CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

AI-ASSISTED PERFORMANCE ANALYSIS AND OPTIMIZATION FOR PARALLEL SCIENTIFIC CODES INTERNSHIP

Wagui Diallo (CASUS/HZDR)

17th International Parallel Tools Workshop

'IONS

CASUS CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF

JFZ HELMHOLTZ Centre for Environmental Research

Uniwersytet Wrocławski

MAX PLANCK INSTITUTE OF MOLECULAR CELL BIOLOGY AND GENETICS

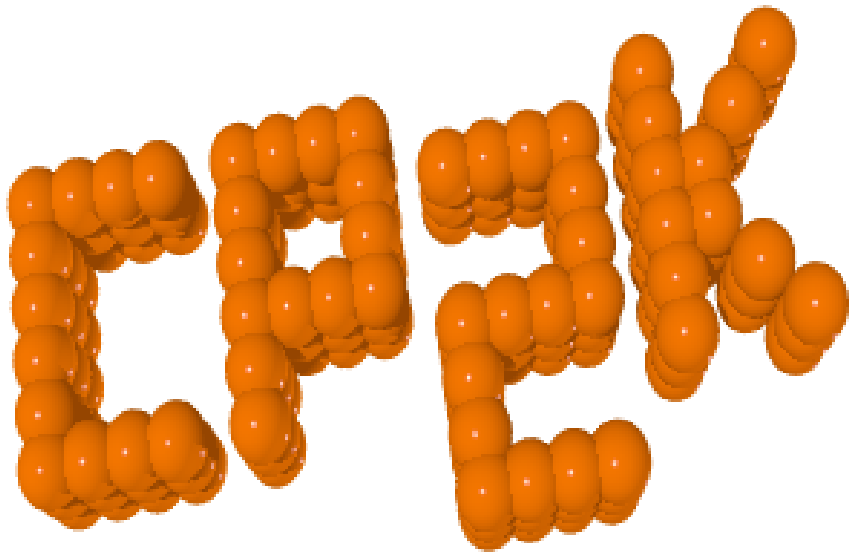
FUNDED BY

Bundesministerium für Forschung, Technologie und Raumfahrt

SACHSEN

Diese Maßnahme wird mitfinanziert mit Steuermitteln auf Grundlage des vom Sächsischen Landtag beschlossenen Haushaltes

GOAL: OPTIMIZING THE TRACES OF CP2K RUN



TRACING WITH CP2K

We are able to generate otf2 traces of CP2K run by using the tracing tool Score-P.



THE SCRIPTS WE ARE USING

We are using python scripts to sparse the data directly from the otf2 trace file. These scripts help us highlight the important highlights of the trace. All of these scripts store their output in a *.csv* and a *.json* file.

The scripts read the multi-gigabyte binary traces directly through the native OTF2 API (the otf2 Python bindings over libotf2), so they are fast and never text-parse the event streams.

THE DIFFERENT SCRIPTS

The git repository

- *trace_overview.py* : Gives a detailed overview of the trace (archive metadata, the process/thread/rank model, definition inventory, regions by paradigm/role, communicator-group sizes, and a per-location table).
- *hotspots.py* : List per process the top functions by inclusive/exclusive time, with a compute-vs-MPI + wait breakdown.
- *comm_analysis.py* : MPI communication analysis (collectives, MPI-call time split by wait/collective/p2p,load-imbalance stats, P2P message/byte matrix).
- *timeline_phases.py* : Detects repeated phases by following the ordered call stack of one process.
- *balance.py* : Give for each rank details about load balance & thread utilization.

Below are screenshots of the terminal output of the scripts.

trace_overview.py

CASUS
CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

```
======================================================================
OTF2 TRACE OVERVIEW
======================================================================

[archive]
  anchor:                                scorep-20260929_1635_83207848791678720
  on-disk size:                          837,864,127 bytes
  OTF2 version:                          3.2.0
  creator:                               Score-P 10.1
  description:                           (none)
  machine:                               (unknown)
  trace id:                              10933514361870235747
  file substrate:                        POSIX
  compression:                           NONE
  SCOREP::MPI_COMMUNICATION_COMPLETE:    true
  SCOREP::MPI_THREAD_LEVEL:              4096
  SCOREP::THREAD_FORK_JOIN_EVENT_COMPLETE: true
  SCOREP::THREAD_CREATE_WAIT_EVENT_COMPLETE: true
  SCOREP::THREAD_LOCK_EVENT_COMPLETE:    true
  SCOREP::PTHREAD_LOCATION_REUSED:       false

[timing]
  timer resolution:                      2,999,961,628 ticks/s
  wall-clock length:                     2 m 39.5 s

[locations / parallelism]
  locations:                             2
  processes:                             2
  MPI ranks detected:                    True
  threads per process:                   1 .. 1
  events per process:                    31,986,701 .. 32,045,673
  total events:                          64,032,374

[definitions]
  locations:                             2
  location_groups:                       2
  regions:                               2,377
  callsites:                             0
  callpaths:                             36,299
  calling_contexts:                      0
  groups:                                8
  comms:                                 330
  metrics:                               0
  metric_members:                        8
  parameters:                            4
  attributes:                            4
  cart_topologies:                       206
  rma_wins:                              0
  string_definitions:                    4,968

[regions]
  COMPILER                 2,143
  MPI                        231
  MEASUREMENT_SYSTEM           2
  USER                         1
  roles: FUNCTION=2,255, FILE_IO=35, COLL_ALL2ALL=26, POINT2POINT=23, ATOMIC=6, COLL_ONE2ALL=6, COLL_ALL2ONE=6, FILE_IO_METADATA=5

[communicator group sizes]
  COMM_LOCATION_GROUPS:  size 2 x1
  COMM_GROUP:            size 1 x4, size 2 x1
  COMM_SELF:             size 0 x1
  COMM_LOCATIONS:        size 2 x1

[locations]
  ref    name                 process        events       rank   thr#
  0      Master thread        MPI Rank 0     31,986,701   0      0
  1      Master thread        MPI Rank 1     32,045,673   1      0


======================================================================
```

