Mitglied der Helmholtz-Gemeinschaft

JÜLICH FORSCHUNGSZENTRUM

Scalasca components with reuse potential

July 21th 2008

Bernd Mohr, Felix Wolf

Outline

- Overview of Scalasca
- Components with reuse potential
  - OPARI OpenMP source-code instrumenter
  - MPI tracing wrappers and wrapper generator
  - Compiler event adapters
  - Library for efficient parallel file I/O
  - Profile browser
- Ongoing and planned collaborations

scalasca

- Started in January 2006
- Scalable performance-analysis toolset for parallel codes
  - Emphasis on detection of wait states
- Designed for large-scale systems such as IBM Blue Gene or Cray XT
- Funded through Helmholtz Impulse and Networking Funds
- Developed in cooperation with the University of Tennessee
- http://www.scalasca.org/

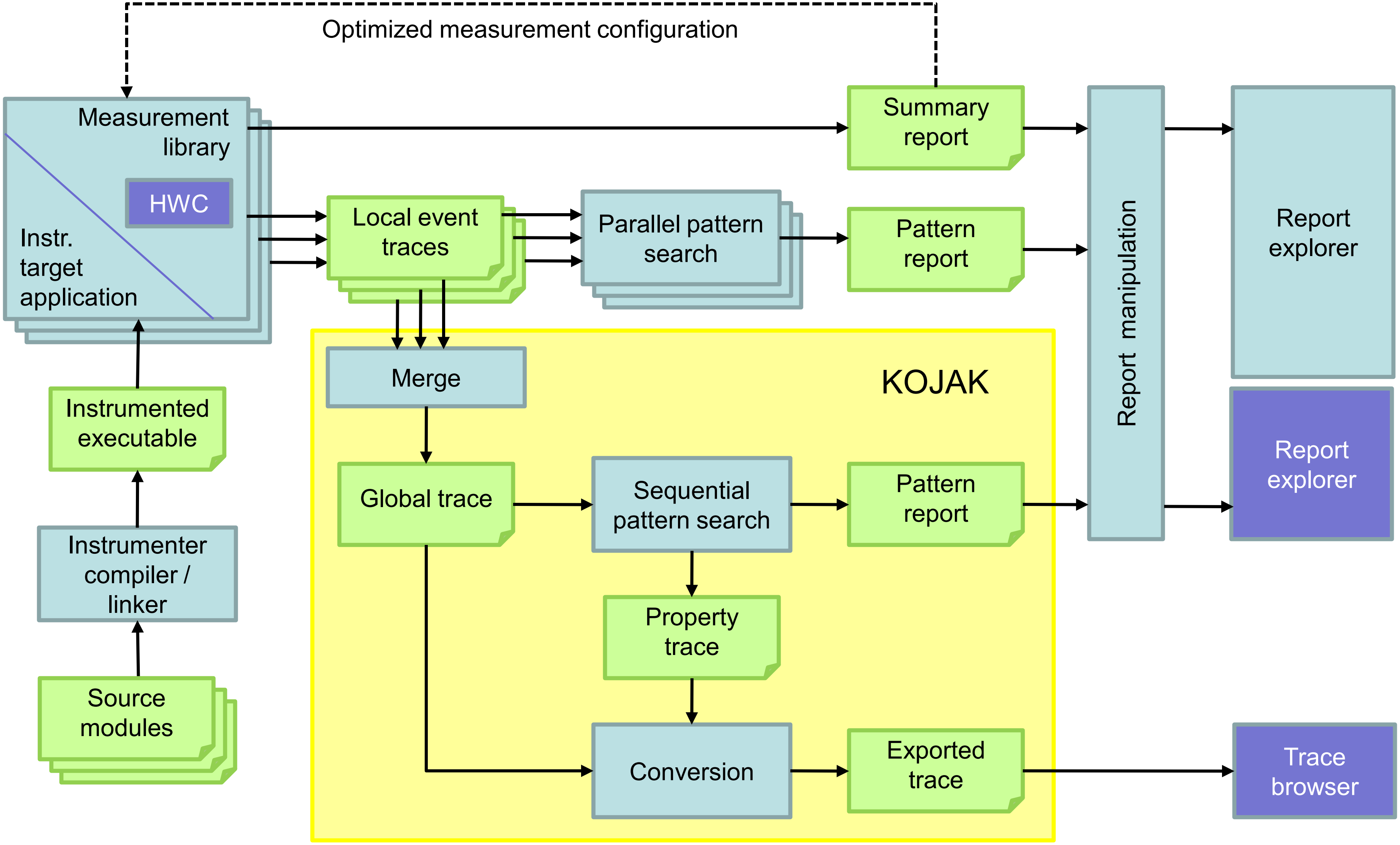
HELMHOLTZ | GEMEINSCHAFT

THE UNIVERSITY of TENNESSEE UT

Functionality

- Integrated performance analysis procedure
  - Runtime summaries (i.e., profiles)
    - *Overview of performance behavior*
    - *Refinement of instrumentation*
  - In-depth study of application behavior via event traces
    - *Localization and quantification of wait states*
  - Switching between both options without recompilation or re-linking
