WW3 Tutorial 4.4: Profiling ww3_multi

Purpose

In this tutorial exercise we will use the build-in high-level profiling capabilities of the multi-grid version of WAVEWATCH III ®. The multi-grid version is described in Tolman (2008), and the build-in profiling deals with how to assign resources to individual grids in a mosaic or multi-grid model application. We will quickly set up one of the multi-grid test cases with six individual grids in n MPI environment, and then concentrate on using the profiling tools to assess the impact of various ways of loading the work on the set of processors available.

Input files

The test case used here is a version of a standard WW3 test case (formerly m`ww3_test_03`). Files needed to be able to run these tests are found in the directory `day_4/tutorial_profiling`.

```
switch              (switch file)
clean.sh            (auxiliary scripts)
run_it.sh
ww3_grid.XXXX       (input files, XXXX = 9 grids: lowN hghN point)
ww3_strt.inp
ww3_multi.*         (9 preset wave model setups with above grids)
gx_outf.inp
colorset.gs         (GrADS scripts)
map_NN.gs           (NN identifies grid combinations used).
profile_NN.gs
figs                (directory with example graphics)
```

Setting up

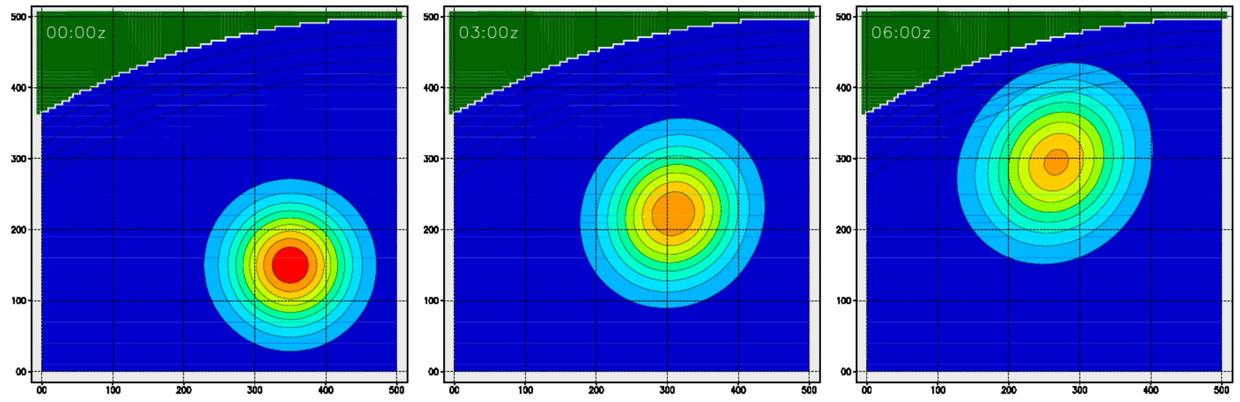
The case used here represents propagation unto a beach without any source terms. The corresponding `switch` file is in the data directory, and needs to be copied to the WW3 work directory, after which the model needs to be re-compiled (including MPI version of model codes).

```
cd ~/day_4/tutorial_profiling
cp switch ~/wwatch3/bin
make_MPI
```

Note the use of `make_MPI` instead of `w3_make`. We will use the third order propagation scheme with default GSE alleviation (PR2 switch), and for this test, the multi-grid profiling data generation switch (MPRF) is used, resulting in the use of the following switches

```
F90 NOGRB LRB4 SHRD NOPA PR3 FLX0 LN0 ST0 NL0 BT0 DB0 TR0 BS0 XX0
WNX1 WNT1 CRX1 CRT1 O0 O1 O2 O3 O4 O5 O6 O7 O11 O14 MPRF
```

The test case used here consists of one or more low-resolution grids at constant depth, combined with one or more high-resolution grids describing a curved beach area. An initial swell field is propagated toward the beach in a 6h model integration, with resulting wave heights as illustrated below.