hotspots.py

CASUS
CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

```
================================================================================
HOTSPOTS (aggregate time, selection: all)
================================================================================

[pass]
  locations processed:                2
  events read:                        64,032,374
  pass duration:                      1 m 54.4 s
  wall-clock length:                  2 m 39.5 s
  aggregate inclusive time:           1 h 41 m
  active regions:                     2,184
  ordering:                           inclusive time, descending

[breakdown]
  COMPILER paradigm:                  1 h 41 m  ( 99.7%)
  MPI paradigm:                       17.068 s  (  0.3%)
  MPI, inside calls:                  17.068 s
    - role COLL_ALL2ALL:              6.114 s
    - role POINT2POINT:               5.659 s
    - role FUNCTION:                  5.156 s
    - role COLL_ONE2ALL:              0.131 s
    - role COLL_ALL2ONE:              0.006 s
    - role BARRIER:                   0.000841 s
  MPI wait-like calls:                6.754 s  ( 39.6% of MPI)
  compute (COMPILER+USER):            1 h 41 m

[top 25 regions by aggregate inclusive time]
  #                                          region  calls      inc_s      exc_s   %agg      p-min     p-mean      p-max
  1                                            cp2k      2    318.917      0.060    5.2    159.447    159.458    159.470
  2                           cp2k_runs::run_input      2    317.185      0.000    5.2    158.592    158.592    158.593
  3                            cp2k_runs::cp2k_run      2    317.185      0.001    5.2    158.592    158.592    158.593
  4                             md_run::qs_mol_dyn      2    316.350      0.000    5.2    158.162    158.175    158.188
  5                         md_run::qs_mol_dyn_low      2    316.349      0.026    5.2    158.161    158.175    158.188
  6   force_env_methods::force_env_calc_energy_for      4    316.162      0.000    5.2    158.081    158.081    158.081
  7                 qs_force::qs_calc_energy_force      4    316.160      0.000    5.2    158.080    158.080    158.080
  8                           qs_force::qs_forces      4    316.160      0.000    5.2    158.080    158.080    158.080
  9                         qs_energy::qs_energies      4    305.442      0.026    5.0    152.721    152.721    152.721
  10                                  qs_scf::scf      4    300.996      0.026    4.9    150.498    150.498    150.498
  11                       qs_scf::scf_env_do_scf      4    294.005      0.002    4.8    147.003    147.003    147.003
  12           qs_ks_methods::qs_ks_update_qs_env    134    140.719      0.001    2.3     70.359     70.359     70.360
  13             qs_ks_methods::rebuild_ks_matrix    130    139.557      0.000    2.3     69.777     69.778     69.780
  14   qs_ks_methods::qs_ks_build_kohn_sham_matrix    130    139.556      0.004    2.3     69.776     69.778     69.780
  15            qs_rho_methods::qs_rho_update_rho    130    125.905      0.000    2.1     62.952     62.953     62.953
  16         qs_rho_methods::qs_rho_update_rho_low    130    125.905      0.000    2.1     62.952     62.952     62.953
  17      qs_collocate_density::calculate_rho_elec    130    125.904      0.001    2.1     62.951     62.952     62.953
  18          qs_scf_loop_utils::qs_scf_rho_update    126    121.634      0.000    2.0     60.817     60.817     60.818
  19            qs_ks_utils::sum_up_and_integrate    130    116.736      0.001    1.9     57.573     58.368     59.164
  20  qs_integrate_potential_product::integrate_v_    130    116.610      0.001    1.9     57.509     58.305     59.101
  21           grid_api::grid_collocate_task_list    130    110.062      0.000    1.8     54.750     55.031     55.312
  22                     grid_collocate_task_list    130    110.060      0.000    1.8     54.749     55.030     55.311
  23                 grid_cpu_collocate_task_list    130    110.059      0.002    1.8     54.749     55.030     55.310
  24                    collocate_one_grid_level    520    110.057    110.057    1.8     54.747     55.029     55.310
  25           grid_api::grid_integrate_task_list    130    108.074      0.000    1.8     53.245     54.037     54.830
  ... (2184 regions total, full table in hotspots_top.csv)

================================================================================
```