- Programming models supported
  - MPI-1
  - MPI-2 + other one-sided models (in progress)
  - OpenMP (in progress)

Performance data flow



🟪 = Third-party component

OPARI OpenMP source-code instrumenter

- Instruments Fortran, C, C++ OpenMP 2.5 codes with POMP instrumentation calls
- Used by KOJAK, Scalasca, TAU, VampirTrace, ompP
- Not perfect, but works for us

- Ongoing work
  - Removal of limitations
    - *Nested and dynamic threading*
    - *Inter-compilation units dependencies*
  - Support for OpenMP 3.0 features

MPI tracing wrappers and wrapper generator

- Complete MPI-2 tracing wrappers
  - Enter, Exit, Send, Recv, Collective, Get, Put events
  - C/C++ and Fortran support
- Basis also for Vampirtrace
- Very flexible wrapper generator
- Testsuite

Compiler event adapters

- Many compilers have (sometimes unsupported and undocumented) options for user function instrumentation
  - GNU, Intel, PGI, Pathscale, IBM XL, Sun f90, NEC, Hitachi
- Used by KOJAK, Scalasca, Vampirtrace

- Compiler event adapter component
  - Translates compiler specific events to generic enter/exit
  - Function filtering at run-time

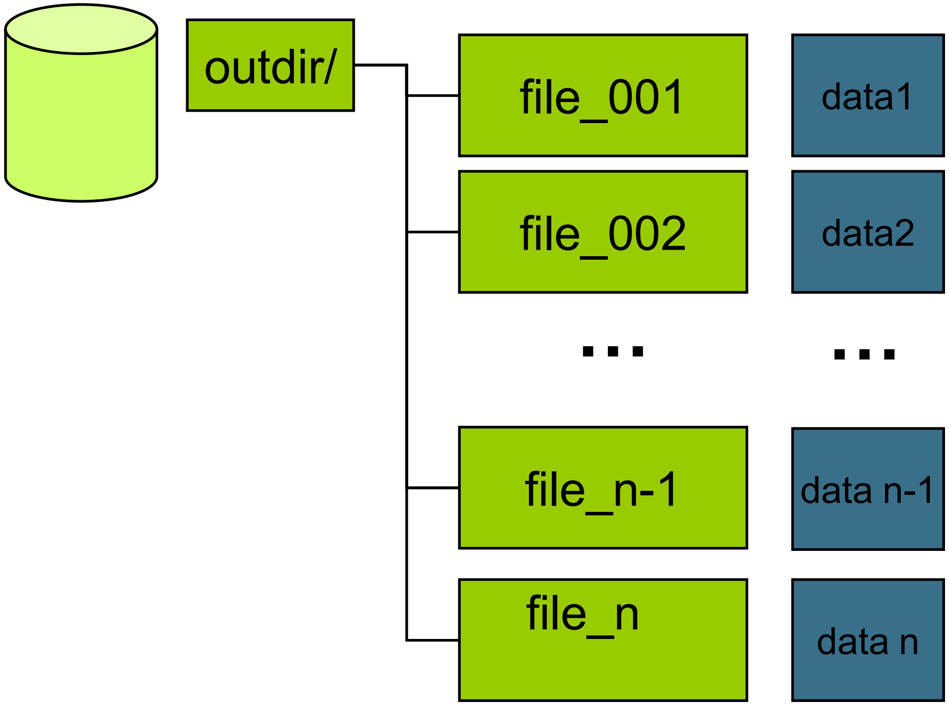
- Planned
  - Function filtering at compile time (GNU, IBM XL)

