Facing The Complex Realities: A Case Study With ICON

Markus Geimer
Jülich Supercomputing Centre

Experiment setup

- Measurements on JUWELS Booster @ JSC
  - Using a single compute node: 4 MPI ranks, each driving one NVIDIA A100 GPU
- ICON 2024.07 open-source release
  - Compiled with NVHPC 24.3, CUDA 12.2 & OpenMPI 4.1.5
  - Analyzed using Score-P 8.4 and CubeGUI 4.8.2
- NextGEMS R2B4 test case
  - 3-days simulation 01. – 03.01.1979
  - Atmosphere, with land (JSBACH lite) & sea ice
  - 20480 grid cells
  - Effective mesh size ~158 km
  - 90 vertical levels up to 75km
  - AES-physics
  - ECHAM Forcing

Step 0: Baseline measurement

- Configuring and building ICON w/o Score-P took ~31 min (using `make -j8`)
  - By default, `make` also runs configuration scripts of sub-components…

- Reported runtime: 374 s

Step 1: Enable Score-P instrumentation (I)

```
% git diff juwels_booster.gpu.ompi_nvhpc-24.3
 MODULES='NVHPC/24.3-CUDA-12 OpenMPI/4.1.5 netCDF-Fortran ecCodes libfyaml/.0.9 CMake'
+MODULES+=' Score-P'

   […]

-CC='mpicc'
+CC='scorep-mpicc'

   […]

-FC='mpif90'
+FC='scorep-mpif90'

   […]

-CUDACXX='nvcc'
+CUDACXX='scorep-nvcc'
```

Load Score-P module
&
Use Score-P compiler wrappers
instead of plain compilers

Step 1: Enable Score-P instrumentation (II)

```
    […]
+EXTRA_CONFIG_ARGS+=' --disable-delayed-config '
    […]
+SCOREP_WRAPPER=off \
 "${ICON_DIR}/configure" \
 BUILD_ENV="$BUILD_ENV" \
 CC="$CC" \
 … \
 ${EXTRA_CONFIG_ARGS} \
 "$@"
```

Enforces running all configuration scripts at configure-time ☺

Disables instrumentation at configure-time – it's pointless to instrument/measure configure checks…

Step 1: Enable Score-P instrumentation (III)

```
% mkdir build & cd build
% export NVCC_PREPEND_FLAGS='-ccbin nvc++'
      # If not set:
      # gcc: error: unrecognized command-line option '-Minstrument=functions';
      #             did you mean '-finstrument-functions'?

% export SCOREP_WRAPPER_INSTRUMENTER_FLAGS='--nomemory --openacc'

% /path/to/config/juwels_booster.gpu.ompi_nvhpc-24.3
% make -j8
```

By default, `nvcc` uses `g++` for host code. But mixing compilers is not supported by Score-P…
⇨ Enforce use of `nvc++`

Configure instrumentation, in particular enable OpenACC support

Step 1: Enable Score-P instrumentation (IV)

```
[…]

 PPFC     mo_yac_finterface.o
sh: -c: line 0: syntax error near unexpected token `('
sh: -c: line 0: 'mpif90 `scorep-config […]` -DHAVE_CONFIG_H -I.
-I/path/to/icon/externals/yac/src -I/path/to/netCDF-Fortran/include -g -O2
-Mpreprocess -Mrecursive -Mallocatable=03 -acc=verystrict\,gpu -gpu=cc80
-Minfo=accel\,inline -Mstack_arrays -Dis_contiguous(arg)=".TRUE." -c  -c
/path/to/icon/externals/yac/src/mo_yac_finterface.F90 -o mo_yac_finterface.o'
[Score-P] ERROR: Execution failed: […]
make[4]: *** [Makefile:800: mo_yac_finterface.o] Error 1
```

A Score-P bug, not handling parentheses correctly…

(Fix implemented – will be in Score-P 9.0)

Step 1: Enable Score-P instrumentation (V)

```
% git diff juwels_booster.gpu.ompi_nvhpc-24.3

   [...]

-ICON_YAC_FCFLAGS="-D'is_contiguous(arg)=.TRUE.'"
+ICON_YAC_FCFLAGS="-D'is_contiguous\(arg\)=.TRUE.'"
```

Temporary hack (Score-P only):
Add extra escaping

Step 1: Enable Score-P instrumentation (VI)

```
[…]
 FC (EX)  lib/icon.il
rm: cannot remove 'mo_math_utilities_1728478922_324804.o': No such file or directory
[…]
```

