Profiling and tracing with Score-P & Cube
ICON case study

DKRZ Tech Talk
20th January 2026

Aleksandar Mitić & Dylan Kierans

Deutsches Klimarechenzentrum (DKRZ)

Structure of the talk

- Mute your microphone
- Turn off your camera
- In case of questions, please write them in a google docs (link in chat or scan the QR code:)

About us

Aleksandar Mitic

- RSE @DKRZ
- Anwendungen

Background:

M.Sc. in Computer Science

Current projects:

natESM

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

Dylan Kierans

- RSE @DKRZ
- Anwendungen

Background:

M.Sc. in HPC &
HPC Research Support

Current projects:

Funded by the European Union

Destination Earth

What will you learn today?

- Performance analysis
  - Workflow
  - Overview of the tools - Cube & Score-P
- Setting up the tools on Levante
- Case Study: ICON
  - Workflow
  - Detecting bottlenecks
  - Optimizations

Why performance analysis matters

HPC applications are:

- Increasingly complex
- Deeply hierarchical (node, NUMA, GPU, network)

Performance problems are:

- Non-obvious
- Often counterintuitive

Faster hardware

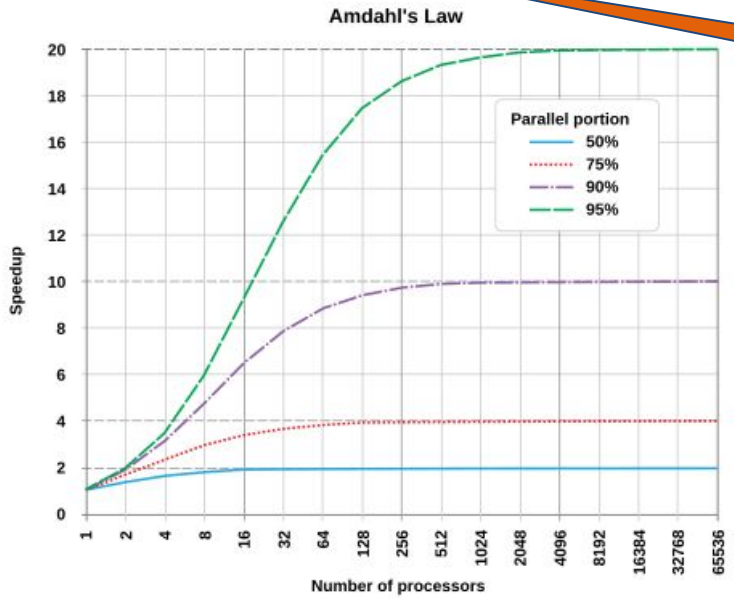
≠

Faster code

Why performance analysis matters

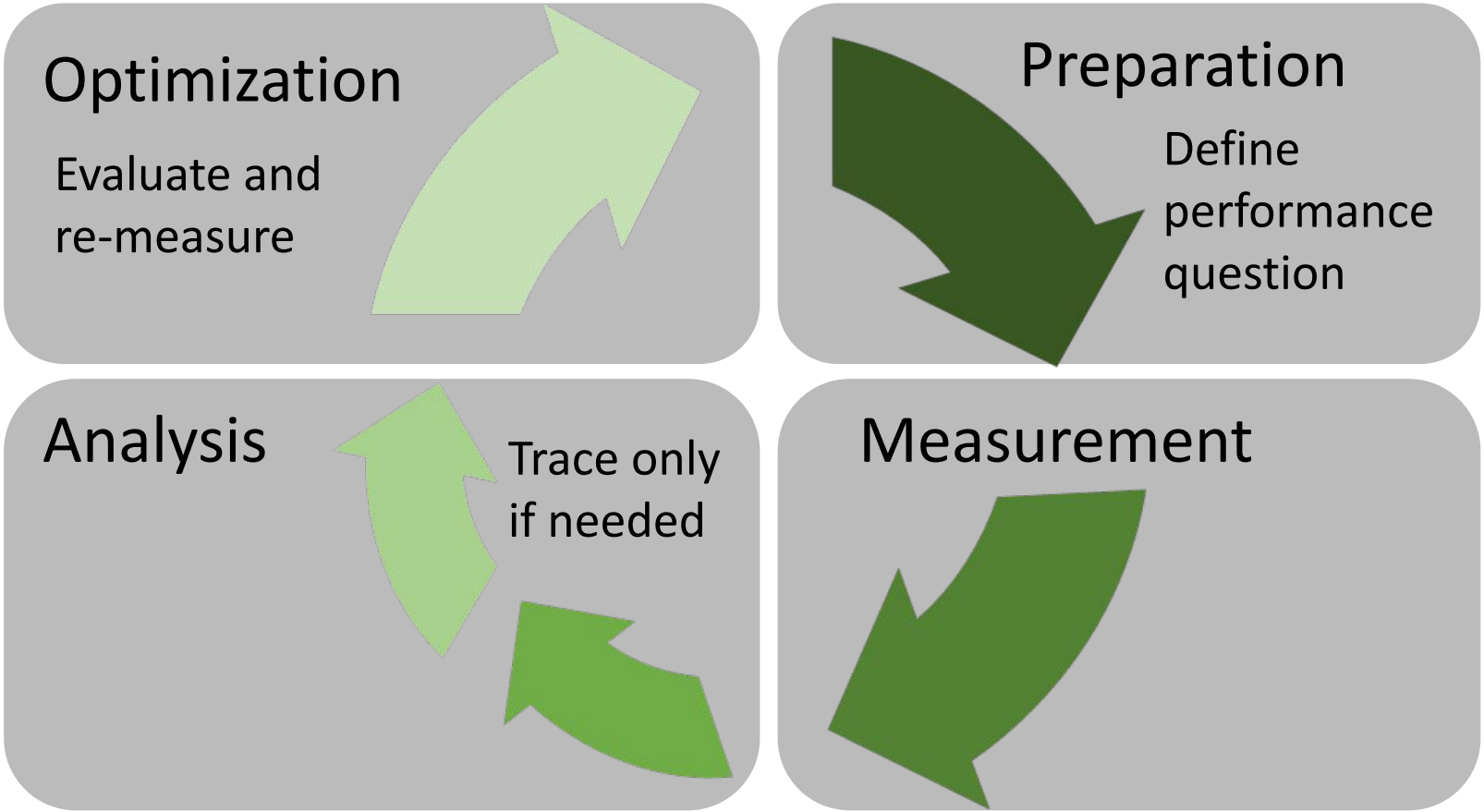
parallelism → faster simulation

caution!