Efficient parallel I/O with sionlib

- **S**calable **I/O** library for **n**ative parallel file access
- Efficiently reading and writing binary files from thousands of processes, e.g.,
  - Process-local scratch/restart files
  - Process-local trace files (Scalasca)
- Simplified file handling
  - Only one large file instead of thousands of small files
- Optimized I/O
  - Alignment to file system blocks
- Minimal source code changes
  - Allows use of standard file pointer (FILE* fp)

Typical use case: parallel I/O to separate files

```
MPI_Init() /*  n tasks */
   …
   fileptr=fopen(file_###)
   …
   fwrite(buffer,fileptr)
   …
   fclose(fileptr)
   …
MPI_Finalize()
```

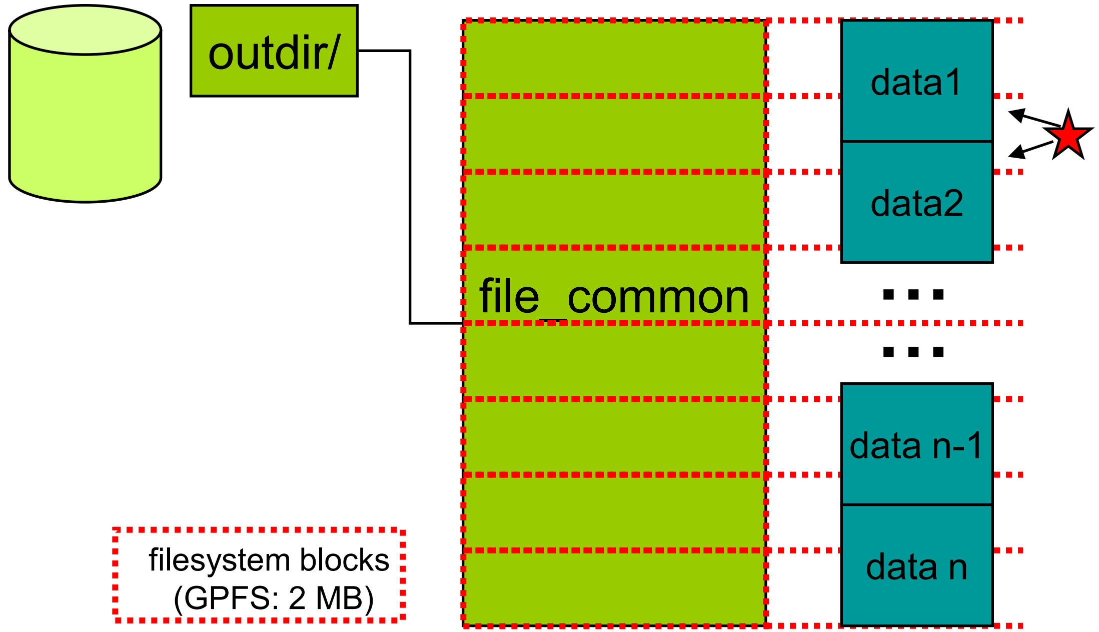


Problem 1: file handling (backup, HSM) ← number of files

Problem 2: slow create & open of files ← Lock on outdir (serialization)