Verbose mode (`make V=1`):

`-Mextract=lib:lib/icon.il,…`

⇨ Creation of inlining library, not handled by Score-P

Step 1: Enable Score-P instrumentation (VII)

```
% git diff juwels_booster.gpu.ompi_nvhpc-24.3

    […]

+EXTRA_CONFIG_ARGS+=' --disable-pgi-inlib '

    […]
```

Disable use of inlining library

Step 2: Collect an initial profile (I)

```
% git diff exp.NextGEMS_R2B4.run
   [...]
+export SCOREP_OPENACC_ENABLE=yes
+export SCOREP_CUDA_ENABLE=yes
+export SCOREP_FILTERING_FILE=filter.txt
   [...]
% cat filter.txt
SCOREP_REGION_NAMES_BEGIN
    EXCLUDE cu[A-Z]*
    INCLUDE cuLaunchKernel
SCOREP_REGION_NAMES_END
```

Enable collecting OpenACC info…

…and the underlying CUDA stuff…

…but filter out some routines…

…in particular, CUDA API calls except for `cuLaunchKernel` (required to correlate kernels to launch call sites)

Step 2: Collect an initial profile (II)

```
   […]

0: [Score-P] src/measurement/profiling/scorep_profile_event_base.c:192:
   Error: Inconsistent profile. Stop profiling:
   Exit event for other than current region occurred at location 0:
      Expected exit for region 'acc_data_enter@mo_pheno_memory_class.f90:171[1510]'.
      Exited region 'mo_pheno_memory_class_init_pheno_memory_[1507]'
0: [Score-P] src/measurement/profiling/scorep_profile_debug c:250:
   Fatal: Cannot continue profiling. Activating core files
   (export SCOREP_PROFILING_ENABLE_CORE_FILES=1) might provide insight.
0: [Score-P] Please report this to support@score-p.org. Thank you
0: [Score-P] Try also to preserve any generated core dumps.

   […]
```

Turned out to be an NVHPC runtime bug…
(Reported as issue 4884228)

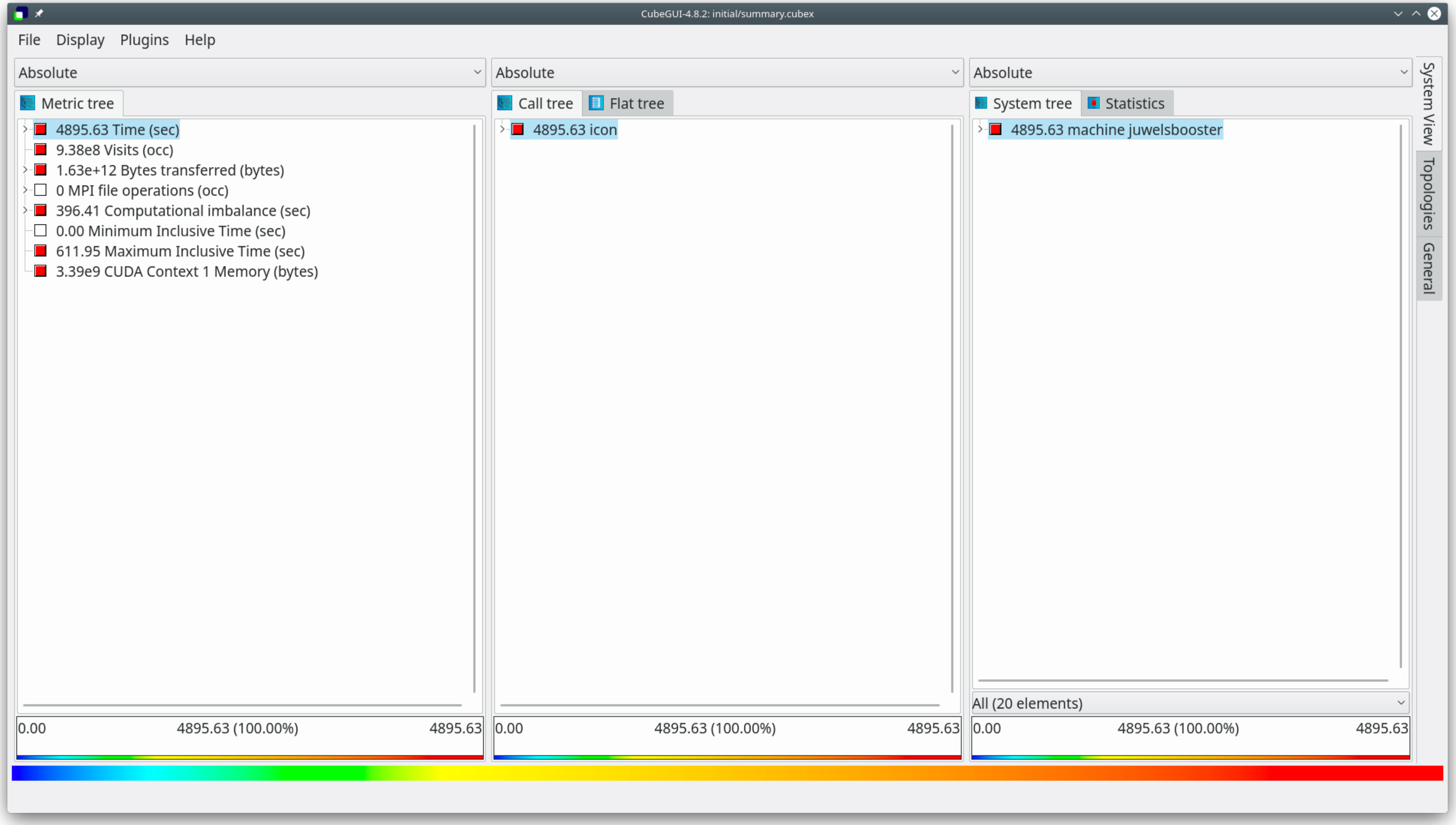
Step 2: Collect an initial profile (III)

```
% git diff mo_pheno_memory_class.f90

    […]

+#ifdef _OPENACC
+    USE openacc, ONLY: acc_attach
+#endif

    […]

-        !$ACC ENTER DATA ATTACH(mem%lai_mon, mem%fract_fpc_mon)
+#ifdef _OPENACC
+        CALL acc_attach(mem%lai_mon)
+        CALL acc_attach(mem%fract_fpc_mon)
+#endif
```