The low resolution grid `low0` covers the entire area considered, whereas grid `low1-3` cover the same area with three overlapping grids. The test case uses either the single grid, or the three overlapping grids. Similarly, there is either a single high-resolution grid (`hgh0`), or a set of three overlapping grids (`hgh1-3`). As in the previous multi-grid time-stepping tests each of the grids used needs to be set up using `ww3_grid`, and initial conditions need to be produced using `ww3_strt`. After these preparations are made, the model is run using one of the prepared input files for `ww3_multi`. This run will produce profiling information as will be discussed below, as well as the usual raw output files for the individual grids. These raw files then can be processed to produce output as in previous exercises, among others, producing the output presented above. Because the focus of this exercise is on profiling, we will not spend much time on setting up the model, and instead directly use the prepared case script `run_it.sh`. To use this script, select one of the provide model setup for ww3_multi. We will start with a model consisting of the 6 overlapping grids `low1`, `low2`, `low3`, `hgh1`, `hgh2`, and `hgh3`. This is done by linking or copying the proper input file for `ww3_multi`

```
  ln -sf ww3_multi.33_wide.inp ww3_multi.inp
```

where ‘33’ indicates three ‘low’ and three ‘high’ grids. The model is run using

```
  run_it.sh
```

producing the normal raw model output files line `ww3_outf.low1`, log files `low.mww3`, `log.low1`, etc. Also produced are the various GrADS data files for each grid. The above graphics can be generated by running

```
  grads -pc "run map_33"
```

where ‘33’ again identifies the grid configuration. These graphics can always be generated, but we will generally not consider them because the focus here is on the profiling tools. Profiling is made possible due to the generation of data files `prf.`*`nnn`*`.mww3`, where *`nnn`* represents the process number starting with 001 and ending with the total number of processes assigned to the computation (padded with 0s). The information contained in these data files can be displayed with the GrADS script `profile.gs`, that is provided with the distribution of WW3. Generally, using this script requires two manual interventions. First, the wallclock time is normally taken from as being the entire model run time, but can be set by hand to get a clearer graph. In the script the lines

```
*
* Min and max times to plot  - - - - - - - - - - - - - - - - - - - -
*
  tdmin = tmin
  tdmax = tmax
*
* tdmin = 18.000
* tdmax = 48.000
*
```

set the time frame to be plotted as the full run time of the model, unless the highlighted lines are activated and the proper parameter values are provided. Furthermore, near the top of the script, colors are set. Colors 31 and higher are used for the model grids, in the order in which they are defined in `ww3_multi.inp.`

```
*
  'set rgb 31 255   0 255'
  'set rgb 32   0 105   0'
  'set rgb 33   0 190   0'
  'set rgb 34   0 215   0'
  'set rgb 35   0 255   0'
```

For a easier visual assessment of the profiling, the colors should be selected for grids with the same rank to have similar colors. To achieve this, various versions of the script `profile.`*NN*`.gs` are provided, where *NN* again represents the grid layout, as used in the provided model input files for `ww3_multi`. With all the tools in place, we can now look at profiling, and loading a mosaic of grids on a parallel computer. Note that the test problem is rather small, so that application on multiple processors is not necessarily helping run times. However, since we focus on using the profiling tool, this is not relevant for the exercise.

Three low-, and three high-resolution grids

We will start with running a model with three low-resolution and three high-resolution grids. The appropriate input file is linked in by executing:

```
  ln -sf ww3_multi.33_wide.inp ww3_multi.inp
```

In the input file, the six grids are defined as

```
$ WAVEWATCH III multi-scale input file
$ ----------------------------------
  6 0 F 1 F F
$
  'low1'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  1  1  0.00 1.00  F
  'low2'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  1  1  0.00 1.00  F
  'low3'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  1  1  0.00 1.00  F
  'hgh1'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  2  1  0.00 1.00  F
  'hgh2'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  2  1  0.00 1.00  F
  'hgh3'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  2  1  0.00 1.00  F
$
```

where all grids use the full communicator (i.e., all processors), as is requested with the highlighted lowest and highest fraction of the communicator to be used (0 and 1 for each grid). With this, the test case is run by executing

```
  run_it.sh
```

In the log file `log.mww3` generated by this model run shows a table with group information, and a table with resource assignments, reproduced below.