Example: native parallel direct access

```
MPI_Init() /*  n tasks */
   …
   fileptr=fopen(file_common)
   …
   fseek(mypos)
   fwrite(buffer,fileptr)
   …
   fclose(fileptr)
   …
MPI_Finalize()
```



Initial Problem solved: fast open, only one file

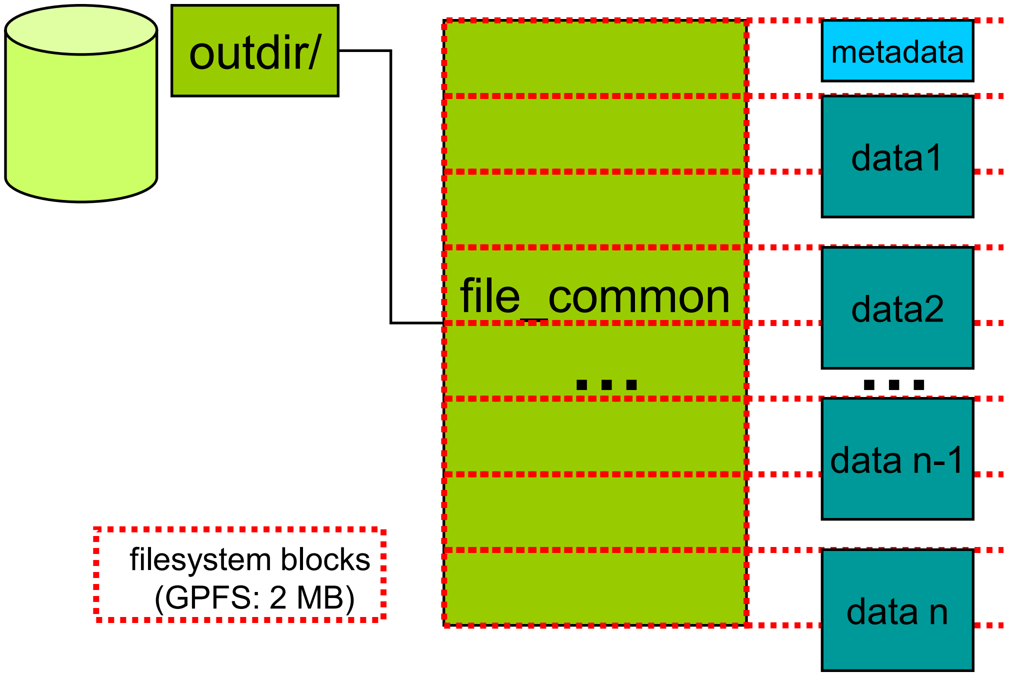
New Problem 1: meta data handling, start positions and length not stored

New Problem 2: file system locks on blocks, overlapping parallel access to blocks ★