Temporary hack:
Replace directive by equivalent API calls
(similarly in 9 other source files…)

Attention:
Take care of `if` and `async` clauses!

Houston, we have a profile!



Step 0: Baseline measurement (revisited)

- We have modified the
  - build configuration to
    - use `nvc++` as CUDA host code compiler
    - disable use of inlining libraries
  - source code to
    - work around a compiler runtime issue
- Thus, we need to redo the baseline measurement!

- New reported runtime: 373 s

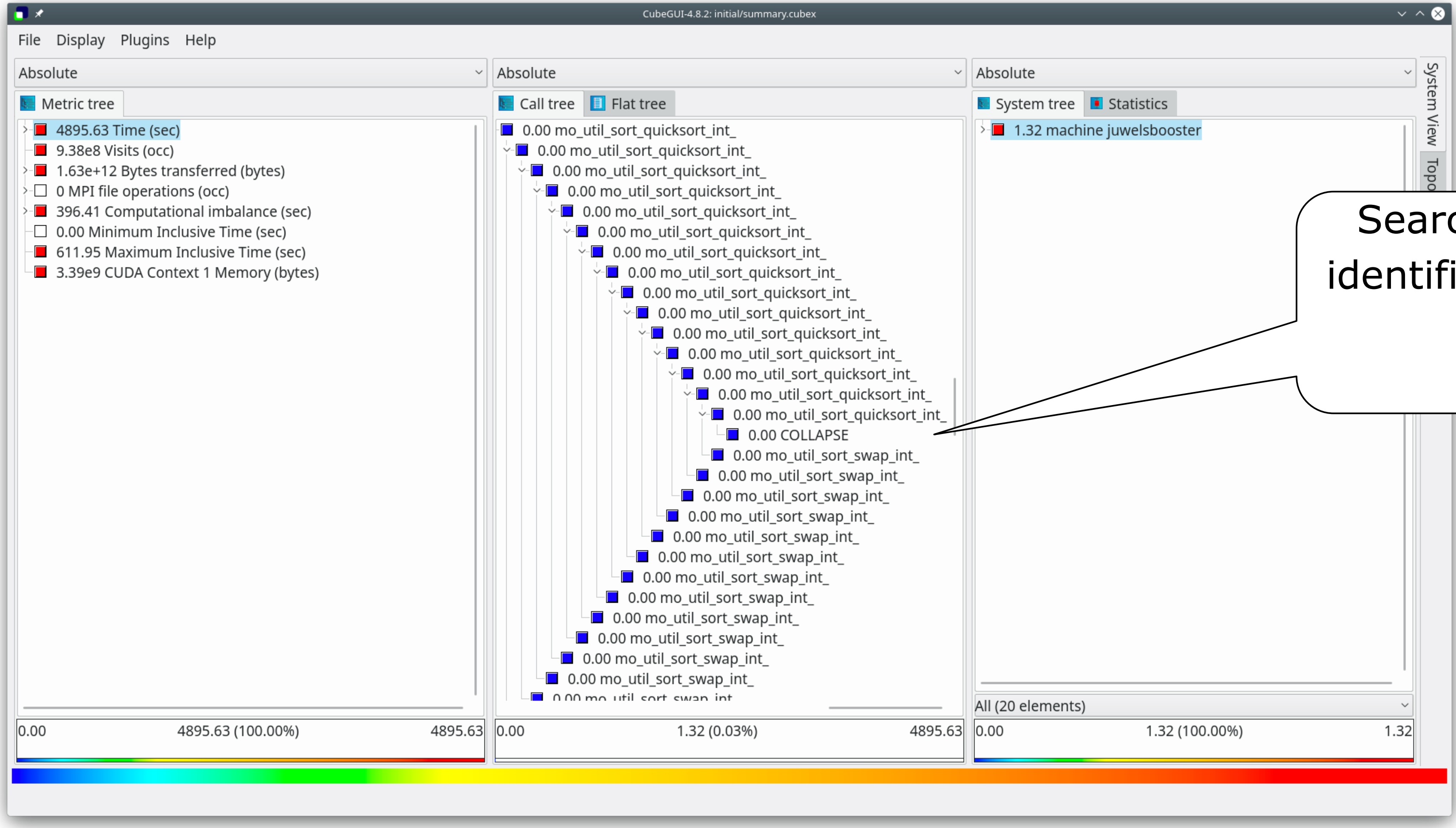
  ⇨ Modifications have no significant impact, result within measurement noise

