profiling-plots

December 7, 2023

1 Introduction

MESH has been compiled in serial mode with netcdf capability against the following libraries on Digital Research Alliance of Canada's (DRA) Graham: 1. intel/2020.1.217 Fortran ifort compiler, 2. netcdf-fortran/4.5.2, 3. netcdf/4.7.4, and 4. hdf5/1.10.0.

Two versions of MESH has been compiled, namely r1773 (commit 75d48bd) and r1860ME (commit 52c7367). Both have been profiled using Intel's vtune/2020.1 profiler on DRA Graham.

2 Issues encountered

1. r1860ME (latest commit 52c7367) compiles successfully but raises the following runtime error running any MESH model setup if the ns value is assigned to BASINAVGWBFILEFLAG flag of the MESH_input_run_options.ini file,

```
WARNING: NON-CONVERGENCE AT POINT AT X,Y:   4873     1
WARNING: NON-CONVERGENCE AT POINT AT X,Y:   4873     1
WARNING: NON-CONVERGENCE AT POINT AT X,Y:   4873     1
WARNING: NON-CONVERGENCE AT POINT AT X,Y:   4873     1
WARNING: NON-CONVERGENCE AT POINT AT X,Y:   4873     1
WARNING: NON-CONVERGENCE AT POINT AT X,Y:   4215     1
WARNING: NON-CONVERGENCE AT POINT AT X,Y:   4873     1
WARNING: NON-CONVERGENCE AT POINT AT X,Y:   4873     1
WARNING: NON-CONVERGENCE AT POINT AT X,Y:   4873     1
WARNING: NON-CONVERGENCE AT POINT AT X,Y:   4873     1
WARNING: NON-CONVERGENCE AT POINT AT X,Y:   4873     1
WARNING: NON-CONVERGENCE AT POINT AT X,Y:   4873     1
WARNING: NON-CONVERGENCE AT POINT AT X,Y:   4873     1
WARNING: NON-CONVERGENCE AT POINT AT X,Y:   4873     1
forrtl: severe (174): SIGSEGV, segmentation fault occurred
Image              PC                Routine            Line        Source
sa_mesh_1860me_se  0000000000B51A3A  Unknown               Unknown  Unknown
libpthread-2.30.s  00002AEA3960A0F0  Unknown               Unknown  Unknown
sa_mesh_1860me_se  0000000000AA9BF6  save_basin_output        1376  save_basin_output.f90
sa_mesh_1860me_se  0000000000AA1DB8  save_basin_output         766  save_basin_output.f90
sa_mesh_1860me_se  0000000000B2191E  MAIN__                   1031  MESH_driver.f90
sa_mesh_1860me_se  000000000040CC12  Unknown               Unknown  Unknown
libc-2.30.so       00002AEA3963CE1B  __libc_start_main     Unknown  Unknown
sa_mesh_1860me_se  000000000040CB2A  Unknown               Unknown  Unknown
```

2. r1860ME (older commit b7f23d1) does not exhibit any issues with the `ns` flag in `MESH` model setups and runs successfully.

3. r1860ME (latest commit 52c7367) compiles successfully with `gfortran/9.3.0` compiler (with relevant dependencies), but echoes the following runtime error running any `MESH` model setup:

```
RUNCLASS36 is active.
ICEBAL_FREEZE_THRESHOLD (FREZTH) override is ACTIVE.
Uniform value:    -2.000000
ICEBAL_SWE_LIMIT (SWELIM) override is ACTIVE.
Uniform value:     100.0000
ICEBAL_SNOW_DENSITY_LIMIT (SNDENLIM) override is ACTIVE.
Uniform value:     900.0000
BASEFLOW component is ACTIVE.
 BASEFLOWFLAG  wf_lzs grid hf=60
         pwr_iak    2.203000
         flz_iak   0.2800000E-04

Program received signal SIGSEGV: Segmentation fault - invalid memory reference.

Backtrace for this error:
#0  0x2b8dd7c99730 in ???
#1  0x2b8dd7c988d5 in ???
#2  0x2b8dd813497f in ???
#3  0x605e24 in __output_variables_MOD_output_variables_group_update_ts
at ./Driver/MESH_Driver/output_variables.f90:1389
#4  0x602d21 in __output_variables_MOD_output_variables_update_ts
at ./Driver/MESH_Driver/output_variables.f90:2078
#5  0x5ff384 in __output_variables_MOD_output_variables_update
at ./Driver/MESH_Driver/output_variables.f90:2530
#6  0x91e471 in runmesh
at ./Driver/MESH_Driver/MESH_driver.f90:847
#7  0x922ab2 in main
at ./Driver/MESH_Driver/MESH_driver.f90:97
Segmentation fault
```

3 Profiling

[1]:

```python
# import libraries
import pandas as pd
import matplotlib.pyplot as plt

import os
import time
```

3.1 Total CPU time

Now, lets check the bottlenecks of `MESH r1860ME` version and its differences with `r1773`. Only 1 year of the `Fraser River Basin` setup was run in serial mode by each version.