comm_analysis.py

CASUS
CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

```
======================================================================
MPI COMMUNICATION ANALYSIS
======================================================================

[pass]
  processes:                        2
  events read:                      64,032,374
  pass duration:                    1 m 30.9 s
  wall-clock length:                2 m 39.5 s

[MPI totals]
  MPI inclusive time:               17.068 s
    of which wait-like:             4.707 s ( 27.6%)
  collectives:                      109,262 calls, 6.238 s
  comm-management collectives:      1,302 calls, 1.338 s
  p2p sends / receives:             45,890 / 45,890 msgs, 9,146,445,968 / 9,146,445,968 bytes
  nonblocking sends:                35,328 (99.99% completed)
  nonblocking completion avg:       2.056 ms
  request tests (polls):            0
  comm create / destroy:            656 / 650
  nonblocking collectives:          250 req / 250 comp

[MPI time by role]
  COLL_ALL2ALL:                     6.114 s ( 35.8%)
  POINT2POINT:                      5.659 s ( 33.2%)
  FUNCTION:                         5.156 s ( 30.2%)
  COLL_ONE2ALL:                     0.131 s (  0.8%)
  COLL_ALL2ONE:                     0.006 s (  0.0%)
  BARRIER:                          0.000841 s (  0.0%)

[collectives (top 12 by time)]
  op                          count     time_s     avg_ms   bytes_sent    bytes_rcv    proc#
  ALLTOALLV                   4,452      2.805      0.630 10,309,802,060 10,309,802,060 2226-2226
  ALLREDUCE                  84,812      2.550      0.030  299,210,160  299,210,160 42406-42406
  ALLTOALL                    6,112      0.320      0.052      142,272      142,272 3056-3056
  ALLGATHER                   2,784      0.298      0.107    2,275,072    2,275,072 1392-1392
  BCAST                      10,082      0.130      0.013  113,279,688  113,279,688 4991-5091
  REDUCE_SCATTER_BLOCK            8      0.128     16.012    4,194,304    4,194,304      4-4
  REDUCE                        800      0.006      0.007   52,428,800   52,428,800  400-400
  BARRIER                       212      0.001      0.004            0            0  106-106

[p2p matrix]
  nonzero cells / density:          4 / 100.0% (max cell 4,491,549,100 bytes)
  top 10 pairs by bytes:
  from                        to                            msgs        bytes
  MPI Rank 1                  MPI Rank 0                  20,264 4,491,549,100
  MPI Rank 0                  MPI Rank 1                  17,148 4,416,978,708
  MPI Rank 0                  MPI Rank 0                   5,652   159,421,600
  MPI Rank 1                  MPI Rank 1                   2,826    78,496,560

[per-process MPI summary]
  process        wall_s    mpi_s    mpi%   wait_s      coll#     sent_B     recv_B      nb_ms
  MPI Rank 0      159.4      9.1    5.7%      2.3     54,681 4,576,400,308 4,574,112,540      2.857
  MPI Rank 1      159.5      8.0    5.0%      2.4     54,581 4,570,045,660 4,572,333,428      1.265

[load imbalance (across processes)]
  mpi_inclusive_s:                  min=7.95 mean=8.53 max=9.12
  mpi_percent:                      min=4.99 mean=5.35 max=5.72
  mpi_wait_s:                       min=2.29 mean=2.35 max=2.42
  sent_bytes:                       min=4.57e+09 mean=4.57e+09 max=4.58e+09
  recv_bytes:                       min=4.57e+09 mean=4.57e+09 max=4.57e+09
  collective_count:                 min=5.46e+04 mean=5.46e+04 max=5.47e+04

======================================================================
```