Step 3: Examine initial profile & optimize configuration (I)

- Reported runtime: 596 s ⇨ ~60% overhead!
- Plus some warnings in the error log:

```
    […]
0: [Score-P] src/measurement/profiling/scorep_profile_collapse.c:82:
   Warning: Score-P callpath depth limitation of 100 exceeded.
0: Reached callpath depth was 580.
0: Consider setting SCOREP_PROFILING_MAX_CALLPATH_DEPTH to 580.
    […]
```

Step 3: Examine initial profile & optimize configuration (II)



Search for “COLLAPSE” callpath identified a small number of highly recursive functions

⇨ Filter!

Step 3: Examine initial profile & optimize configuration (III)

```
% scorep-score -r scorep_icon_initial_4_sum/profile.cubex | less
  […]
 type     max_buf[B]        visits  time[s] time[%]  time/      region
                                                     visit[us]
   USR  590,389,800  90,495,276    10.30     0.2       0.11  mo_parallel_config_idx_1d_
  CUDA  585,226,848  48,671,704   323.22     6.6       6.64  cuLaunchKernel
  CUDA  316,997,902  48,671,708  1810.11    37.0      37.19  COMPUTE IDLE
  CUDA  261,064,800   5,169,600    26.19     0.5       5.07  mo_jsb_tile_class_aggregate_weighted_by_fract_2d_890_gpu
   USR  184,350,296  28,361,584     6.97     0.1       0.25  mo_jsb_control_debug_on_
   MPI  121,860,847   5,476,592     7.62     0.2       1.39  MPI_Irecv
  CUDA  107,414,580  16,525,293   496.91    10.2      30.07  DEVICE SYNCHRONIZE
   USR  103,013,092  15,848,168     3.59     0.1       0.23  mo_mpi_get_comm_acc_queue_
   USR  102,772,800  15,811,200     2.81     0.1       0.18  mo_jsb_var_class_t_jsb_var_real2d_associate_pointers_
  CUDA   95,555,292   1,892,184     8.16     0.2       4.31  mo_communication_orig_exchange_data_mult_2363_gpu
  CUDA   95,555,292   1,892,184     6.73     0.1       3.56  mo_communication_orig_exchange_data_mult_2465_gpu
   USR   87,610,692  13,478,568     5.84     0.1       0.43  mo_jsb_control_timer_on_
   USR   85,802,132  13,200,328     3.42     0.1       0.26  mo_real_timer_util_read_real_time_
   USR   85,802,132  13,200,328     2.20     0.0       0.17  util_read_real_time
  […]
```

Score report shows more candidates for filtering…

Step 3: Examine initial profile & optimize configuration (IV)

- Filtering only reduced runtime overhead by 1%...