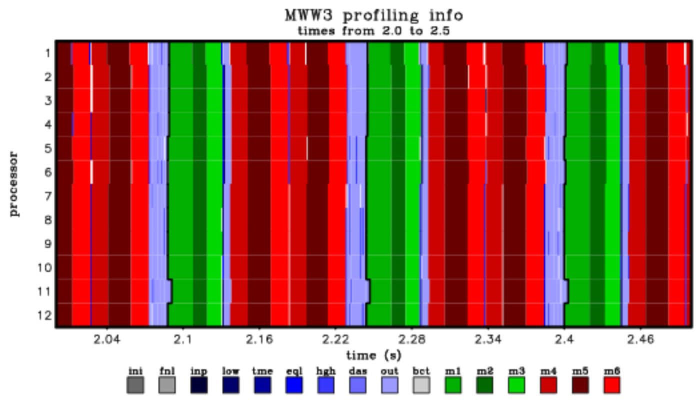
```
  Group information :
  nr   grids (part of comm.)
 ---------------------------------------------------------------------
   1    1 (0.00-1.00)  2 (0.00-1.00)  3 (0.00-1.00)
   2    4 (0.00-1.00)  5 (0.00-1.00)  6 (0.00-1.00)
 ---------------------------------------------------------------------

  Resource assignment (processes) :
  grid         comp.   grd  pnt  trk  rst  bpt  prt
 ----------------------------------------------------
  low1        001-012  011  ---  ---  ---  ---  ---
  low2        001-012  011  ---  ---  ---  ---  ---
  low3        001-012  011  ---  ---  ---  ---  ---
  hgh1        001-012  011  ---  ---  ---  ---  ---
  hgh2        001-012  011  ---  ---  ---  ---  ---
  hgh3        001-012  011  ---  ---  ---  ---  ---
 ----------------------------------------------------
```

The group information table reproduces the communicator fractions (highlighted) from the input file, and the resource table shows which processors are assigned to each individual grid (highlighted). Each grid uses all 12 processors used in this example, and each grid uses processor 11 to gather and write the field output. The run has also produced profiling data sets `prf.001.mww3`, `prf.002.mww3` through `prf.012.mww3`. These data form the basis for producing a profiling graphic by running the corresponding GrADS script

```
  grads -lc "run profile.33"
```

which results in the following graphic, showing model activities between 2 and 2.5s into the computations..



In this graphic, the six grids/models are identified as m1 through m6 in the legend, with m1-3 representing the three low-resolution grids in green, and m4-6 representing the three high-resolution grids in red. From the graphic, it follows that the three low-resolution grids are run sequentially on each processor (green), after which they are reconciled. Then the three high-resolution (red) run consecutively, are reconciled (blue), and are run again. After this all grids are reconciled, and output is generated. Effectively, all grids are run in the same ways as they are run on a single processor.

The '`low`' grids as well as the '`hgh`' grids are of equal rank within their respective group, and can be run simultaneously, rather than sequentially. The next activity in this exercise is to run the '`hgh`' grids simultaneously, while leaving the running of the '`low`' grids as before. This is achieved by using the prepared model input file

```
  ln -sf ww3_multi.33_wsbs.inp ww3_multi.inp
```

In the input file, the six grids are set up as follows

```
$ WAVEWATCH III multi-scale input file
$ ------------------------------------
  6 0 F 1 F F
$
  'low1'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  1  1  0.00 1.00  F
  'low2'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  1  1  0.00 1.00  F
  'low3'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  1  1  0.00 1.00  F
  'hgh1'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  2  1  0.00 0.25  F
  'hgh2'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  2  1  0.25 0.75  F
  'hgh3'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  2  1  0.75 1.00  F
$
```

where differences with the previous input file are highlighted. Grids `hgh1` and `hgh3` use the first and last quart of the communicator, while grid `hgh2` uses the middle half. Note that the model input has identical ending and starting communicator fractions between grids, and that the model is smart enough to assure that this does not result in individual processes shared by the grids. With this, the test case is run by executing

```
  run_it.sh
```