timeline_phases.py

```
======================================================================
TIMELINE PHASE ANALYSIS (call-stack method, rank 0)
======================================================================

[pass]
  locations (rank 0):             31,986,701 events
  pass duration:                  2 m 37.7 s

[phase classes (15)]
  id  reps   count-slices signature
  0   1966080    1966080 qs_energy::qs_energies||qs_scf::scf||qs_scf_initiallization::qs_scf_env_initialize||qs_scf_initialization::init_scf_run||qs_scf_initialization::scf_env_initial_rho_setup||qs_initial_guess::calculate_first_density_matrix||cp_fm_types::cp_fm_init_random||parallel_rng_types::fill_1||parallel_rng_
types::next_real||parallel_rng_types::rn53
  1   1639415    1639415 qs_energy::qs_energies||qs_scf::scf||qs_scf::scf_env_do_scf||qs_scf_loop_utils::qs_scf_new_mos||qs_scf_loop_utils::qs_scf_loop_do_ot||qs_ot_scf::ot_scf_mini||qs_ot::qs_ot_get_p||qs_ot::qs_ot_p2m_diag
  2   1638400    1638400 qs_energy::qs_energies||qs_scf::scf||qs_scf::scf_env_do_scf||qs_scf_loop_utils::qs_scf_new_mos||qs_scf_loop_utils::qs_scf_loop_do_ot||qs_ot_scf::ot_scf_mini||qs_ot::qs_ot_get_p||qs_ot::qs_ot_p2m_diag||qs_ot::qs_ot_sincf
  3   1310720    1310720 qs_energy::qs_energies||qs_scf::scf||qs_scf_initialization::qs_scf_env_initialize||qs_scf_initialization::init_scf_run||qs_scf_initialization::scf_env_initial_rho_setup||qs_initial_guess::calculate_first_density_matrix||cp_fm_types::cp_fm_init_random||parallel_rng_types::fill_1||parallel_rng_
types::next_real
  4   1310720    1310720 qs_energy::qs_energies||qs_scf::scf||qs_scf_initialization::qs_scf_env_initialize||qs_scf_initialization::init_scf_run||qs_scf_initialization::scf_env_initial_rho_setup||qs_initial_guess::calculate_first_density_matrix||cp_fm_types::cp_fm_init_random||parallel_rng_types::fill_1||parallel_rng_
types::next_real||parallel_rng_types::rn53||parallel_rng_types::rn32
  5   905030     905030 qs_energy::qs_energies||qs_scf::scf||qs_scf::scf_env_do_scf||qs_ks_methods::qs_ks_update_qs_env||qs_ks_methods::evaluate_core_matrix_traces||qs_core_energies::calculate_ptrace_kp||cp_dbcsr_contrib::dbcsr_dot
  6   903854     903854 qs_energy::qs_energies||qs_scf::scf||qs_scf::scf_env_do_scf||qs_ks_methods::qs_ks_update_qs_env||qs_ks_methods::evaluate_core_matrix_traces||qs_core_energies::calculate_ptrace_kp||cp_dbcsr_contrib::dbcsr_dot||cp_dbcsr_api::dbcsr_get_readonly_block_p
  7   655616     655616 qs_energy::qs_energies||qs_scf::scf||qs_scf_initialization::qs_scf_env_initialize||qs_scf_initialization::init_scf_run||qs_scf_initialization::scf_env_initial_rho_setup||qs_initial_guess::calculate_first_density_matrix||cp_fm_types::cp_fm_init_random||parallel_rng_types::fill_1
  8   624294     624294 qs_energy::qs_energies||qs_energy_init::qs_energies_init||qs_energy_init::qs_energies_init_hamiltonians||qs_update_s_mstruct::qs_env_update_s_mstruct||qs_update_s_mstruct::qs_create_task_list||task_list_methods::generate_qs_task_list||task_list_methods::task_list_inner_loop||task_list_methods:
:pgf_to_tasks
  9   624288     624288 qs_energy::qs_energies||qs_energy_init::qs_energies_init||qs_energy_init::qs_energies_init_hamiltonians||qs_update_s_mstruct::qs_env_update_s_mstruct||qs_update_s_mstruct::qs_create_task_list||task_list_methods::generate_qs_task_list||task_list_methods::task_list_inner_loop||task_list_methods:
:compute_pgf_properties
  10  624172     624172 velocity_verlet_control::velocity_verlet||integrator::nve||qs_energy::qs_energies||qs_energy_init::qs_energies_init||qs_energy_init::qs_energies_init_hamiltonians||qs_update_s_mstruct::qs_env_update_s_mstruct||qs_update_s_mstruct::qs_create_task_list||task_list_methods::generate_qs_task_list||
task_list_methods::task_list_inner_loop||task_list_methods::compute_pgf_properties
  11  624172     624172 velocity_verlet_control::velocity_verlet||integrator::nve||qs_energy::qs_energies||qs_energy_init::qs_energies_init||qs_energy_init::qs_energies_init_hamiltonians||qs_update_s_mstruct::qs_env_update_s_mstruct||qs_update_s_mstruct::qs_create_task_list||task_list_methods::generate_qs_task_list||
task_list_methods::task_list_inner_loop||task_list_methods::pgf_to_tasks
  12  495008     495008 qs_energy::qs_energies||qs_energy_init::qs_energies_init||qs_energy_init::qs_energies_init_hamiltonians||qs_update_s_mstruct::qs_env_update_s_mstruct||qs_update_s_mstruct::qs_create_task_list||task_list_methods::generate_qs_task_list||task_list_methods::task_list_inner_loop||task_list_methods:
:compute_pgf_properties||ao_util::exp_radius_very_extended
  13  494949     494949 velocity_verlet_control::velocity_verlet||integrator::nve||qs_energy::qs_energies||qs_energy_init::qs_energies_init||qs_energy_init::qs_energies_init_hamiltonians||qs_update_s_mstruct::qs_env_update_s_mstruct||qs_update_s_mstruct::qs_create_task_list||task_list_methods::generate_qs_task_list||
task_list_methods::task_list_inner_loop||task_list_methods::compute_pgf_properties||ao_util::exp_radius_very_extended
  14  487354     487354 qs_energy::qs_energies||qs_energy_init::qs_energies_init||qs_energy_init::qs_energies_init_hamiltonians||qs_update_s_mstruct::qs_env_update_s_mstruct||qs_update_s_mstruct::qs_create_task_list||task_list_methods::generate_qs_task_list||task_list_methods::task_list_inner_loop

[phases (859)]
  #     class                      start_s      end_s      dur_s     enter out?
  1     other[cp2k,run_input,cp2k_      0.000      1.838      1.838 2,949,521
  2     qs_energy::qs_energies||qs      1.864      1.902      0.038   453,878
  3     qs_energy::qs_energies||qs      1.902      1.904      0.002    26,873
  4     qs_energy::qs_energies||qs      1.904      1.905      0.001    13,425
  5     qs_energy::qs_energies||qs      1.905      1.907      0.002    24,441
  6     qs_energy::qs_energies||qs      1.907      1.908      0.001    10,397
  7     qs_energy::qs_energies||qs      1.908      1.909      0.001    13,414
  8     qs_energy::qs_energies||qs      1.909      1.911      0.002    26,526
  9     qs_energy::qs_energies||qs      1.911      1.914      0.003    35,101
  10    qs_energy::qs_energies||qs      1.914      1.917      0.003    40,255 *
  11    qs_energy::qs_energies||qs      1.917      1.918      0.001    12,834
  12    qs_energy::qs_energies||qs      1.918      1.921      0.003    35,554 *
  13    qs_energy::qs_energies||qs      1.921      1.924      0.003    39,738
  14    qs_energy::qs_energies||qs      1.924      1.925      0.001     8,134
  15    qs_energy::qs_energies||qs      1.925      1.927      0.002    21,170
  16    qs_energy::qs_energies||qs      1.927      1.929      0.002    26,957
  17    qs_energy::qs_energies||qs      1.929      1.930      0.001    12,969
  18    qs_energy::qs_energies||qs      1.930      1.931      0.001    13,544
  19    qs_energy::qs_energies||qs      1.931      1.933      0.002    26,039
  20    qs_energy::qs_energies||qs      1.933      1.934      0.001     8,062
  21    qs_energy::qs_energies||qs      1.934      1.936      0.002    27,016
  22    qs_energy::qs_energies||qs      1.936      1.937      0.001    13,515
  23    qs_energy::qs_energies||qs      1.937      1.938      0.001    12,918
  24    qs_energy::qs_energies||qs      1.938      1.939      0.001    13,487
  25    qs_energy::qs_energies||qs      1.939      1.942      0.003    34,795
  ... (859 phases total, all in phases.csv)
  * = duration outlier (> mean+2*sd of its class); 47 outliers total

======================================================================
```