Source [Online]:
en.wikipedia.org/wiki/Amdahl%27s_law

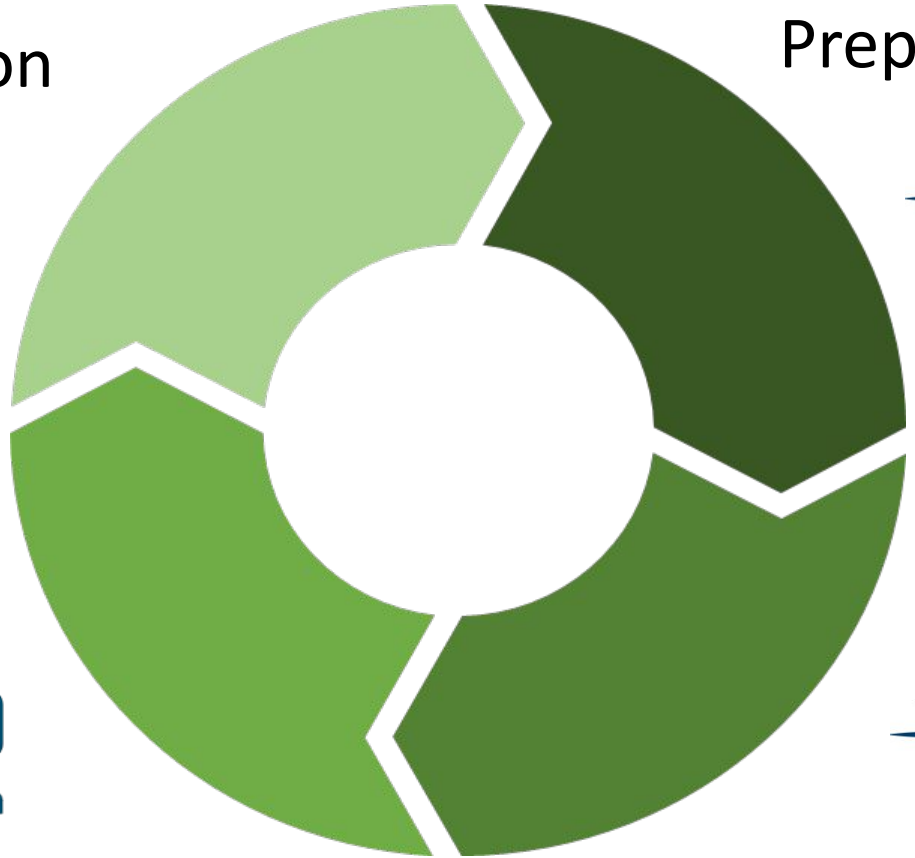
Performance engineering workflow



Performance engineering workflow

Optimization

Human being



Preparation

Score-P
Scalable performance measurement infrastructure for parallel codes

Analysis

cube
scalasca

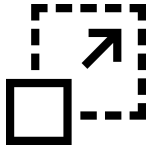
Measurement

Score-P
Scalable performance measurement infrastructure for parallel codes

Tools

Score-P features

- Open source 
- Portability: supports all major HPC platforms 

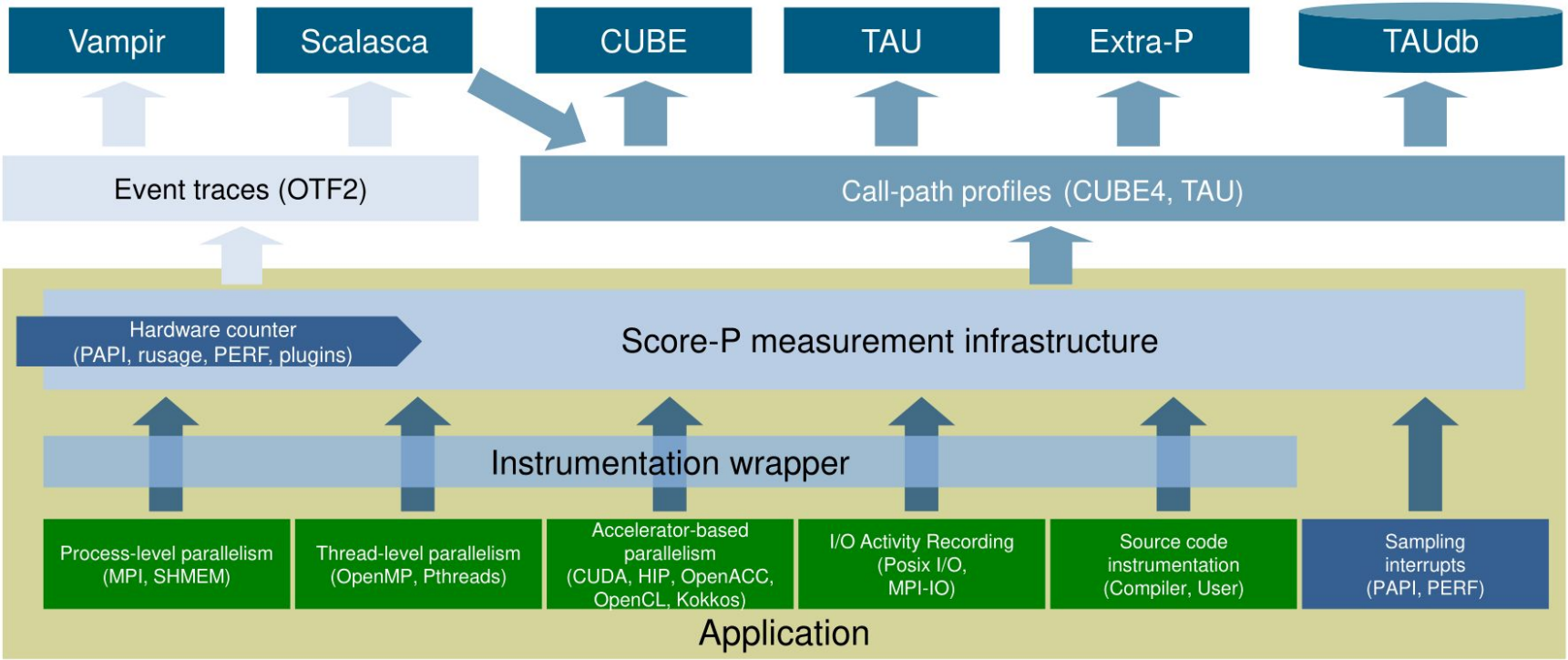
- Scalability 
- Functionality 

Score-P
Scalable performance measurement infrastructure for parallel codes

https://doi.org/10.5281/zenodo.17964650

image: Flaticon.com

Score-P ecosystem



Source [Online]: gitlab.jsc.fz-juelich.de/natesm/natesm/-/blob/main/Slides/PA-Course/00_Engineering.pdf

Cube

Cube framework provides libraries:

- writing
- reading measurement profiles

CubeGUI: https://doi.org/10.5281/zenodo.18079702

CubeLib: https://doi.org/10.5281/zenodo.18079696

Also, it includes a set of tools:

- to manipulate profiles
- to export
- to visualize data via graphical user interface for the manual performance analysis

Cube


File Display Plugins Help
Synchronize state of ...
Metric tree
2.5311e+10 Visits (occ)
23121.7450 Time (sec)
0.0000 Minimum Inclusive Time (sec)
180.6408 Maximum Inclusive Time (sec)
0 bytes_put (bytes)
1161465828 bytes_get (bytes)
1.4697e+10 io_bytes_read (bytes)
0 io_bytes_written (bytes)
1387583884.0000 Host Memory (C) (bytes)
0.0000 Host Memory (C++) (bytes)
0.0000 Host Memory (HBW) (bytes)
2.8234e+11 allocation_size (bytes)
2.7383e+11 deallocation_size (bytes)
7775482485 bytes_leaked (bytes)
1387583884.0000 maximum_heap_memory_alloc
9.2717e+10 bytes_sent (bytes)
9.2717e+10 bytes_received (bytes)
Call tree
Flat tree
23121.7450 icon
System tree
Statistics
Sunburst
23121.7450 machine Linux
System View
Topologies
General
0.0000
23121.7450 (100.0000%)
23121.7450

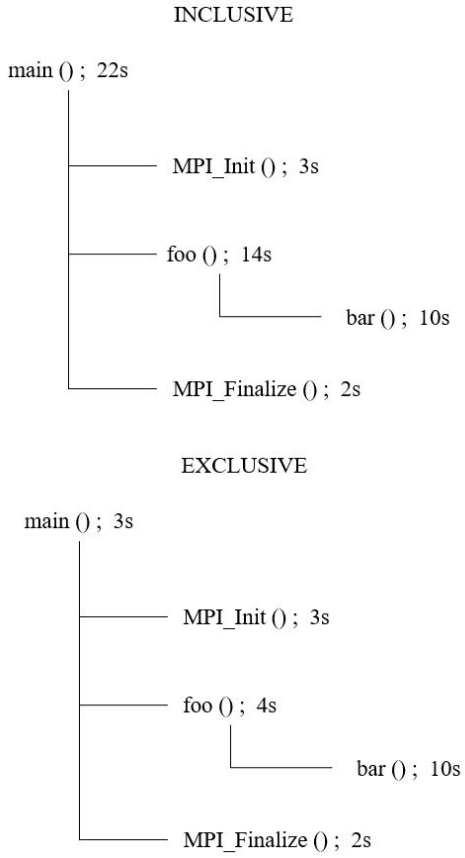
Metrics

Inclusive metrics

Accumulate values from the start of a call path to its end, including the time spent in all called functions.

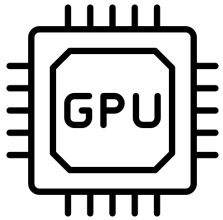
Exclusive metrics

Measure only the time or resources used directly in a function, excluding any calls to other functions.



In a nutshell: Performance questions

- Where is time spent?
- Why does scaling stop?
- Which ranks are idle?
- Is communication dominating?
- Are accelerators underutilized?



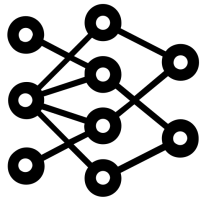


image: Flaticon.com

Setting up the tools on Levante

Score-P & Cube Levante configurations

Currently

- installed as a module:
  `scorep/7.0-intel-2021.5.0`
  `cube/4.6-gcc-11.2.0`
- installed via spack:
  - `scorep@7.0%intel`
  - `scorep@7.0%gcc`
  - `scorep@9.2%gcc` <- latest provided
  - `cube@4.6%gcc`
  - `cube-gui@4.9`

```
➜ ssh -Y levante
```

Build your own environment

- Where to find
  - vi-hps.org/projects/score-p/download/download.html

```
➜ wget perftools.pages.jsc.fz-juelich.de/cicd/scorep/tags/scorep-9.3/scorep-9.3.tar.gz
➜ tar xzvf scorep-9.3.tar.gz
```

- Reading the documentation helps sometimes ☺️
- Make it as a module available for everyone

or

- Everytime configure your environment manually, e.g:

```
➜ PATH=$PATH:/path/to/scorep/9.3-intel-2022.1.0/bin
➜ export LD_LIBRARY_PATH=/path/to/scorep/9.3-intel-2022.1.0/lib:$LD_LIBRARY_PATH
```

Build your own environment

- Intel

```
➜ cd scorep-9.2 && mkdir vpath_intel
➜ cd vpath_intel
➜ module load intel-oneapi-compilers/2022.0.1-gcc-11.2.0
➜ module load openmpi/4.1.2-intel-2021.5.0
➜ ../configure --prefix=/path/to/scorep/9.2-intel-2022.1.0 \
--with-mpi=openmpi \
--with-nocross-compiler-suite=intel \
--enable-shared \
--with-libgotcha=download
➜ make -j6
➜ make install
```