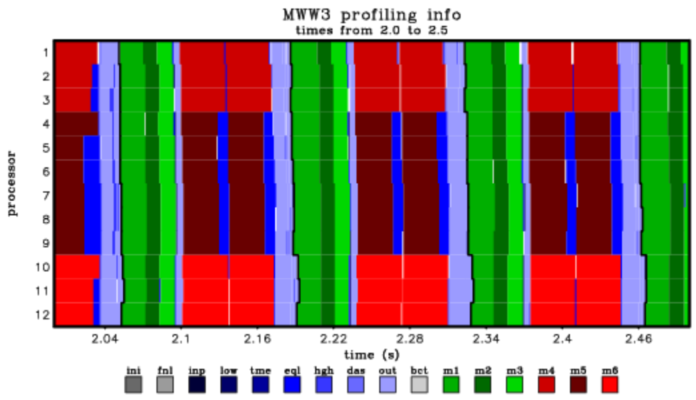
and the log file shows the following resource assignments table.

```
  Resource assignement (processes) :
  grid          comp.   grd  pnt  trk  rst  bpt  prt
 ----------------------------------------------------
  low1        001-012  011  ---  ---  ---  ---  ---
  low2        001-012  011  ---  ---  ---  ---  ---
  low3        001-012  011  ---  ---  ---  ---  ---
  hgh1        001-003  002  ---  ---  ---  ---  ---
  hgh2        004-009  008  ---  ---  ---  ---  ---
  hgh3        010-012  011  ---  ---  ---  ---  ---
  ----------------------------------------------------
```

Differences with the previous log file are again highlighted in **bold**. The three `low' grids still use all processors, but grid `hgh1` uses processors 1-3, grid `hgh2` uses processors 4-9, and grid `hgh3` uses procesors 10-12. The processors processing gridded field output for these three grids have moved accordingly. Running the GrADS script

```
  grads -lc "run profile.33"
```

results in the following profiling map



The low-resolution models (green) are running as before, but the high-resolution grids are running simultaneously on different processors. The grid m5 (`HGH2`) has the most resources and runs fastest. Clearly the three red models are not that well balanced yet.

The next activity in this exercise is to run the `hgh' grids simultaneously in a more balanced way, while also running of the ``low' grids side-by-side. This is achieved by using the prepared model input file

```
  ln -sf ww3_multi.33_sbs.inp ww3_multi.inp
```

In the input file, the six grids are set up as follows

```
$ WAVEWATCH III multi-scale input file
$ ------------------------------------
  6 0 F 1 F F
$
  'low1'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  1  1  0.00 0.25  F
  'low2'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  1  1  0.25 0.75  F
  'low3'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  1  1  0.75 1.00  F
  'hgh1'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  2  1  0.00 0.33  F
  'hgh2'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  2  1  0.33 0.66  F
  'hgh3'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  2  1  0.66 1.00  F
$
```

where differences with the previous input file are highlighted. Grids `hgh1-3` now all use equal parts of the communicator, and the `low' grids are set up as the high grids were set up in the previous case. With this, the test case is run by executing

```
  run_it.sh
```

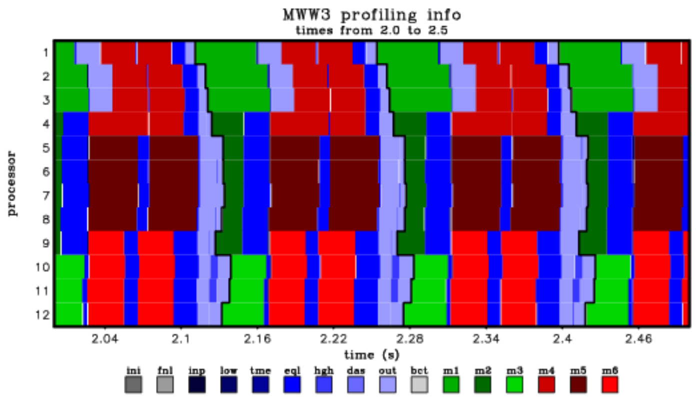
and the log file shows the following resource assignments table.

```
  Resource assignement (processes) :
  grid          comp.   grd  pnt  trk  rst  bpt  prt
 -----------------------------------------------------
  low1         001-003  002  ---  ---  ---  ---  ---
  low2         004-009  008  ---  ---  ---  ---  ---
  low3         010-012  011  ---  ---  ---  ---  ---
  hgh1         001-004  003  ---  ---  ---  ---  ---
  hgh2         005-008  007  ---  ---  ---  ---  ---
  hgh3         009-012  011  ---  ---  ---  ---  ---
 -----------------------------------------------------
```

Differences with the previous log file are again highlighted in **bold**. Running the GrADS script

```
  grads -lc "run profile.33"
```

results in the following profiling map



The red (high-resolution) grids now are more balanced, and the green (low) grids now run side-by-side, with too much resources given to the `low2` (m2) grid. Note that for the present test case, there is quite a bit run-to-run variability in the details of the profiling.