balance.py

```
================================================================================
>>> balance     /usr/bin/python3 /data/home2/diallo39/cp2k-tests/tools/balance.py scorep-20260929_1635_83207848791678720/
================================================================================
reading: scorep-20260929_1635_83207848791678720
analysing load balance (single ENTER/LEAVE pass)...
================================================================================
LOAD BALANCE ANALYSIS
================================================================================

[pass]
  processes:                      2
  threads:                        2
  events read:                    64,032,374
  pass duration:                  1 m 28.6 s
  wall-clock length:              2 m 39.5 s

[load balance]
  load balance (mean/max busy):    99.61%
  busy_s:                         min=150.331  mean=150.924  max=151.518
  busy_percent:                   min=94.280  mean=94.645  max=95.010
  idle_s:                         min=7.952  mean=8.534  max=9.116
  mpi_s:                          min=7.952  mean=8.534  max=9.116
  total busy / idle:              5 m 1.8 s / 17.068 s (5.35% idle)

[per-process]
  process          thr    wall_s   busy%   busy_s   idle_s    mpi_s    mpi%    mstr%   orph%
  MPI Rank 0         1    159.45    94.3   150.33     9.12     9.12     5.7     94.3       -
  MPI Rank 1         1    159.47    95.0   151.52     7.95     7.95     5.0     95.0       -

most idle process:             MPI Rank 0 (9.116 s idle, 5.72%)

================================================================================
```

CASUS
CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

TRACE REPORT

Once we have the data of all the previous scripts, we use a final script *report.py* which load the JSON results of the five analysis tools and give us a detailed report of the most important details of the trace, and based on different parameters, gives us the bottlenecks that could be fixed and some recommendations that could improve the trace. (Example below)

Every bottlenecks/recommendations is a predictable template with a tunable treshold (example: if the load imbalance variable is greater than the treshold, the report prings a "rebalance the domain decomposition, inspect per-process busy times in balance").

REPORT EXAMPLE

CASUS
CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

```
================================================================================
OTF2 TRACE BOTTLENECK REPORT
================================================================================

Note: results missing for: phases (run with --run to compute).

[run facts]
  creator      : Score-P 10.1
  processes    : 8   locations(threads): 32
  wall-clock   : 155.793 s
  events       : 820,605,690

[work distribution]
  load balance (mean/max busy): 100.00%
  process busy / idle          : 99.2% / 0.8%
  master / worker busy        : 44.0% / 99.1%  (gap 55.1 pt)
  MPI time (per rank)          : 87.22 s  ( 56.0% of wall)
  heaviest compute kernels (incl. time / rank):
  main dispatchers (small self time):
    cp2k                                    155.77 s/rank
    cp2k_runs::run_input                    154.55 s/rank
    cp2k_runs::cp2k_run                     154.55 s/rank
    md_run::qs_mol_dyn                      154.18 s/rank
    md_run::qs_mol_dyn_low                  154.18 s/rank
    force_env_methods::force_env_calc_energy  154.13 s/rank

[communication]
  MPI % of wall             : 56.0%
  wait-like MPI (Wait/Test/Recv/Probe): 51.9% of MPI
  collectives                : 801,928 (268.66 s, +12,392 comm-mgmt)
  p2p messages (sent side)    : 613,754 (431,107,172,760 bytes)
    ALLTOALLV              37832 calls  124.654 s  (avg 3.295 ms)
    ALLREDUCE             593384 calls   48.296 s  (avg 0.081 ms)
    ALLTOALL               38688 calls   35.648 s  (avg 0.921 ms)
    ALLGATHER              17688 calls   33.021 s  (avg 1.867 ms)
    REDUCE                 46888 calls   13.859 s  (avg 0.296 ms)
  top p2p pairs (bytes):
    MPI Rank 1 -> MPI Rank 0  39,759,228,848 bytes  (36,928 msgs)
    MPI Rank 6 -> MPI Rank 5  39,504,719,104 bytes  (20,120 msgs)
    MPI Rank 2 -> MPI Rank 1  38,127,772,716 bytes  (34,742 msgs)

[time structure]

[bottlenecks (ranked)]
  1. Communication-bound: 56% of wall in MPI, 52% of that wait-like. Reduce MPI ranks or enlarge the system.
  2. MPI is dominated by waiting: 52% of MPI time in Wait/Test/Recv/Probe.
  3. 3D-FFT redistribution dominates MPI: 35% of MPI time in ALLTOALL/ALLTOALLV.
  4. Hybrid thread imbalance: worker busy 99% vs master 44% (gap 55 pt).

[recommendations]
  * Communication-bound: test fewer MPI ranks with more threads each; watch the top collectives (ALLTOALLV, ALLREDUCE, ALLTOALL) and MPI waits.
  * Overlap waits: more threads per rank / fewer sync points; profile the Waitall/Testall hotspots.
  * Tune the plane-wave grid layout / FFT backend / 2D decomposition, or reduce rank count to shrink the alltoallv volume.
  * Thread imbalance: the worker thread is the bottleneck; add worker threads or offload more work from master to worker (grid/FFT).
  * Large trace (820605690 events): for production use Score-P filtering/whitelisting, SION, or profiling-only (CUBEX).
```