- Tweaking CUDA measurement options to bare minimum gave another 15%
  - … but no way to get the overhead below 40% if device kernel data is measured (significantly better than NVIDIA’s Nsight Systems, but still quite high)
  - Interpret the data with a large grain of salt!

- Measuring only host-side events has ~10% overhead ⇨ acceptable

Side note: Score-P development

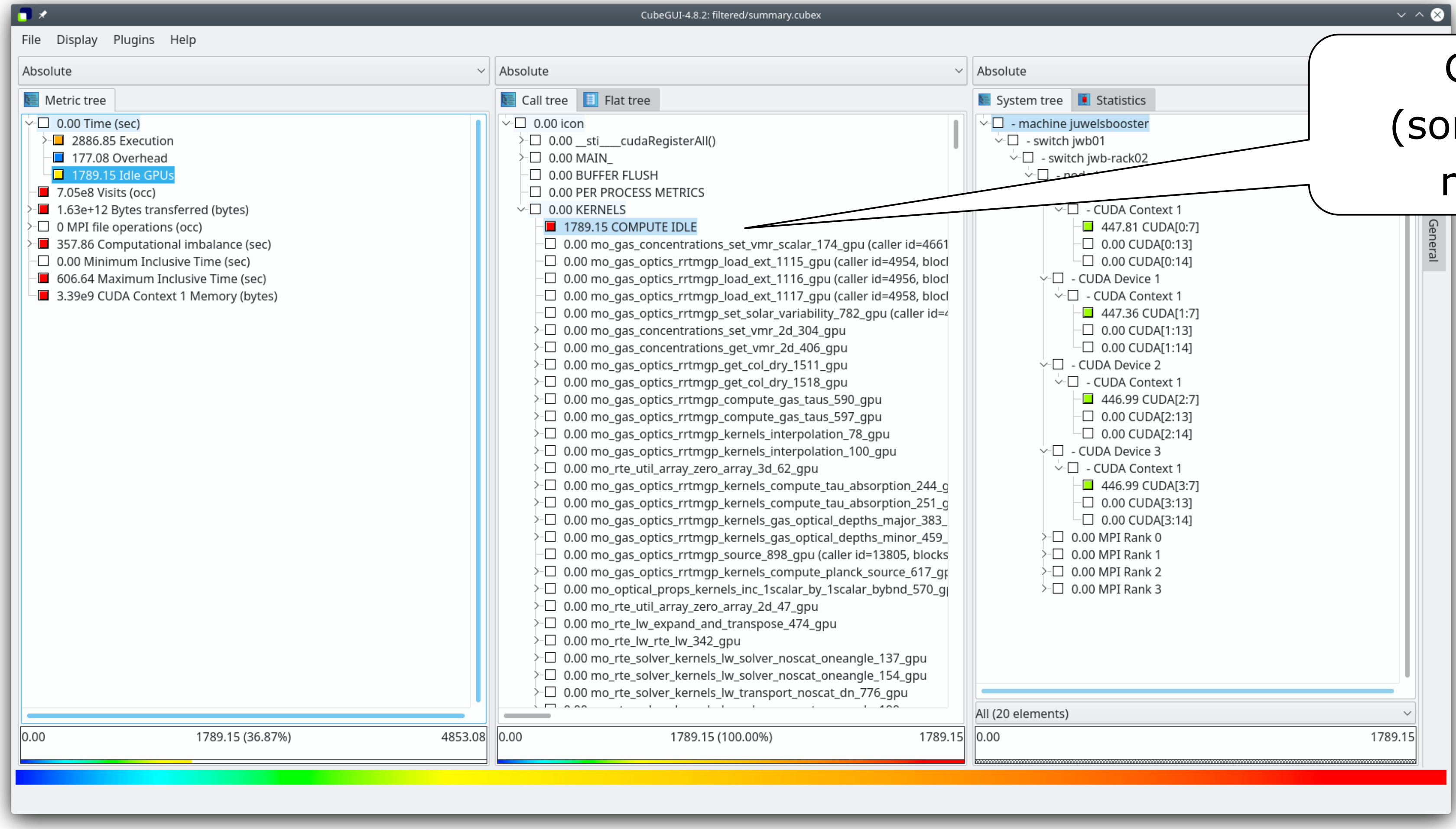
- We applied a profiler to profile Score-P while profiling ICON ☺

- Analyzing the identified hotspots, we found there is some room for improvement
  - Our current code works pretty well for many, many cases
  - But ICON is really a stress test…

- Some smaller improvements may already make it into Score-P 9.0, others require more thoughts and coding work
  - Please stay tuned!