We will end now with the 6-grid setup. When interested, student can try to balance the low grids better, or move individual grids to other processors.

One low-, and three high-resolution grids

We will now consider a case with one low-resolution grid and three high-resolution grids. We will directly go to a balanced side-by-side approach for the high-resolution grids. This setup is prepared in the input file

```
  ln -sf ww3_multi.03_sbs.inp ww3_multi.inp
```

In the input file, four grids are used. These four grids include low0, hgh1, hgh2 and hgh3, which are set up as follows

```
$
$ WAVEWATCH III multi-scale input file
$ ------------------------------------
  4 0 F 1 F F
$
  'low0'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  1  1  0.00 1.00  F
  'hgh1'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  2  1  0.00 0.33  F
  'hgh2'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  2  1  0.33 0.66  F
  'hgh3'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  2  1  0.66 1.00  F
$
```

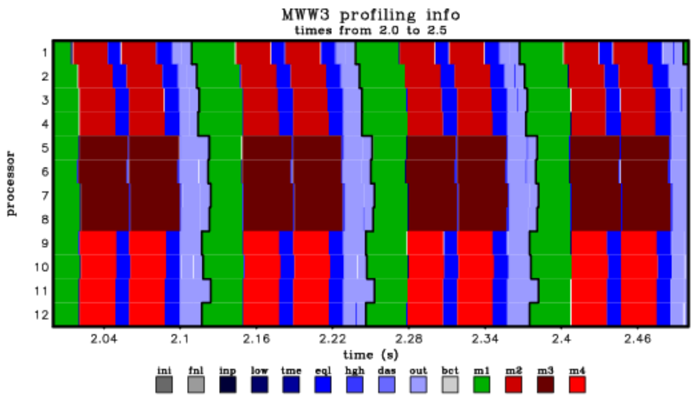
Note the reduction to 4 grids, and the balanced resource assignment for the high-resolution grids. Running this test case

```
  run_it.sh
  grads –lc "run profile.03"
```

results in the following resource assignments

```
  Resource assignement (processes) :
  grid          comp.    grd  pnt   trk   rst   bpt   prt
 ---------------------------------------------------
  low0         001-012  011  ---   ---   ---   ---   ---
  hgh1         001-004  003  ---   ---   ---   ---   ---
  hgh2         005-008  007  ---   ---   ---   ---   ---
  hgh3         009-012  011  ---   ---   ---   ---   ---
 ---------------------------------------------------
```

and profiling graph



which by now should be easy to interpret. It is left to the student to experiment more with this setup, or to experiment with the alternative setup of 4 grids with 1 high-resolution and 3 low-resolution grids (`ww3_multi.30_sbs.inp`).

One low-, and one high-resolution grid

Finally, we will consider a single high-, and a single low-resolution grid. Whereas there is little room to load-balance these grids, the setup can, nevertheless, be used to illustrate some features of the multi-grid model. This setup is prepared in the input file

```
  ln -sf ww3_multi.00_wide.inp ww3_multi.inp
```

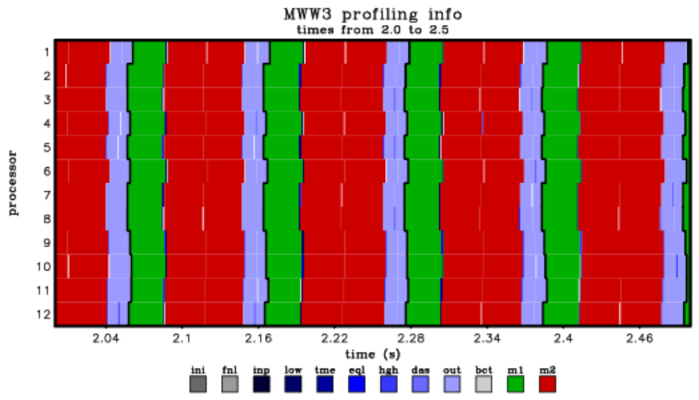
with the following input file

```
$ WAVEWATCH III multi-scale input file
$ ------------------------------------
  2 0 F 1 F F
$
  'low0'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  1  1  0.00 1.00  F
  'hgh0'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  2  1  0.00 1.00  F
$
```