TRACE COMPARAISON

If we have genereted a report for two similar trace, we can use the script *compare_trace.py* to get a detailed comparaison between the two traces and pin point what parameters improved and which one regressed. (Example below)

COMPARAISON EXAMPLE

CASUS
CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

```
================================================================================================
TRACE COMPARISON
================================================================================================
metric                                H2O128-8*4                  H2O128-4*8
------------------------------------------------------------------------------------------------

[runtime]
  Wall time                           1 m 8.5 s                   1 m 42.9 s                  regresses
  MPI time (of wall)                  49.1                        54.0                        regresses

[comm]
  MPI wait time (aggregate)           2 m 28.1 s                  1 m 54.9 s                  improves
  Collectives                         651,464                     323,404                     improves
  Comm-mgmt collectives               5,048                       2,524                       improves
  P2P messages (sent)                 429,284                     152,648                     improves
  P2P bytes (sent)                    212,123,470,416             103,230,872,736             improves

[threads]
  Process busy                        93.9                        96.2                        improves
  Process idle (both threads)         6.1                         3.8                         improves
  Master thread busy                  50.9                        46.0                        regresses
  Worker thread busy                  93.9                        96.2                        improves
  Thread gap                          43.0                        50.2                        regresses
  Load balance                        100.0                       100.0

[trace]
  Trace events                        311,090,822                 296,584,154                 improves

[phases]
  Phases detected                     -                           74
  Mean phase duration                 -                           1.391 s
  Outlier phases                      -                           2
  Marker-kernel time (of wall)        -                           82.2

[hotspots]
  #1 hotspot (per rank)               collocate_one_grid_level (10.5 -

note: missing result sets -> A:phases B:-

================================================================================================
WHAT IMPROVED (in H2O128-4*8)
================================================================================================
  + MPI wait time (aggregate) (22% change)
  + Collectives (50% change)
  + Comm-mgmt collectives (50% change)
  + P2P messages (sent) (64% change)
  + P2P bytes (sent) (51% change)
  + Process busy (2% change)
  + Process idle (both threads) (38% change)
  + Worker thread busy (2% change)
  + Trace events (5% change)

WHAT REGRESSED (in H2O128-4*8)
  - Wall time (50% change)
  - MPI time (of wall) (10% change)
  - Master thread busy (10% change)
  - Thread gap (17% change)

================================================================================================
```

MORE DETAILS ON THE PHASE SCRIPTS

timeline_phases.py detects repeated phases by following the ordered **call stack** of one process (all its threads). At every ENTER/LEAVE event it records the current stack signature — the ordered tuple of active regions, with measurement-noise frames stripped — and buckets contiguous time by the recurring signatures:

- signatures that recur ≥ `--min-reps` times form **phase classes** (each is a combination of functions simultaneously on the stack, e.g. an MD/SCF step).
- the timeline is segmented into **phases**, runs of slices dominated by one class, each with start/end/duration and event count.
- Phases whose duration is an outlier within their group are flagged.
- Every phase that isn't classified by recurring signatures is flagged as *other*

THE EXACT VS PREFIX METHOD

There is two way of ordering class of call stacks by recurring signatures:

- The ***exact*** method exact keeps every distinct call-stack tuple as its own class (default).
- The ***prefix*** method clusters signatures by shared common prefix so near-identical stack tails merge into one class. Each class's signature is the longest common prefix of its merged members.

The exact classes give a better, more actionable analysis on a smaller part of the traces while the prefix classes are more readable as a high-level overview but hide the information you would actually act on.

SOME MORE PHASES ALGORITHM

Once we have the output of *timeline_phase.py* we have some additionall algorithm that give us more informations about the phase analysis.

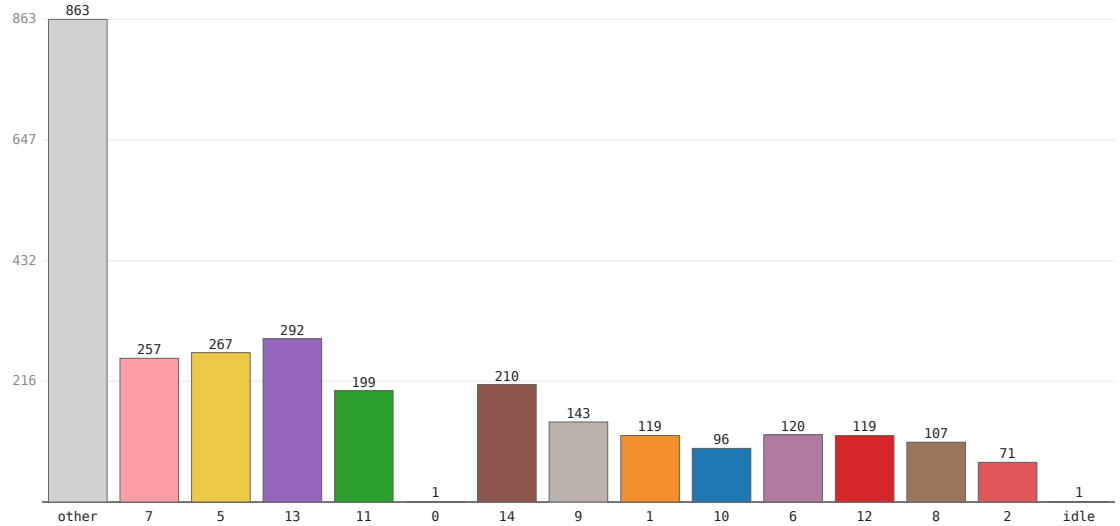
- *cut_outlier_traces.py* : Read the output of *timeline_phase.py* and create new cut otf2 trace files containing only phases that are flagged as outlier.
- *phase_transitions.py* : Builds the transition matrix between consecutive phase classes (how often a phase of class A is immediately followed by a phase of class B).
- *phase_comm_split.py* : For each phase of the trace, give the compute/MPI-wait split.
- *phase_rank_skew.py* : Per phase lag-analysis.