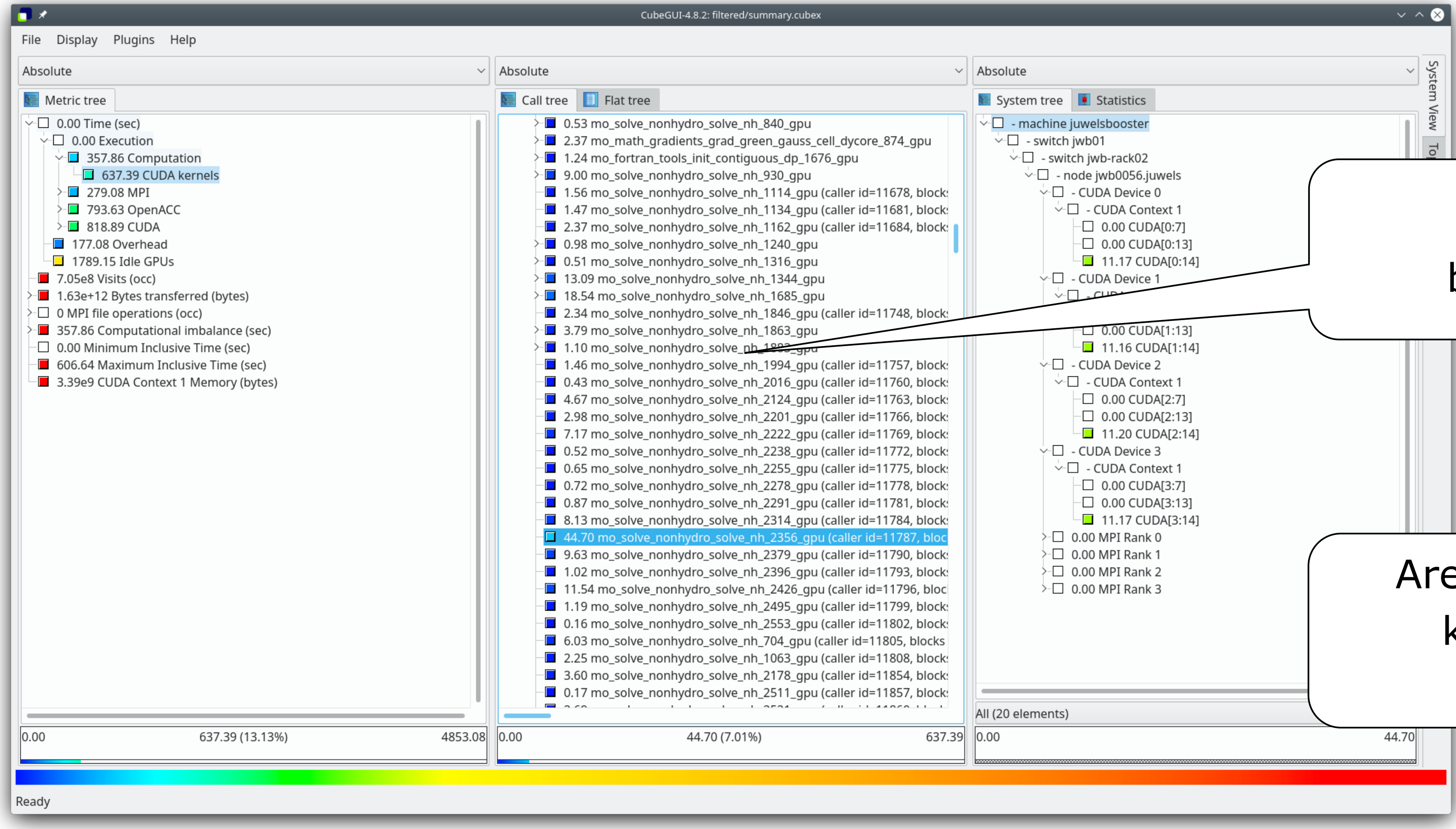
- But there are limitations in what we can do, imposed by the tools interfaces we use
  - Here: CUPTI, provided by CUDA

Step 4: Examine "optimized" profile w/ device data (I)



GPUs are idle quite a bit (some of this likely caused by measurement overhead)