Running this

```
  run_it.sh
  grads -lc "run profile.00"
```

results in the following profiling graph



which shows expected results. More interesting is running the model setup

```
  ln -sf ww3_multi.00_unbalanced.inp ww3_multi.inp
```

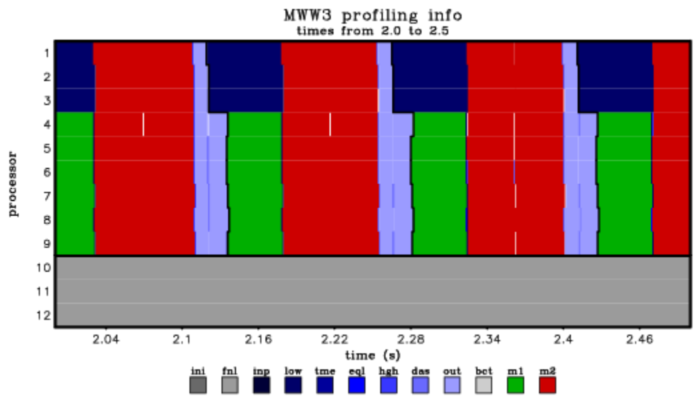
with the following model set up

```
$ WAVEWATCH III multi-scale input file
$ ------------------------------------
  2 0 F 1 F F
$
  'low0'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  1  1  0.25 0.75  F
  'hgh0'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  2  1  0.00 0.75  F
$
```

Both grids request only part of the communicator, and part of the communicator is not used at all.

```
  run_it.sh
  grads -lc "run profile.03"
```

shows this graphically



The low-resolution (green) grid runs on processors 4-9, only, whereas the high-resolution (red) grid runs on processors 1-9. Processors 10-12 have no work assigned at all, and are already in 'finalize' mode, while the other processors are computing. While processes 1-3 are not working on the high-resolution grid (red), they are waiting to get data transferred from the low-resolution grid (dark blue). This illustrates two things:

1) If you load work onto the communicator in an unbalanced way, then the model will do exactly as you request, and the model is robust enough to deal with this. The model does, however, provide a warning in the standard output of ww3_multi, in the case run above, it reported

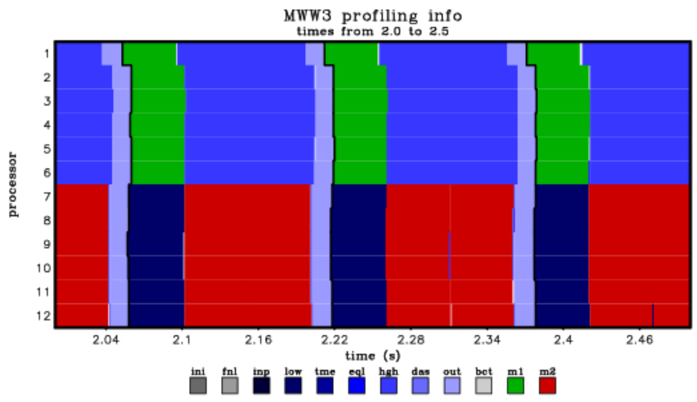
```
 Assigning resources :
---------------------------------------------------
      No (other) dedicated output processes.

      grid         comp.   grd  pnt  trk  rst  bpt  prt
     ---------------------------------------------------
      low0        004-009  008  ---  ---  ---  ---  ---
      hgh0        001-009  008  ---  ---  ---  ---  ---
     ---------------------------------------------------

*** WAVEWATCH III WARNING IN W3MLTI : ***
    POSSIBLE LOAD IMBALANCE GROUP  1 :     0     1

*** WAVEWATCH III WARNING IN W3MLTI : ***
    POSSIBLE LOAD IMBALANCE GROUP  2 :     0     1
```

2) This can also be a benefit, as it allows for more subtle memory usage management by only having selected use of certain processors for computing. An example of this is shown below, where the two grids each use an exclusive half of the resources, waiting on communications (blue) while the other processors work on the other grid. This maximizes the memory for individual grids.