SOME MORE PHASES ALGORITHM

- *phase_inspect.py* : Given a rank and it's phase number prints everything that helps understand that specific phase in context.
- *phase_cadence.py* ` : Per-class duration/wait trend over repeats. Catches systematic drift that single-outlier analysis misses.
- *phase_source_map.py* : Map each phases/classes to the source code it is calling in it's call stack.
- And multiple scripts that gives us an svg visual representation of the output of the other scripts. (Examples below)

rank 0 — number of phases per class

unit: phases



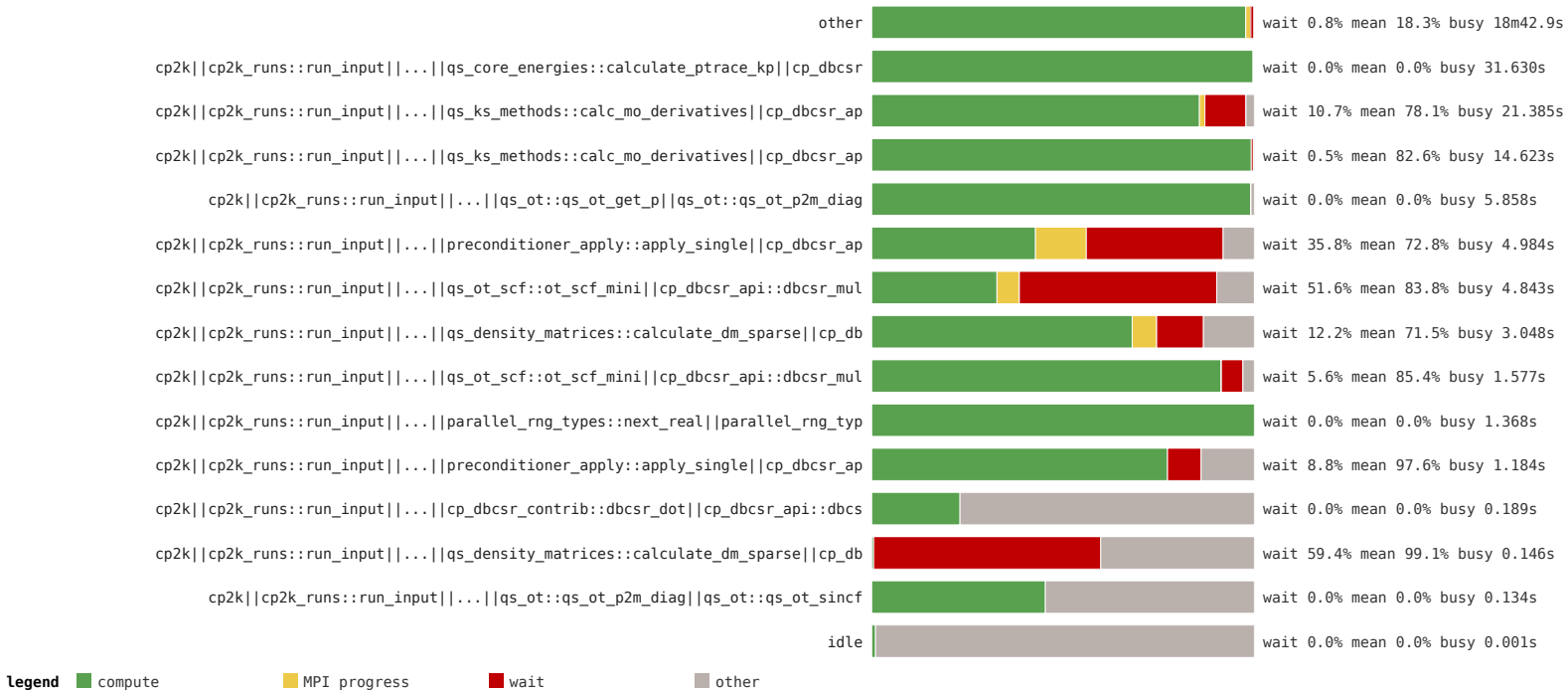
legend (class id — signature)

- 7: cp2k||cp2k_runs::run_input||...||qs_ks_methods::calc_mo_derivatives||cp_dbcsr_api::dbcsr_multiply
- 5: cp2k||cp2k_runs::run_input||...||qs_ot_scf::ot_scf_mini||cp_dbcsr_api::dbcsr_multiply
- 13: cp2k||cp2k_runs::run_input||...||preconditioner_apply::apply_single||cp_dbcsr_api::dbcsr_multiply
- 11: cp2k||cp2k_runs::run_input||...||qs_density_matrices::calculate_dm_sparse||cp_dbcsr_api::dbcsr_multiply
- 0: cp2k||cp2k_runs::run_input||...||parallel_rng_types::next_real||parallel_rng_types::rn53
- 14: cp2k||cp2k_runs::run_input||...||preconditioner_apply::apply_single||cp_dbcsr_api::dbcsr_multiply||MPI_Testany
- 9: cp2k||cp2k_runs::run_input||...||qs_core_energies::calculate_ptrace_kp||cp_dbcsr_contrib::dbcsr_dot
- 1: cp2k||cp2k_runs::run_input||...||qs_ot::qs_ot_get_p||qs_ot::qs_ot_p2m_diag
- 10: cp2k||cp2k_runs::run_input||...||cp_dbcsr_contrib::dbcsr_dot||cp_dbcsr_api::dbcsr_get_readonly_block_p
- 6: cp2k||cp2k_runs::run_input||...||qs_ot_scf::ot_scf_mini||cp_dbcsr_api::dbcsr_multiply||MPI_Testany
- 12: cp2k||cp2k_runs::run_input||...||qs_density_matrices::calculate_dm_sparse||cp_dbcsr_api::dbcsr_multiply||MPI_Testany
- 8: cp2k||cp2k_runs::run_input||...||qs_ks_methods::calc_mo_derivatives||cp_dbcsr_api::dbcsr_multiply||MPI_Testany
- 2: cp2k||cp2k_runs::run_input||...||qs_ot::qs_ot_p2m_diag||qs_ot::qs_ot_sincf

