10%, it is therefore necessary to remove subroutines whose CPU or elapsed time is below 3 ms.