Build your own environment

- NVHPC-24.9 with OpenACC and CUDA

```
➜ spack load nvhpc@24.9
➜ spack load openmpi@4.1.5%nvhpc@24.9
➜ export CUDA_ROOT=/sw/spack-levante/nvhpc-24.9-p7iohv/Linux_x86_64/24.9/cuda/12.6
➜ export PATH=$CUDA_ROOT/bin:$PATH
➜ export LD_LIBRARY_PATH=$CUDA_ROOT/lib64:$LD_LIBRARY_PATH
➜ export GCC_ROOT=/sw/spack-levante/gcc-11.2.0-bcn7mb
➜ export CXXFLAGS="-std=c++11 -I${GCC_ROOT}/include/c++/11.2.0
-I${GCC_ROOT}/include/c++/11.2.0/x86_64-pc-linux-gnu"
➜ export LDFLAGS="-L${GCC_ROOT}/lib64 -Wl,-rpath,${GCC_ROOT}/lib64"
➜ cd scorep-9.3 && mkdir vpath_nvhpc-24.9_openacc_cuda
➜ cd vpath_nvhpc-24.9_openacc_cuda
➜ ../configure --prefix=/path/to/scorep/9.3-nvhpc-24.9 \
--with-mpi=openmpi --with-nocross-compiler=nvhpc \
--enable-cuda --enable-openacc --with-cuda=$CUDA_ROOT \
--with-libcuda-lib=$CUDA_ROOT/targets/x86_64-linux/lib/stubs \
--with-libcudart=/sw/spack-levante/nvhpc-24.9-p7iohv/Linux_x86_64/24.9/cuda/12.6 \
--enable-shared --with-libgotcha=download CC=nvc CXX=nvc++ FC=nvfortran
➜ make -j6
➜ make install
```

Build your own environment

- NVHPC-24.9 with OpenACC and CUDA + **PAPI**

```
➜ module load gcc/11.2.0-gcc-11.2.0
➜ spack load nvhpc@24.9
➜ spack load openmpi@4.1.5%nvhpc@24.9
➜ export CUDA_ROOT=/sw/spack-levante/nvhpc-24.9-p7iohv/Linux_x86_64/24.9/cuda/12.6
➜ export PATH=$CUDA_ROOT/bin:$PATH
➜ export LD_LIBRARY_PATH=$CUDA_ROOT/lib64:$LD_LIBRARY_PATH
➜ cd scorep-9.3 && mkdir vpath_nvhpc-24.9_openacc_cuda_papi
➜ cd vpath_nvhpc-24.9_openacc_cuda_papi
➜ ../configure --prefix=/path/to/scorep/9.3-nvhpc-24.9_papi \
--with-mpi=openmpi --with-nocross-compiler=nvhpc \
--enable-cuda --enable-openacc --with-cuda=$CUDA_ROOT \
--with-libcuda-lib=$CUDA_ROOT/targets/x86_64-linux/lib/stubs \
--with-libcudart=/sw/spack-levante/nvhpc-24.9-p7iohv/Linux_x86_64/24.9/cuda/12.6 \
--enable-shared --with-libgotcha=download CC=nvc CXX=nvc++ FC=nvfortran \
--with-papi-header=/path/to/papi/7.2.0-gcc-11.2.0/include \
--with-papi-lib=/path/to/papi/7.2.0-gcc-11.2.0/lib
➜ make -j6
➜ make install
```

Provide your own PAPI installation

```
export GCC_ROOT=/sw/spack-levante/gcc-11.2.0-bcn7mb
export CXXFLAGS="-std=c++11 -I${GCC_ROOT}/include/c++/11.2.0
-I${GCC_ROOT}/include/c++/11.2.0/x86_64-pc-linux-gnu"
export LDFLAGS="-L${GCC_ROOT}/lib64 -Wl,-rpath,${GCC_ROOT}/lib64"
```

ICON case study

I - Profiling & Tracing load imbalances

II- Hardware Performance Counters

ICON case study - I

- Simulation run on Levante @ DKRZ
  - Single compute node -> 128 MPI ranks
- ICON-mpim | `master` | Release icon-2025.10
  - `levante.intel-2021.5.0` wrapper
  - Score-P 9.2 & Cube 4.8.2
- ocean_omip_R2B7_01.run
  - 1-day 01. - 02 January 2000
  - `modelTimeStep="PT1200S"`
  - 64 vertical levels
  - CG solver

Recreating the workflow (i)

- Prepare the environment

```
➜ module load intel-oneapi-compilers/2022.0.1-gcc-11.2.0
➜ module load openmpi/4.1.2-intel-2021.5.0
➜ module load module load scorep/9.2-intel-2022.1.0
➜ export SCOREP_WRAPPER_INSTRUMENTER_FLAGS=' --user —-verbose --nomemory'
➜ cd $ICON_DIR
```

Recreating the workflow (ii)

- Adapt ICON's configure file to enable Score-P instrumentation

```
diff --git a/config/dkrz/levante.intel-2021.5.0
b/config/dkrz/levante.intel-2021.5.0
index 8cbde4e54e..70490ce1d4 100755
--- a/config/dkrz/levante.intel-2021.5.0
+++ b/config/dkrz/levante.intel-2021.5.0
@@ -101,9 +101,12 @@  EXTRA_CONFIG_ARGS="\
 --enable-waves \
 --enable-yaxt \
 "
-
+EXTRA_CONFIG_ARGS+=' --disable-delayed-config'
+CC='scorep-mpicc'
+FC='scorep-mpif90'
 ################################################################################
-
+SCOREP_WRAPPER=off \
 "${icon_dir}/configure" \
 AR="${AR}" \
 BUILD_ENV="${BUILD_ENV}" \
```

Baseline profile (iii)

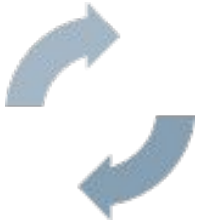
- Build ICON

```
➜ make build_intel && cd build_intel
➜ ../config/dkrz/levante.intel-2021.5.0
➜ make -j6
```

- Run ICON and check for Score-p measurement folder

```
➜ sbatch run/exp.ocean_omip_R2B7_01.run
### after successful run
➜ ls experiments/ocean_omip_R2B7_01
.   finish.status         initial_state.nc             NAMELIST_ocean_omip_R2B7_01.log
ocean-flux.nc   R2B7_ocean-grid.nc
..  icon_master.namelist  NAMELIST_ocean_omip_R2B7_01
NAMELIST_ocean_omip_R2B7_01_output  ocean-relax.nc
scorep-20251128_2353_1972646764543273
```

Baseline profile (iv)

- Write down the first results from the baseline profile
- And  to confirm that the results are reproducible

Getting the desired profile

```
➜ scorep-score -r profile.cubex

Estimated aggregate size of event trace:                   860GB
Estimated requirements for largest trace buffer (max_buf): 28GB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       28GB
(warning: The memory requirements cannot be satisfied by Score-P to avoid
 intermediate flushes when tracing. Set SCOREP_TOTAL_MEMORY=4G to get the
 maximum supported memory or reduce requirements using USR regions filters.)

flt     type     max_buf[B]         visits  time[s] time[%] time/visit[us]  region
        ALL 29,929,092,707 35,455,939,233 37076.48   100.0           1.05  ALL
        USR 29,878,720,118 35,243,056,766 10729.29    28.9           0.30  USR
        COM     42,095,742    198,908,111  2980.19     8.0          14.98  COM
        MPI     10,696,252     13,974,228 23366.85    63.0        1672.14  MPI
    SCOREP             41            128     0.15     0.0        1167.92  SCOREP
```

- Additional Score-p measurement configuration variables

```
➜ export SCOREP_TOTAL_MEMORY=4G
➜ export SCOREP_PROFILING_MAX_CALLPATH_DEPTH=1305
```

User instrumentation

```
diff --git a/src/ocean/drivers/mo_hydro_ocean_run.f90 b/src/ocean/drivers/mo_hydro_ocean_run.f90
index 01bb5ba670..4cf562d96f 100644
--- a/src/ocean/drivers/mo_hydro_ocean_run.f90
+++ b/src/ocean/drivers/mo_hydro_ocean_run.f90
@@ -12,6 +12,7 @@
 ! Contains the main stepping routine the 3-dim hydrostatic ocean model.

 !-------------------------------
+#include "scorep/SCOREP_User.inc"
 #include "icon_definitions.inc"
 !-------------------------------
 MODULE mo_hydro_ocean_run
@@ -268,6 +269,7 @@ CONTAINS

     TYPE(datetime), POINTER             :: current_time     => NULL()
     LOGICAL :: lzacc
+    SCOREP_USER_REGION_DEFINE( my_region_handle )

     lzacc = .FALSE.

@@ -366,6 +368,10 @@ CONTAINS

     jstep = jstep0
     TIME_LOOP: DO
+    ! Skip profiling for warmup iterations
+      IF (jstep == 2 ) THEN
+        SCOREP_USER_REGION_BEGIN( my_region_handle, "ocean_main_loop_2nd_iter", SCOREP_USER_REGION_TYPE_COMM
ON )
+      ENDIF

       IF(lsediment_only) THEN
         CALL sed_only_time_step()
@@ -383,6 +389,7 @@ CONTAINS
       END IF

     ENDDO TIME_LOOP
+    SCOREP_USER_REGION_END( my_region_handle )

 #ifdef _OPENACC
     IF ( vert_cor_type == 1 ) THEN
```

include Score-P at the beginning of the file

Define the Score-P region call

Start recording from the 2nd iteration

User instrumentation - Filter file

- To generate suggested filter file do:

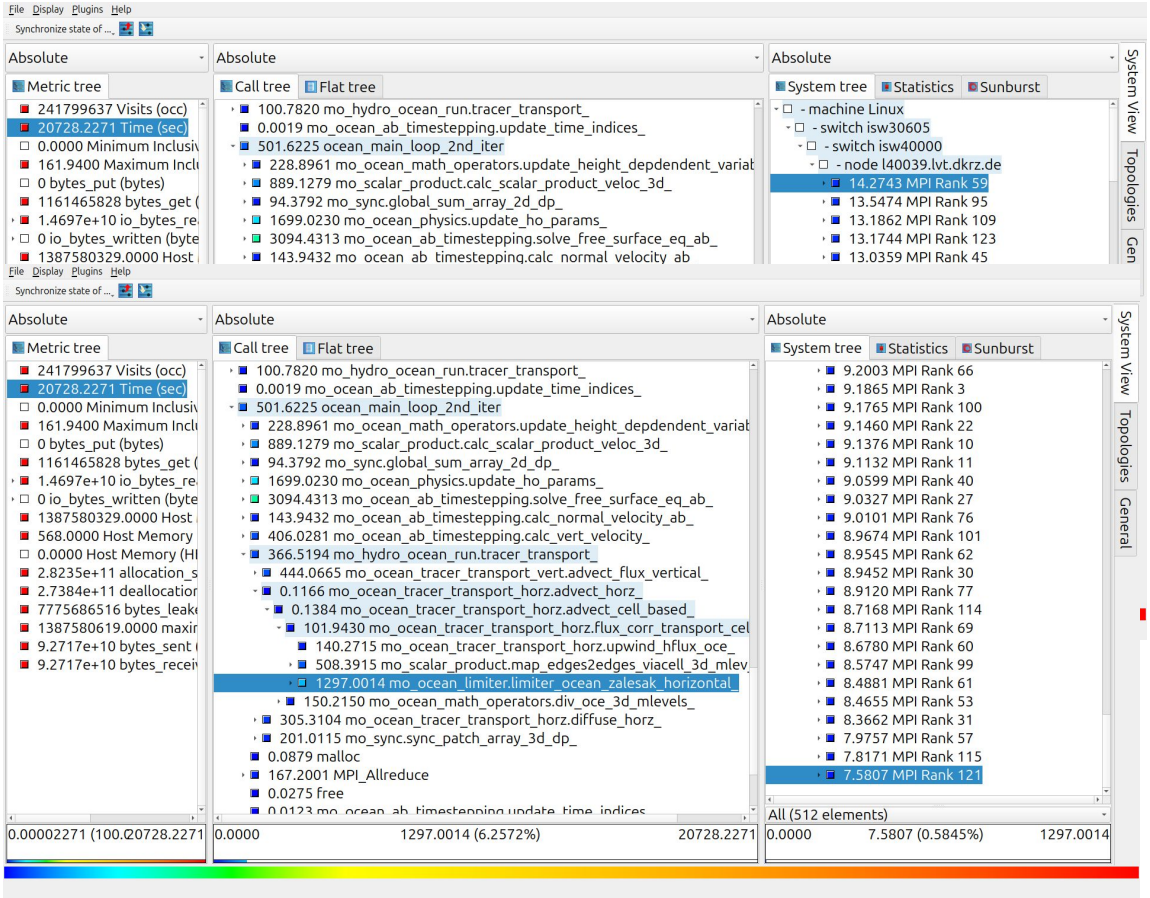
```
scorep-score -g -r profile.cubex
```

- Call paths of interest:

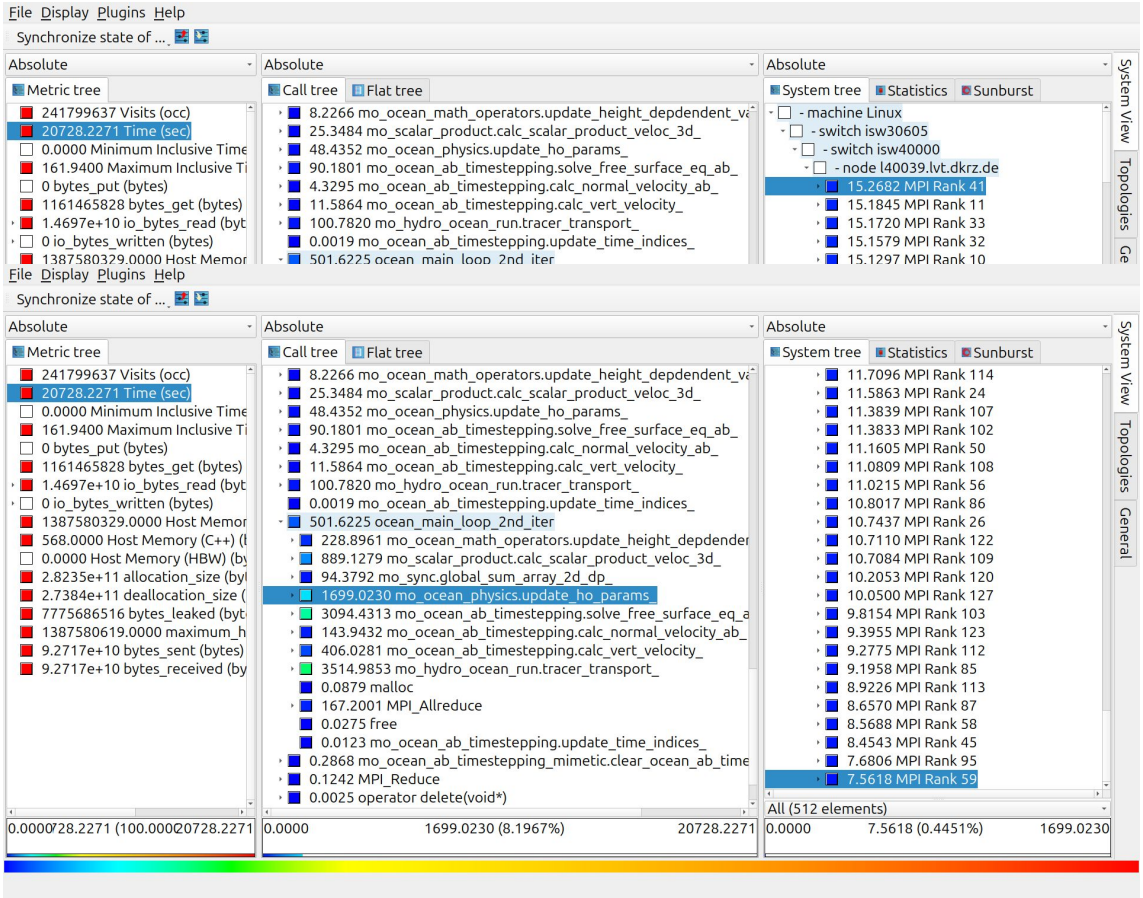
```
➜ export SCOREP_FILTERING_FILE=/path/to/filt.txt
```

```
SCOREP_REGION_NAMES_BEGIN
  EXCLUDE *
    # Exclude high-frequency utility functions that clutter the profile

  INCLUDE
    # User-defined regions - capture everything in these regions
    MANGLED ocean_main_loop_2nd_iter

    MANGLED tracer_transport
    MANGLED *tracer_transport*

    MANGLED update_ho_params
    MANGLED *update_ho_params*

SCOREP_REGION_NAMES_END
```

Load imbalances



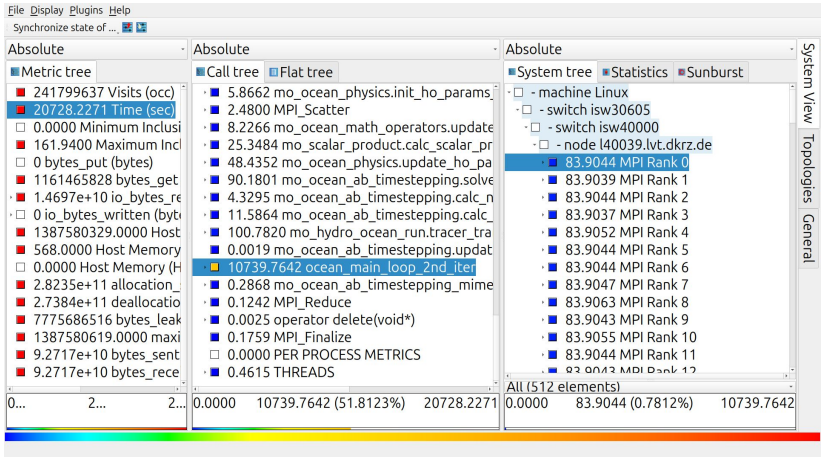
Load imbalances



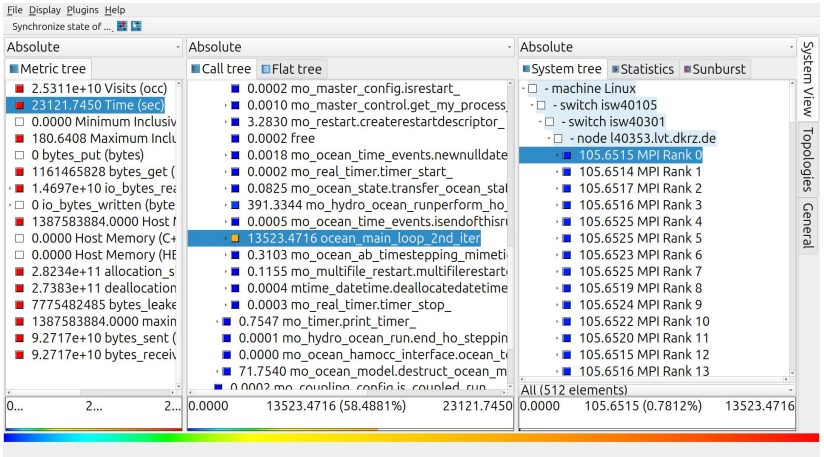
Results

Changing the communication patterns

orig



YAXT



Communication orig vs YAXT

- We gain ~ 10 sec in `mo_read_interface.read_dist_real_3d` in the communication routines using YAXT
- ~ 30 sec loss using YAXT in:
  - `calc_vertical_stability` ~ 8sec loss
  - `calc_tke_scalar` ~ 8sec loss
  - `mo_communication_orig.exchange_data_r3d` ~ 12sec loss in waiting in `mo_ocean_ab_timestepping.solve_free_surface_eq_ab_`