Restriction: space required by each process must be known in advance

Access with sionlib

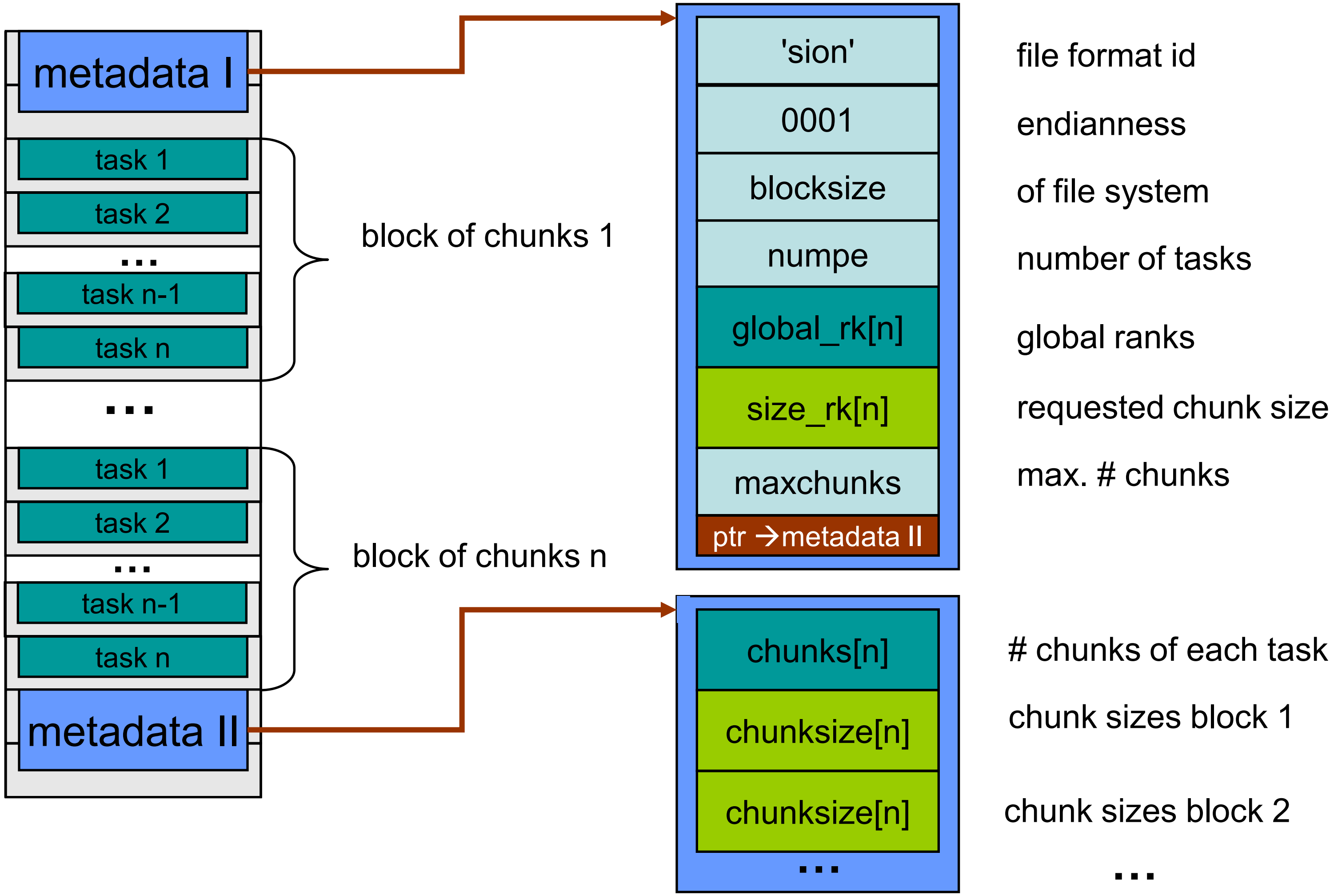
```
MPI_Init() /*  n tasks */
   …
 sid=sion_paropen_mpi(fname,
       localsize, fsblocksize,…, &fileptr)
 …
 sion_ensure_free_space(sid, nbytes)
 fwrite(buffer,fileptr)
 …
 sion_ensure_free_space(sid, nbytes)
 fwrite(buffer,fileptr)
 …
 sion_parclose(sid)
 …
MPI_Finalize()
```



Problems solved: simple file handling, fast open and fast I/O (fs block alignment)

Restriction: space required by each process must be known in advance
→ new allocation at the end of the file if writing more data than initially allocated

sionlib: internal file format



sionlib: comand line tools

```
siondump [-a] <sionfile>
```

- prints on stdout all information from the first meta data block , with -a also all chunk sizes from the second meta data block

```
sionsplit [-d digits] <sionfile>  <prefix>
```