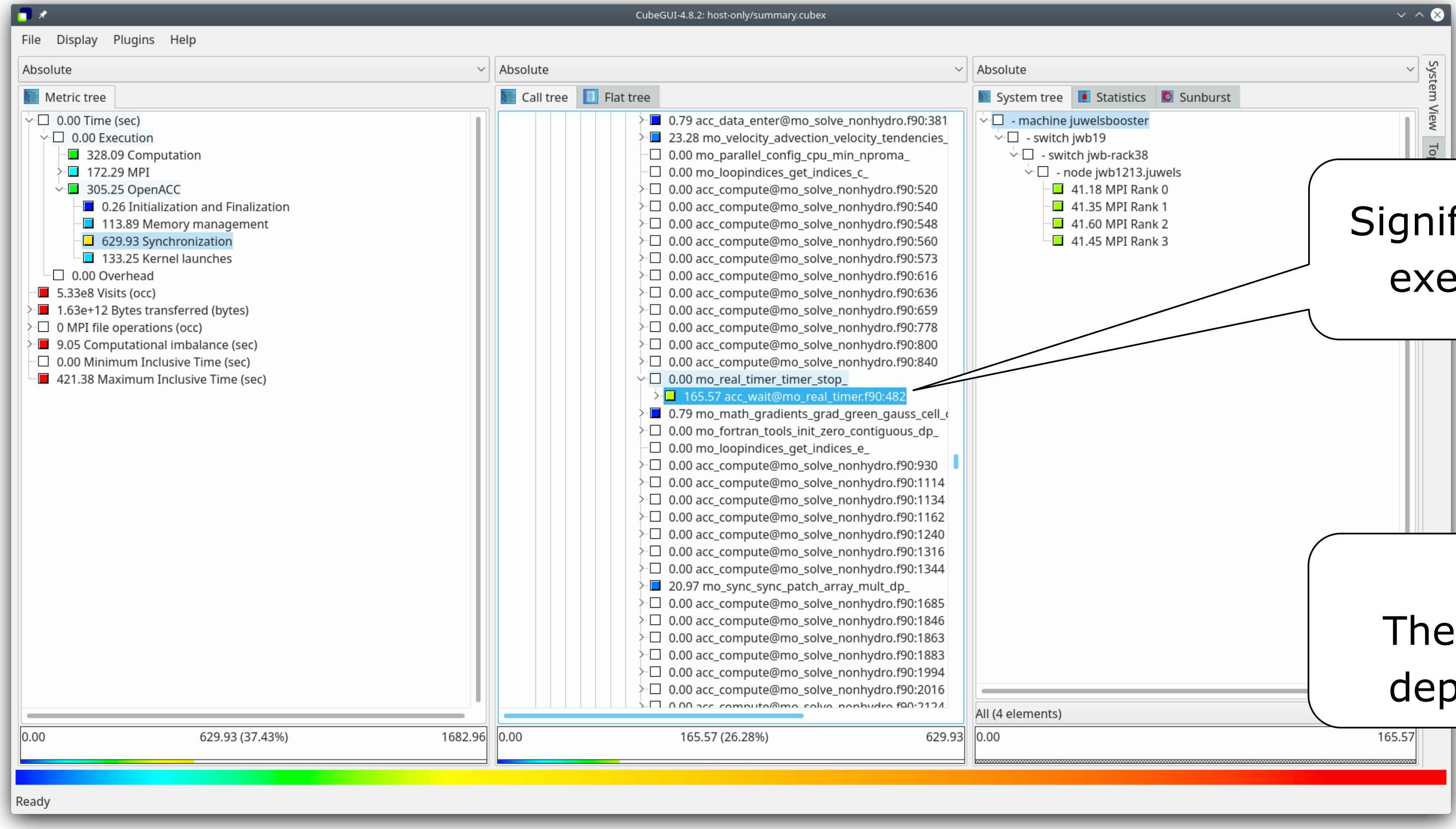
Step 4: Examine “optimized” profile w/ device data (II)



Many frequently called, but rather small kernels

Are there possibilities to fuse kernels, thus decreasing overheads?