Low level optimizations

```
diff --git a/src/ocean/tracer_transport/mo_ocean_limiter.f90 b/src/ocean/tracer_transport/mo_ocean_limiter.f90
index 1a2ed43f06..8b27246955 100644
--- a/src/ocean/tracer_transport/mo_ocean_limiter.f90
+++ b/src/ocean/tracer_transport/mo_ocean_limiter.f90
@@ -731,6 +731,9 @@ CONTAINS
 !ICON_OMP z_fluxdiv_c, edge_blk1, edge_idx1, edge_blk2, edge_idx2, edge_blk3, edge_idx3) ICON_OMP_DEFAULT_SCHEDULE
     DO blockNo = cells_start_block, cells_end_block
       CALL get_index_range(cells_in_domain, blockNo, start_index, end_index)
+
+      ! Early exit optimization: skip blocks with no ocean cells
+      IF (MAXVAL(dolic_c(start_index:end_index,blockNo)) == 0) CYCLE

 !       z_tracer_new_low(:,:,blockNo)    = 0.0_wp
 !       z_tracer_update_horz(:,:,blockNo)= 0.0_wp
@@ -884,6 +887,9 @@ CONTAINS
 !       r_p(:,:,blockNo) = 0.0_wp

       CALL get_index_range(cells_in_domain, blockNo, start_index, end_index)
+
+      ! Early exit optimization: skip blocks with no ocean cells
+      IF (MAXVAL(dolic_c(start_index:end_index,blockNo)) == 0) CYCLE

       !$ACC PARALLEL DEFAULT(PRESENT) ASYNC(1) IF(lzacc)
       !$ACC LOOP GANG VECTOR PRIVATE(edge_blk1, edge_idx1, edge_blk2, edge_idx2, edge_blk3, edge_idx3) &
@@ -1011,6 +1017,9 @@ CONTAINS
 !ICON_OMP_PARALLEL_DO PRIVATE(start_index, end_index, edge_index, level, z_signum, r_frac) ICON_OMP_DEFAULT_SCHEDULE
     DO blockNo = edges_start_block, edges_end_block
       CALL get_index_range(edges_in_domain, blockNo, start_index, end_index)
+
+      ! Early exit optimization: skip blocks with no ocean edges
+      IF (MAXVAL(dolic_e(start_index:end_index,blockNo)) == 0) CYCLE

       !$ACC KERNELS DEFAULT(PRESENT) ASYNC(1) IF(lzacc)
       flx_tracer_final(:,:,blockNo) = 0.0_wp
```

Low level optimizations

1 Hour

Timers	Old(max)	New(max)
ICON	1min13.5sec	1min10.38sec

Table 1

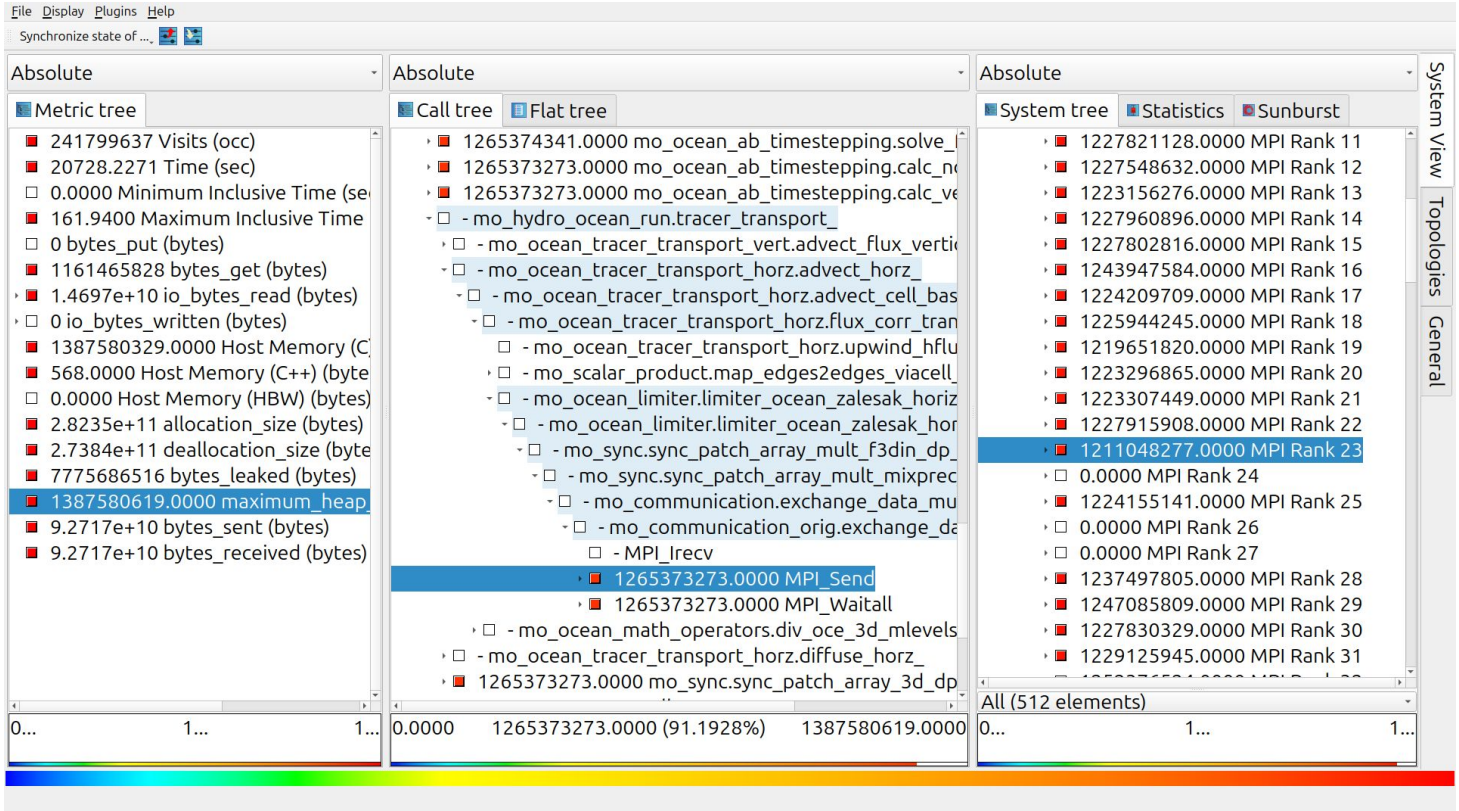
1 Day

Timers	Old(max)	New(max)
ICON	2min07sec	2min05sec

Table 2

Additional Score-P metric

```
➜ export SCOREP_MEMORY_RECORDING=true
```



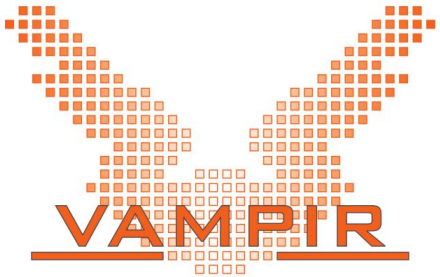
Tracing

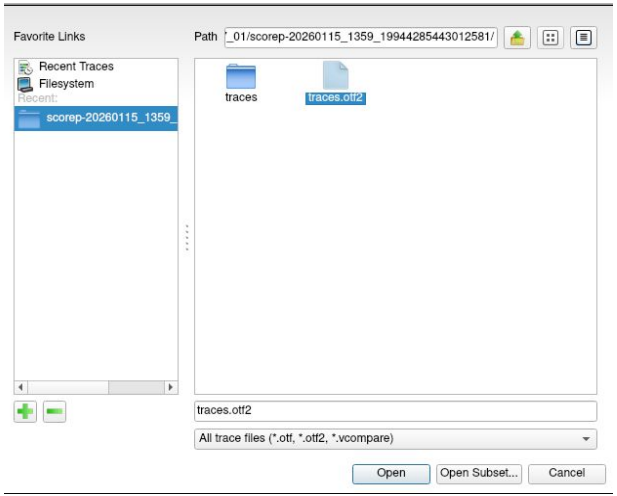
- To enable recording of tracing events

```
➜ export SCOREP_ENABLE_TRACING=true
➜ export SCOREP_ENABLE_PROFILING=false
```