[2]:

```
# read the profiling results
r1773 = pd.read_csv('./profiling-results/1773.csv')
r1860me = pd.read_csv('./profiling-results/1860me.csv')
```

Determining total CPU time used by each version (not considering the wall time):

```
r1773 used 04:58:08
r1860me used 05:37:05
```

3.2 Function call differences between r1860ME and r1773

Determining functions that are called/available in `r1860ME` but not in `r1773` plus see how much time they have used the CPU:

[4]:

```
diffs = r1860me[~r1860me.iloc[:,0].isin(r1773.iloc[:,0])].sort_values(by='CPU␣
 ↪Time', axis='rows', ascending=False)
```

The top 6 **new** function calls of `r1860ME` with considerable computation time are listed in the following:

[5]:

```
   Source Function / Function / Call Stack   CPU Time     Module  \
3                copy_field_scalar_to_scalar  1040.2800  [Unknown]
14                               map_field_2d   576.4670  [Unknown]
19                             func@0x151900   531.8390  [Unknown]
26                map_field_to_ranked_output   193.3040  [Unknown]
55                             func@0x40c720    28.0808  [Unknown]
56                      read_frame_from_file    26.1351  [Unknown]

                Function (Full)          Source File  Start Address
3   copy_field_scalar_to_scalar  field_utilities.f90              0
14                 map_field_2d  field_utilities.f90              0
19                func@0x151900            [Unknown]              0
26   map_field_to_ranked_output          mesh_io.F90              0
55                func@0x40c720            [Unknown]              0
56         read_frame_from_file          mesh_io.F90              0
```

From the profiling analysis, the top 10 function calls introduced newly in `r1860ME` can explain the time difference of ~40 minutes:

```
New function calls in r1860ME are responsible for 00:40:12
```

It seems that the following stacks/files added in `r1860ME` are mostly in charge of the differences:

```
Function calls:
field_utilities.f90,
mesh_io.F90,
sa_mesh_run_within_grid.f90,
variable_maps.f90
```

3.3 Analysis of common function calls in r1860ME and r1773

[11]:

```
r1860me = r1860me.set_index('Source Function / Function / Call Stack').loc[:,␣
↪'CPU Time'].groupby('Source Function / Function / Call Stack').sum().
↪sort_values(ascending=False)
```

[12]:

```
r1773 = r1773.set_index('Source Function / Function / Call Stack').loc[:, 'CPU␣
↪Time'].groupby(['Source Function / Function / Call Stack']).sum().
↪sort_values(ascending=False)
```

[13]:

```
all = pd.merge(r1860me,
               r1773,
               left_index=True,
               right_index=True,
               how='inner',
               suffixes=('_r1860me', '_r1773'),
              )
```

[14]:

```
total_common_delay = (all['CPU Time_r1860me'] - all['CPU Time_r1773']).sum()
```

For the common functions calls shared between the two versions, the `r1860ME` is perfoming quite close to the `r1773` version.

```
Overal common function calls in r1860ME are about 00:00:59 slower than r1773.
```

And, here is a list of top 10 common functions and their processing times in the two version:

[16]:

```
                                         CPU Time_r1860me  CPU Time_r1773
Source Function / Function / Call Stack
classw                                           1396.490        1244.420
grdran                                           1083.770         990.855
runclass36_within_tile                           1045.140        1183.150
wflow                                             955.822         956.044
tmcalc                                            886.655         890.895
__libm_powf_l9                                    791.024         671.511
aprep                                             773.889         817.789
flxsurfz                                          676.447         692.902
watrof                                            663.026         664.725
tsolvc                                            658.972         704.973
```

3.4 Overal module contributions in each version

[18]:

```
td_1773 = pd.read_csv('./profiling-results/top-down_1773.csv')
td_1860me = pd.read_csv('./profiling-results/top-down_1860.csv')
```

Here, you can see the percentage of each module within `MESH r1773` taking over the total computation:

```
[57]: Function Stack
          runmesh                         100.000000
           run_within_tile                 94.030769
           run_between_grid                 3.043447
           run_within_grid                  2.192695
           climate_module_update_data       0.648610
           output_variables_reset           0.068582
           run_save_basin_output            0.007378
           read_initial_inputs              0.003298
           run_within_tile_init             0.001453
           run_within_tile_finalize         0.001342
      Name: CPU Time: Total, dtype: float64
```

And, also for `r1860ME`:

```
[59]: Function Stack
          runmesh                       100.000000
           run_within_tile               82.421842
           read_input_forcing_frame      12.110546
           run_within_grid                2.706731
           run_between_grid               2.698741
           output_variables_reset         0.046276
           run_save_basin_output          0.006081
           read_initial_inputs            0.003263
           open_input_forcing_files       0.002670
           run_within_tile_finalize       0.001187
      Name: CPU Time: Total, dtype: float64
```

4 Final notes

It seems that the `I/O` processes are causing newer version to take more time to complete, and therefore, there is a difference in the computation time.

There are many other details hidden in the profiling results, they could be shared with you. Let me know if you need to have access to them.

Profiling results are done using `vtune/2020.1` on DRA Graham