Output processors

Finally, we will use the two-grid setup to demonstrate using separate processors for output. This can speed up computing by providing a-synchronous output, and help with memory management as described in the presentation of this morning. This setup is provided in

```
  ln -sf ww3_multi.00_unipts.inp ww3_multi.inp
```

with the following model set up

```
$ WAVEWATCH III multi-scale input file
$ ------------------------------------
  2 0 T 2 T T
$
  'point'
$
  'low0'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  1  1  0.00 1.00  F
  'hgh0'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  2  1  0.00 1.00  F
$
```

In this setup, a unified point output (i.e., one point output file with data for each output points from the grid in which it is best resolved in space), and an IO setup with separate processors for output (`IOTYPE = 2`) is used. Furthermore, an additional model definition file ‘`mod_def.points`” is used to define the spectral resolution for the unified point output, resulting in a file ‘`out_pnt.point’`. Running this

```
  run_it.sh
```

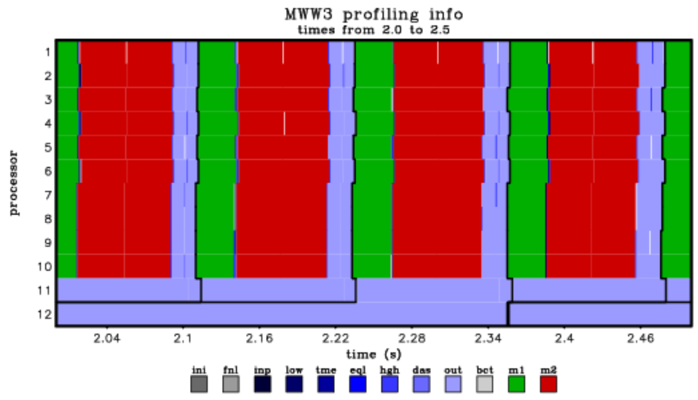
shows the processors reserved for IO only.

```
  Resource assignement (processes) :
  grid          comp.   grd  pnt  trk  rst  bpt  prt
 ---------------------------------------------------
  low0         001-010  011  012  ---  ---  ---  ---
  hgh0         001-010  011  012  ---  ---  ---  ---
 ---------------------------------------------------
    Unified point output at 012
```

Running the corresponding GrADS script

```
  grads -lc "run profile.03"
```

shows this graphically



Note that the model version used here still has some bugs, making it not possible to run with the following setup

```
$ WAVEWATCH III multi-scale input file
$ ------------------------------------
  2 0 T 2 T F
$
  'point'
$
  'low0'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  1  1  0.00 1.00  F
  'hgh0'  'no' 'no' 'no' 'no' 'no' 'no' 'no'  2  1  0.00 1.00  F
$
```

`IOTYPE = 2, 3` does require the last flag on the first input line to be 'T' to avoid error messages and hanging models. We expect this to be fixed in the svn server in the near future.

Pitfalls and suggestions

There are many ways that a set of grids can be loaded on a parallel computer. Hence, there are also many variants possible to the test provided here. The student is encouraged to play with options and loading schemes.

In larger applications, models are generally run on clusters of processors that share memory, generally referred to as nodes. MPI communications within nodes are generally performed in memory, whereas communications across nodes require a network. For older supercomputers this can clearly be observed in WW3 (e.g., Tolman, 2002). For modern Linux clusters, this difference in communications appears to have become bigger (Tolman, 2003). This implies that considering the compute resources as a set of nodes rather than a set of processors is generally beneficial. As output processors need more memory, it also may be beneficial to put output processors on nodes where not all other processors are used for computing. This may be done at the level of setting up the parallel compute environment, and possibly with the resource assignment as demonstrated and illustrated above.

References

Tolman, H. L., 2002: Distributed memory concepts in the wave model WAVEWATCH III. *Parallel Computing*, **28**, 35-52.
Tolman, H. L., 2003: Running WAVEWATCH III on a Linux cluster. NOAA/NWS/NCEP/MMAB Technical Note **228**, 27 pp.
Tolman, H. L., 2008: A mosaic aproach to wind wave modeling. *Ocean Modelling*, **25**, 35-47.
The manual.

More information:
Hendrik Tolman (Hendrik.Tolman@NOAA.gov)