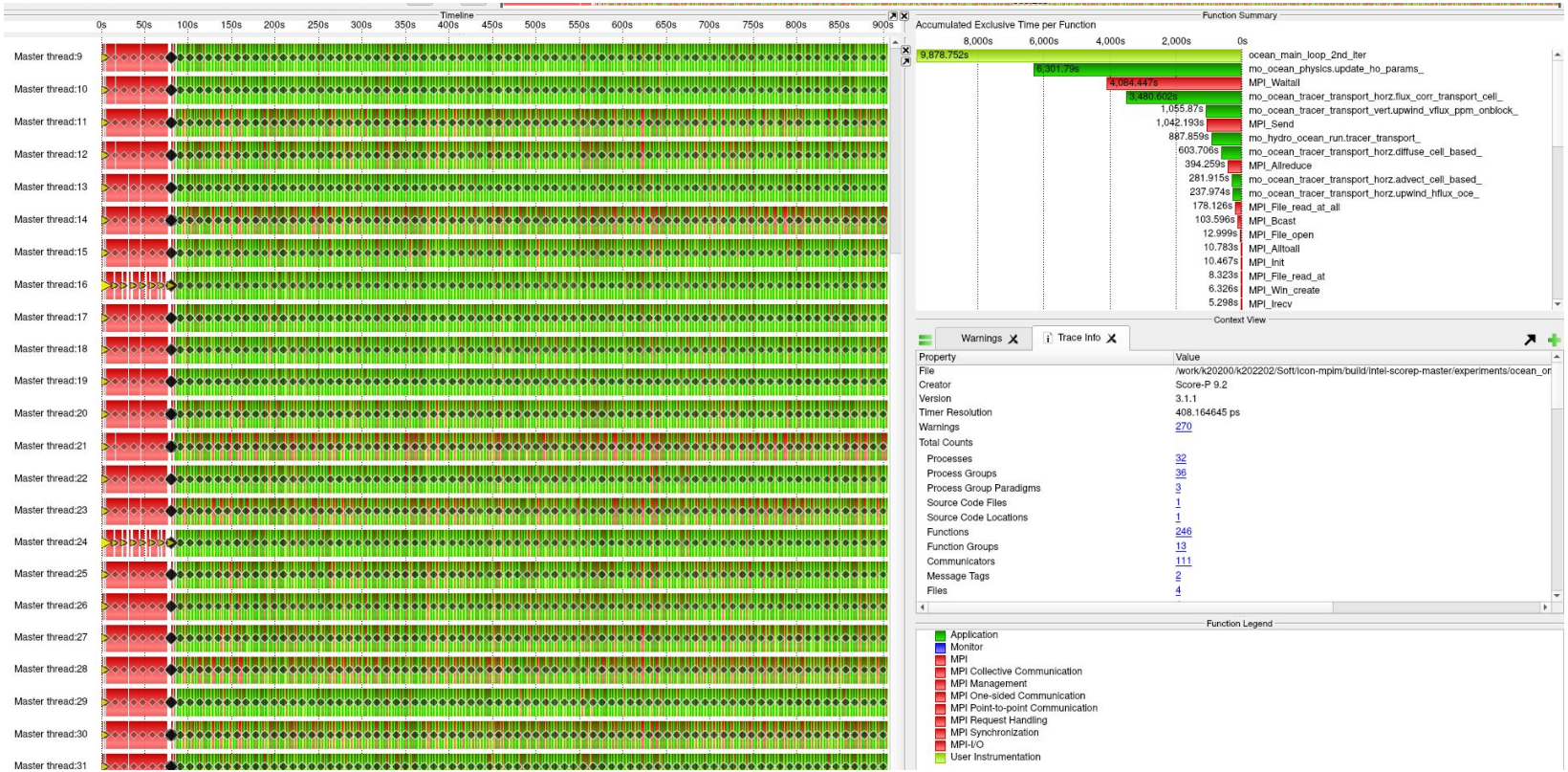
- Vampir Spack package available on Levante

```
➜ spack load vampir
➜ vampir&
```



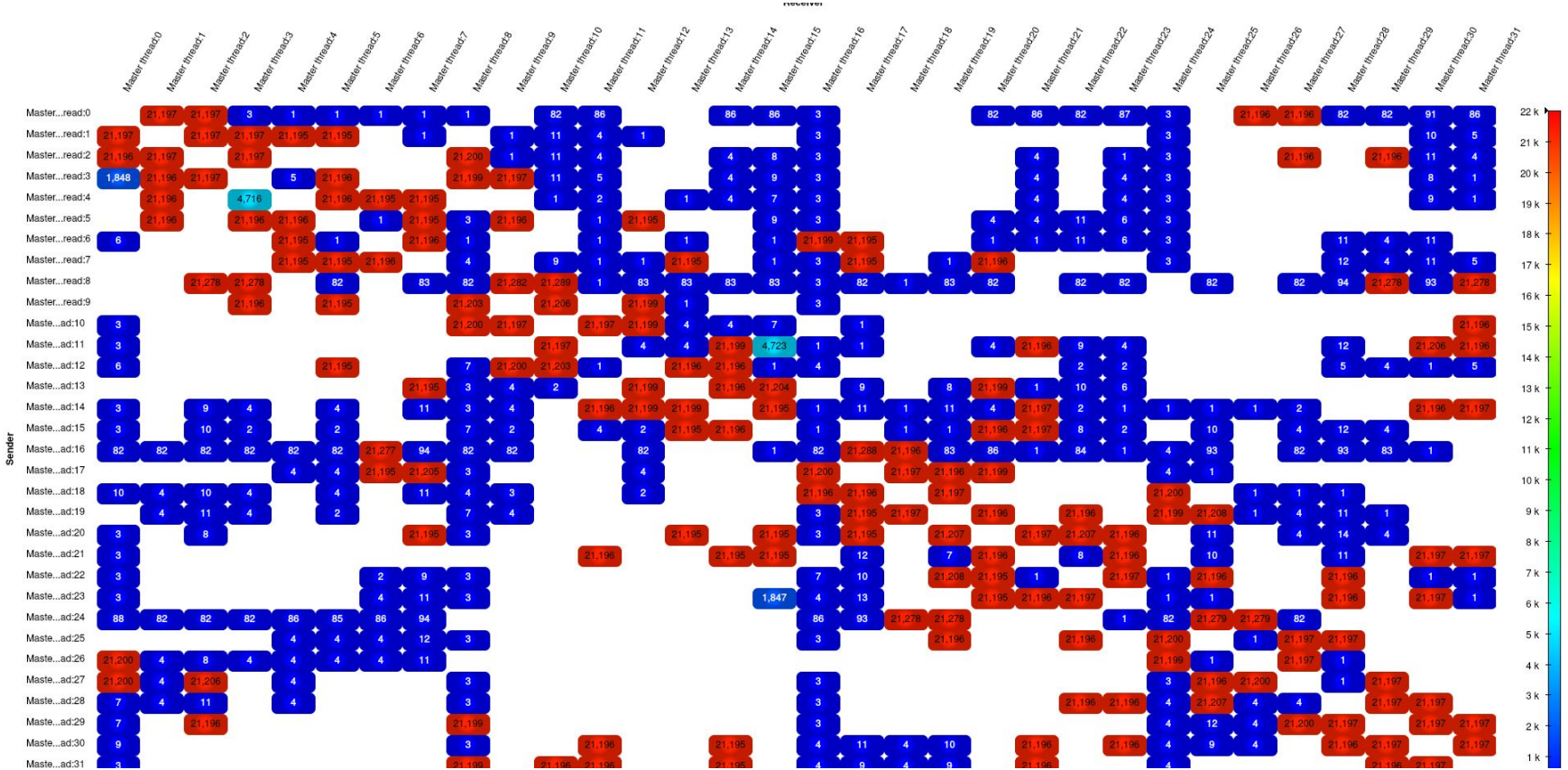


Results - Tracing(i)