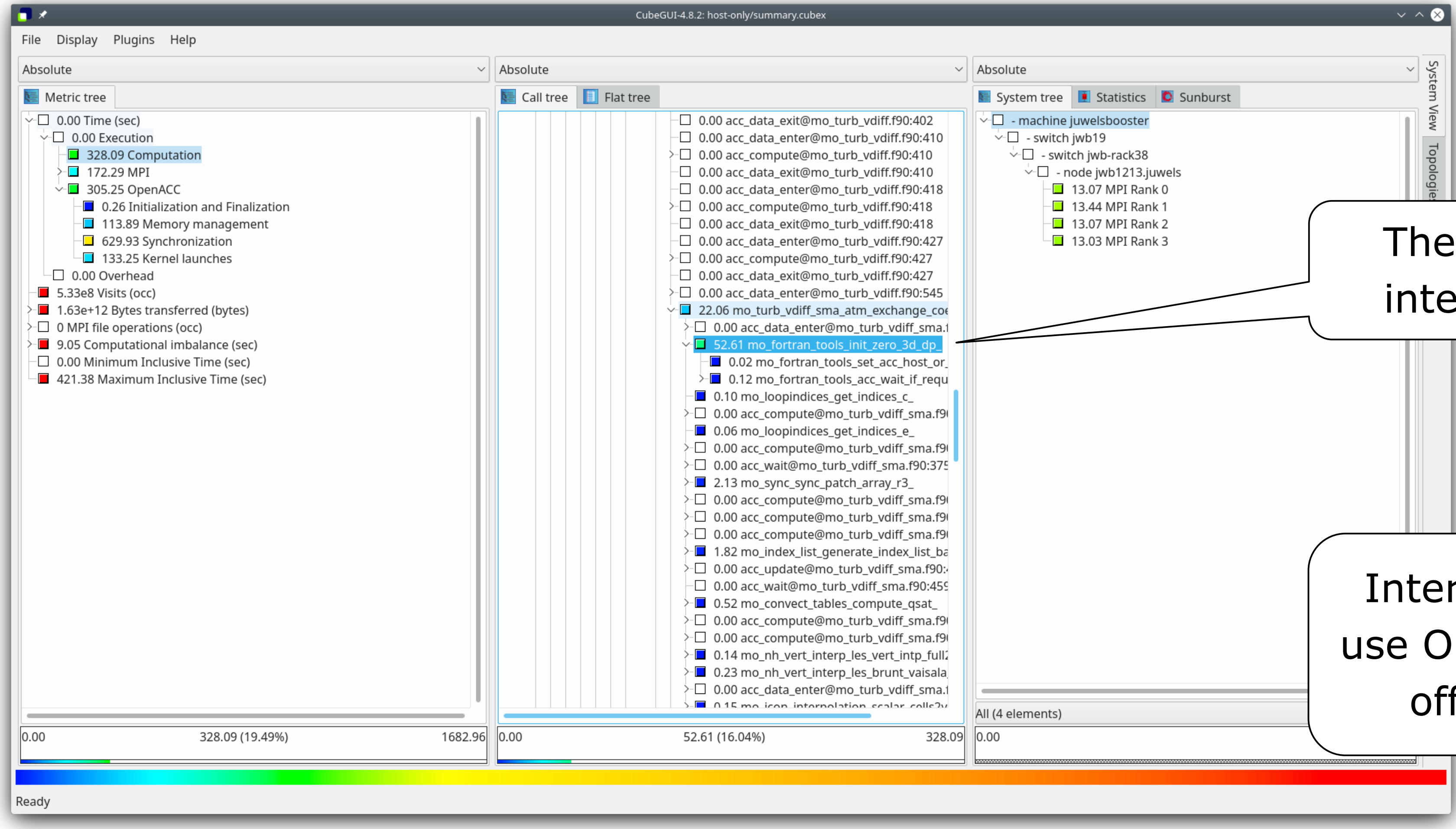
Step 4: Examine “optimized” host-only profile (I)



Significant waiting time for kernel executions in `timer_stop` call

Intentional?
The timer call doesn’t seem to depend on any kernel results.

Step 4: Examine "optimized" host-only profile (II)



There are still computationally intensive routines on the CPU.

Interestingly, the code seems to use OpenACC directives, but is not offloaded… Worth checking.

Take-away messages

- Analyzing large codes like ICON can be challenging
  - May uncover bugs & deficiencies in both tools and compiler runtimes

- The basic process is the same than with small(er) codes
  - Baseline ⇨ instrumentation ⇨ initial profile ⇨ adjust configuration ⇨ optimized profile ⇨ analysis

- Don’t hesitate to contact us if you’re stuck
  - The reasons for warnings and error messages are sometimes non-obvious
    - We’ve often seen them before and can suggest fixes or workarounds
  - We definitely want to know when something is suboptimal or even broken
    - If it’s our fault, we want to fix it
    - If it’s someone else’s fault, we’d like to bug them to fix it ☺