- extracts task related files from a sion file
- a file will be generated for each task with a filename starting with <prefix>
- the task number will be appended to the <prefix>

```
siondefrag [-q blksize] [-s chunksize] <sionfile> <new_sionfile>
```

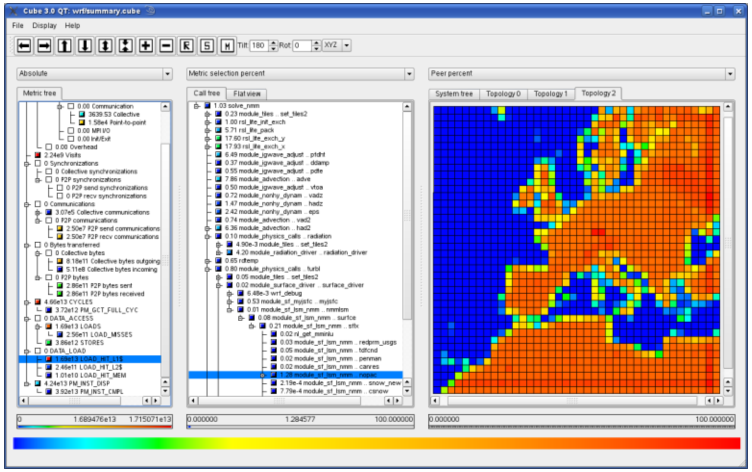
- generates a new sion file from an existing sion file
- the new file will have only one chunk per task which contains the data of all chunks of this task in the old sion file
- generates with “–q 1” a compact sion file without gaps

Measurement on 16 rack Blue Gene/P

- BG/P connected to file server with 128 x 10 GiE
  GPFS file system bandwidth: ~ 6GB/s
- Parallel test: (file server in production)
  - Writing and reading 2 TB data, 32 MB from each task
  - 65536 MPI-tasks, 128 I/O-nodes
  - Parallel open of one SION file → ~ 1s
  - Overall write bandwidth → 3.7 GB/s
    550s for writing 2 TB
  - Overall read bandwidth → 5.4 GB/s
    380s for reading 2 TB

CUBE - Call-path profile browser

- Browser based on tree widgets & topological display
- Data model and format to store call-path profiles
- Utilities to manipulate & analyze instances
  - Difference, mean, merge, cut, rank
- New version based on Qt
- Current applications
  - Scalasca trace analysis results & runtime summaries
  - TAU call-path profiles
  - MARMOT runtime errors



```
      0.00 Communication
        3639.53 Collective
        1.58e4 Point-to-point
      0.00 MPI I/O
      0.00 Init/Exit
    0.00 Overhead
  2.24e9 Visits
  0 Synchronizations
    0 Collective synchronizations
    0 P2P synchronizations
      0 P2P send synchronizations
      0 P2P recv synchronizations
  0 Communications
    3.07e5 Collective communications
    0 P2P communications
      2.50e7 P2P send communications
      2.50e7 P2P recv communications
  0 Bytes transferred
    0 Collective bytes
      8.18e11 Collective bytes outgoing
      5.11e8 Collective bytes incoming
    0 P2P bytes
      2.86e11 P2P bytes sent
      2.86e11 P2P bytes received
  4.66e13 CYCLES
    3.72e12 PM_GCT_FULL_CYC
  0 DATA_ACCESS
    1.69e13 LOADS
      2.56e11 LOAD_MISSES
    3.86e12 STORES
  0 DATA_LOAD
    1.69e13 LOAD_HIT_L1$
    2.46e11 LOAD_HIT_L2$
    1.01e10 LOAD_HIT_MEM
  4.24e13 PM_INST_DISP
    3.92e13 PM_INST_CMPL
```