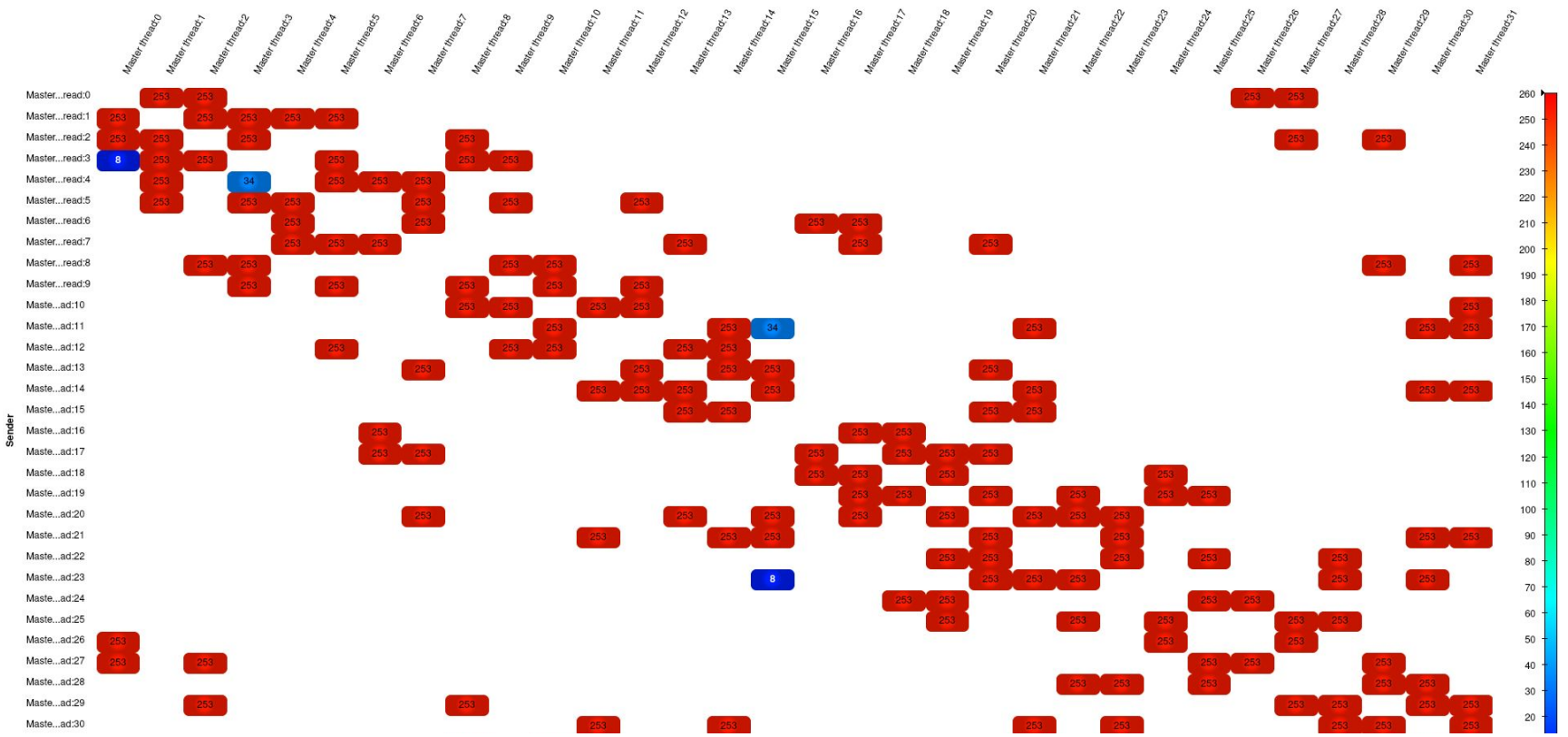


Results - Tracing(ii)

OCEAN - Communication matrix



Time loop communication matrix



ICON case study - II (Dylan)

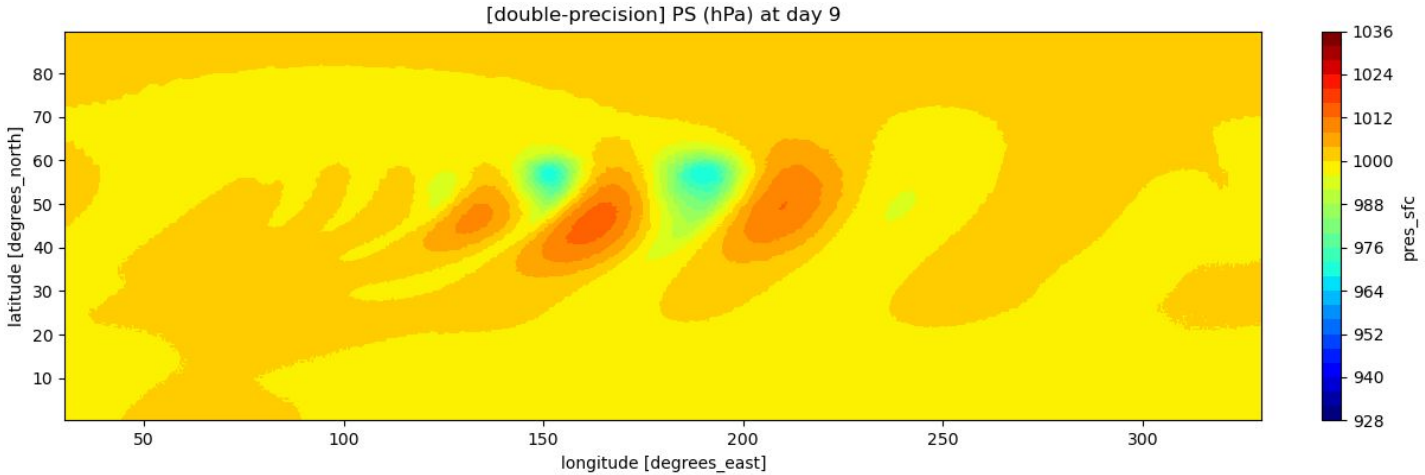
Single-precision ICON development.

- DestinE and WarmWorld projects
- Recent presentation (2025-12)
- Motivation:
  - half bandwidth required
  - half memory footprint
  - some architectures give increased computational throughput

ICON case study - II (Dylan)

Jablonowski-Williamson Baroclinic Instability test for dynamical core.

- ICON tutorial [2]
- Jablonowski Williamson test [1]



ICON case study - II (Dylan)

R2B6 (40km) simulation on Levante AMD 7763 CPU node.

total_avg(s)	total	integrate_nh	model_init	speedup (total time)
cpu, intel, dp	25.971	25.404	4.651	
cpu, intel, mp	18.520	17.853	5.500	1.40
cpu, intel, sp	12.687	12.420	3.477	2.05

R2B6 (40km) simulation on Levante A100 GPU node, using only 1 GPU.

total_avg(s)	total	integrate_nh	model_init	speedup (total time)
gpu, nvhpc, dp	8.124	7.875	163.420?	
gpu, nvhpc, sp	5.254	5.141	121.836?	1.55

ICON case study - II (Dylan)

ScoreP build (including papi for counters):

```
module load intel-oneapi-compilers/2022.0.1-gcc-11.2.0 openmpi/4.1.2-intel-2021.5.0
module load otf2/3.1.1 papi/7.2.0-gcc-11.2.0       # private user modules

wget https://perftools.pages.jsc.fz-juelich.de/cicd/scorep/tags/scorep-9.2/scorep-9.2.tar.gz
tar xvcf scorep-9.2.tar.gz

./configure --with-mpi=openmpi --with-nocross-compiler-suite=intel \
        --enable-shared --with-libgotcha=download \
        –with-otf2=$OTF2_ROOT --with-papi=$PAPI_ROOT
make -j && make install
```

ICON case study - II (Dylan)

ICON build w/ScoreP:

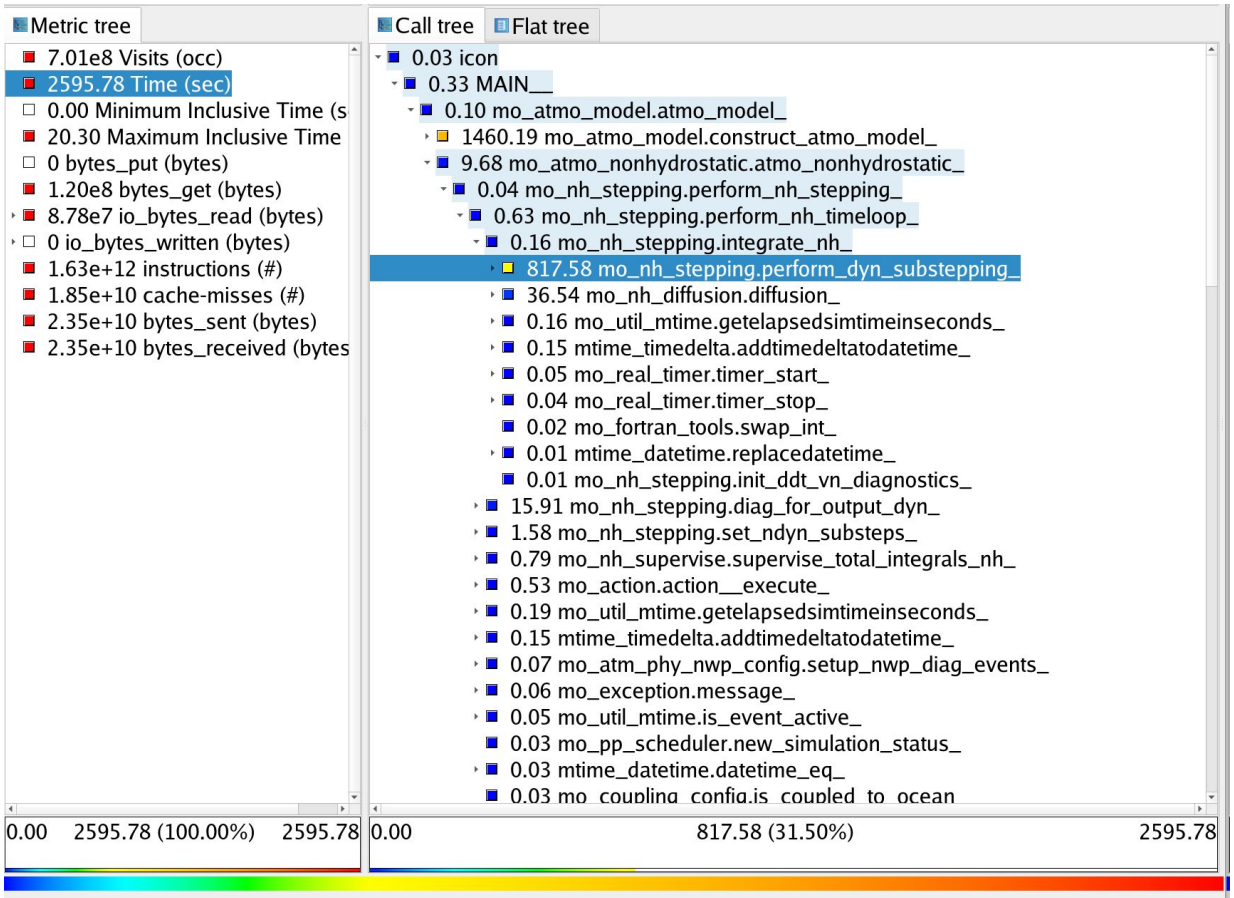
```
# private user module
module load scorep/9.3-openmpi-4.1.2-intel-2021.5.0 # intel openmpi

SCOREP_WRAPPER=off                                    \
SCOREP_WRAPPER_INSTRUMENTER_FLAGS='--no-memory' \
config/dkrz/levante.intel-2021.5.0                    \
        --disable-delayed-config --disable-yaxt --disable-coupling    \
        --disable-comin CC=scorep-mpicc FC=scorep-mpif90
```