CASUS
CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

rank 0 — per-class wait/compute mix (top 15, weighted)

each bar = 100% of the class busy time; sorted by busy (weighted wait%); unit: seconds



GIVING THE DATA TO THE A.I AGENT FOR ANALYSIS

Once we have generated an output with all of the previous scripts, we can give the data to the A.I agent (here the model is *DeepSeek-V4-Flash*) so that it can give us recommendation on how we can improve the CP2K run.

Here I have made a run of H2O-128.inp with 8 *MPI* ranks and 4 *OPENMP* threads. Once I have run all of the scripts on the resulting trace and asked the A.I agent for improvements here are the results:

RUN DETAILS

- H2O-128.inp: 384 atoms, *RUN_TYPE MD*, ***STEPS 10***, NVE, MGRID 280/30, TZVP2-GTH, Pade, OT/DIIS, SCF_GUESS ATOMIC, EXTRAPOLATION_ORDER 3, WF_INTERPOLATION PS.
- 8 ranks × 4 threads (32 locations), wall **155.78 s**, **820.6 M events / 10.6 GB**.
- balance: busy 99.14 %, idle 0.85 %, load balance 100 %, **master busy 44 % vs orphan 99 % → 55 pt gap**.
- comm / hotspots: **MPI = 56 % of wall (697.8 s agg / 87.2 s per rank)**, 52 % of it wait-like; compute 23 349 s agg.

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

WHERE THE TIME GOES (EXCLUSIVE, PER RANK)

- **MPI_Waitall 30.95 s/rank** (684 704 calls = 85 588/rank) — single largest exclusive cost.
- **cp_dbcsr_api::dbcsr_multiply 16.79 s/rank** excl (67.79 incl), 16 440 calls.
- **MPI_Alltoallv 15.59 s/rank** (4729 calls, avg 3.29 ms).
- **collocate_one_grid_level 13.98 + integrate_one_grid_level 12.96 = 26.9 s/rank** (grid).
- MPI_Testany 7.42, MPI_Allreduce 6.05, MPI_Isend 4.75, MPI_Alltoall 4.46, MPI_Allgather 4.13.
- cp_fm_cholesky_invert 4.47 incl / 1.81 excl + decompose 4.04 incl; core_ppl::build_core_ppl 1.80; dbcsr_complete_redistribute 8.39 incl.

COMMUNICATION

- **ALLTOALLV 124.7 s** (3.29 ms avg) + ALLTOALL 35.6 s = **FFT redistribution = ~35 % of MPI time**.
- ALLREDUCE 593 384 calls / 48.3 s (tiny 0.081 ms each — many sync points, `qs_ot`/DBCSR).
- 431 GB sent, p2p matrix 100 % dense and balanced; mpi wait varies 36.7–51.0 s/rank while wall is flat.
- `report` ranks: comm-bound (56 % MPI), wait-dominated (52 %), 3D-FFT redistribution dominates, thread gap 55 pt.

WHERE THE RUN COULD BE IMPROVED:

1. **MPI-bound at 56 %, two concrete drivers:**
   - **3D-FFT ALLTOALLV (124.7 s) + ALLTOALL (35.6 s)** → tune the 2D decomposition / reduce rank count / FFT backend. This is ~10 % of wall per rank and the single biggest MPI contributor.
   - **MPI_Waitall 31 s/rank over 85 588 calls** → sync/overlap; reduce collectives/sync points or overlap with compute.
2. **Thread imbalance is structural:** 3 workers at 99 % busy vs master 44 %. Adding threads won't help (workers saturated); **fewer ranks × more threads (e.g. 4×8)** shifts work to the master but increases its MPI — must be tested (the existing `H20128-8*4__vs__H20128-4*8` compare showed 4×8 regressed wall by 50 %).
3. **Startup is 34 % of a 10-step run:** 52.9 s, of which the first SCF ≈44.4 s because `SCF_GUESS ATOMIC` starts from scratch. `SCF_GUESS RESTART` (currently commented out) or a better initial guess would remove most of this for multi-step MD.

4. **Grid kernels: 26.9 s/rank (collocate+integrate)** with CUTOFF 280 / REL_CUTOFF 30 — worth a cutoff / `REL_CUTOFF` test.
5. **DBCSR path:** `dbcsr_multiply` 16.8 s/rank excl + `dbcsr_complete_redistribute` 8.4 s/rank incl (COSMA/stack tuning).
6. **Trace overhead:** 820 M events / 10.6 GB for 10 steps — use Score-P filtering for production.

And for applying these changes, the agent proposed to modify multiples *cp2k/src* files.