ScoreP runtime variables in `*exp.testcase_jabw.run*`:

```
export SCOREP_ENABLE_PROFILING=true
export SCOREP_PROFILING_MAX_CALLPATH_DEPTH=150
export SCOREP_METRIC_PAPI=PAPI_FP_OPS
export SCOREP_METRIC_PERF=instructions,cache-misses
```

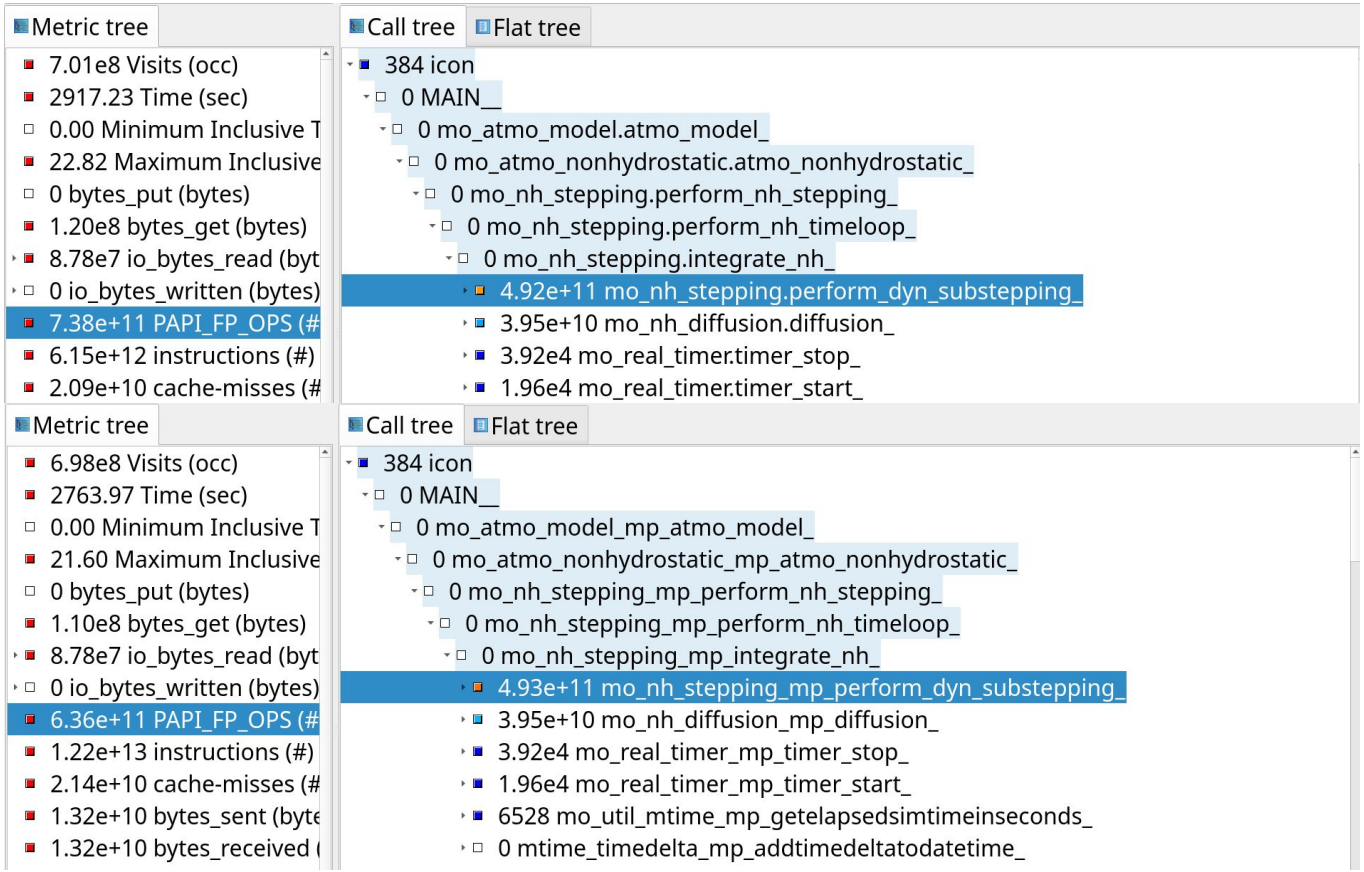
ICON case study - II (Dylan)

A brief look at CPU performance metrics:



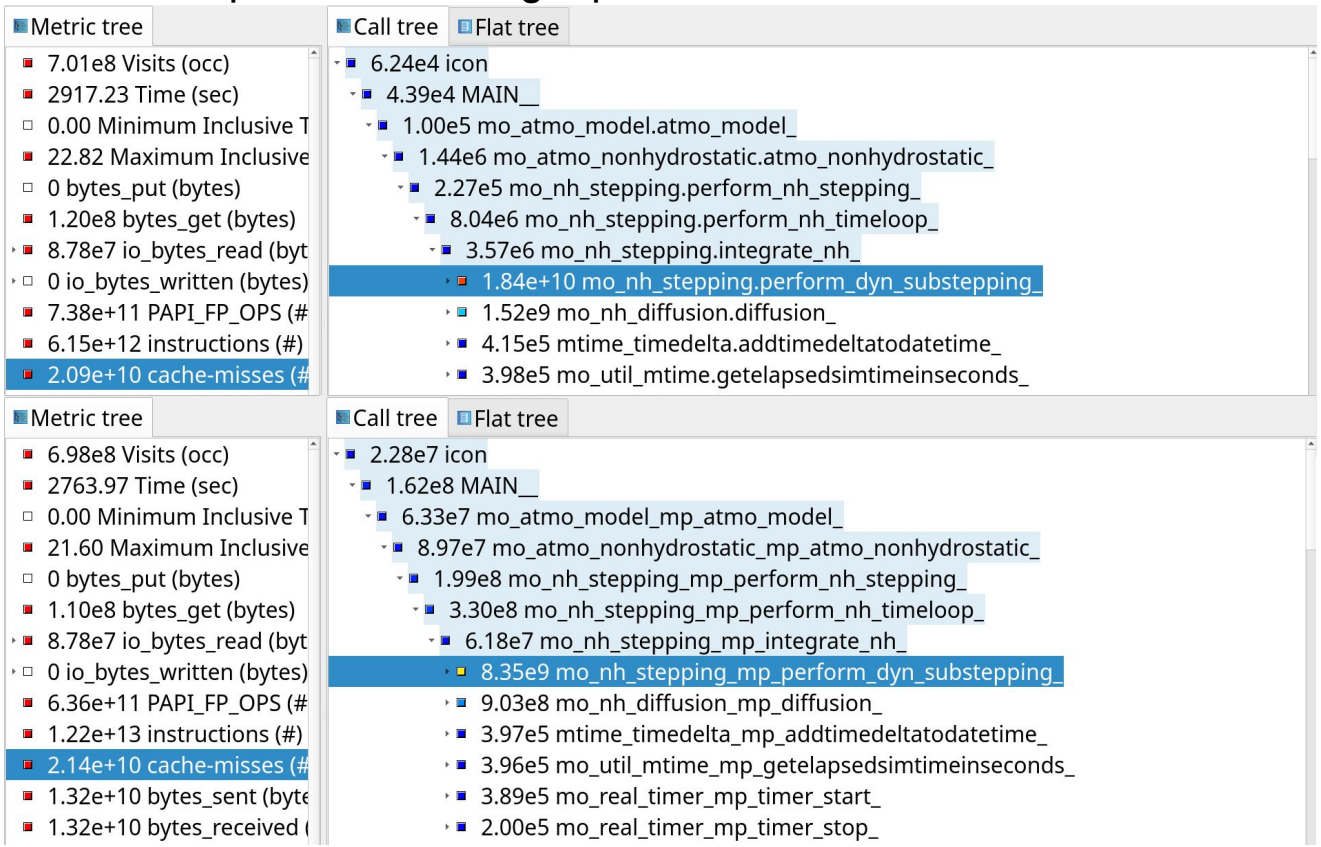
ICON case study - II (Dylan)

(sanity-check) floating point operations double-precision vs single-precision



ICON case study - II (Dylan)

cache-misses double-precision vs single-precision



Summary

Performance engineering is a structured process

- Start from a well-defined performance question
- Prefer profiling first, use tracing selectively
- Always re-measure after optimization

Score-P & Cube provide complementary insight

- Score-P enables scalable measurements
- Cube supports root-cause analysis across code and hardware

ICON case studies highlight typical HPC challenges

- Load imbalance and MPI waiting dominate at scale
- Precision choice has measurable performance impact

Q & A

References

[1] Jablonowski, C., and D. L. Williamson, 2006: A baroclinic instability test case for atmospheric model dynamical cores. Q. J. R. Meteorol. Soc., 132, 2943–2975.

[2] Prill, F., Reinert, D., Rieger, D. and Zängl, G., 2022. ICON tutorial. ICON, 1.