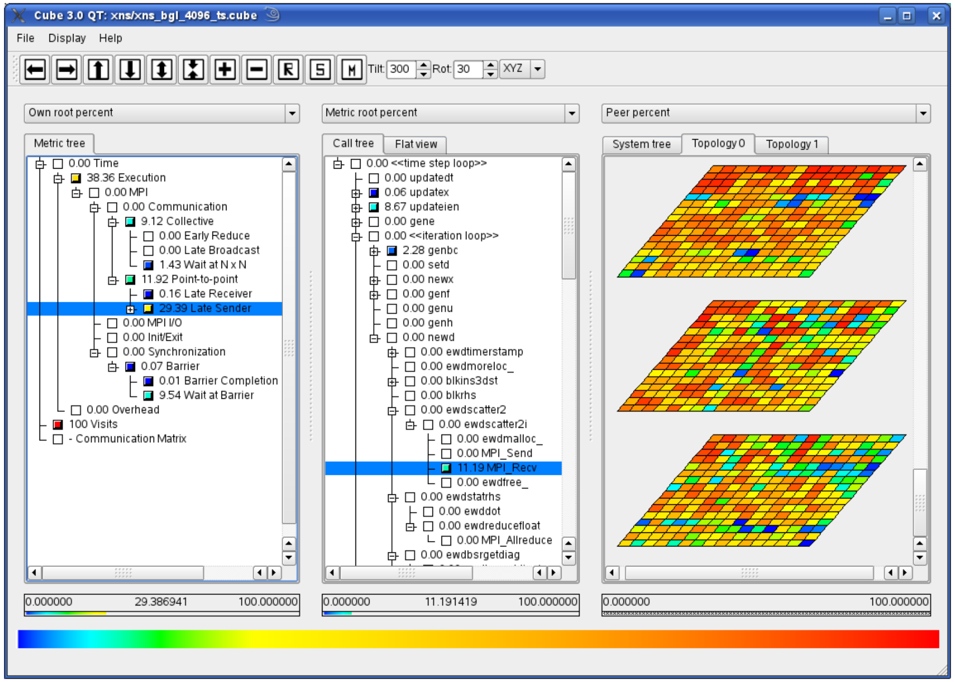
```
1.03 solve_nmm
  0.23 module_tiles .. set_tiles2
  1.00 rsl_lite_init_exch
  5.71 rsl_lite_pack
  17.60 rsl_lite_exch_y
  17.93 rsl_lite_exch_x
  6.49 module_igwave_adjust .. ptdht
  0.37 module_igwave_adjust .. ddamp
  0.55 module_igwave_adjust .. pdte
  7.86 module_advection .. adve
  0.50 module_igwave_adjust .. vtoa
  0.72 module_nonhy_dynam .. vadz
  1.47 module_nonhy_dynam .. hadz
  2.42 module_nonhy_dynam .. eps
  0.74 module_advection .. vad2
  6.36 module_advection .. had2
  0.10 module_physics_calls .. radiation
    4.90e-3 module_tiles .. set_tiles2
    4.20 module_radiation_driver .. radiation_driver
  0.65 rdtemp
  0.80 module_physics_calls .. turbl
    0.05 module_tiles .. set_tiles2
    0.02 module_surface_driver .. surface_driver
      6.48e-3 wrf_debug
      0.53 module_sf_myjsfc .. myjsfc
      0.01 module_sf_lsm_nmm .. nmmlsm
        0.08 module_sf_lsm_nmm .. surfce
          0.21 module_sf_lsm_nmm .. sflx
            0.02 nl_get_mminlu
            0.03 module_sf_lsm_nmm .. redprm_usgs
            0.05 module_sf_lsm_nmm .. tdfcnd
            0.02 module_sf_lsm_nmm .. penman
            0.02 module_sf_lsm_nmm .. canres
            1.28 module_sf_lsm_nmm .. nopac
            2.19e-4 module_sf_lsm_nmm .. snow_new
            7.79e-4 module_sf_lsm_nmm .. csnow
```

```
0    1.689476e13    1.715071e13
0.000000    1.284577    100.000000
0.000000    100.000000
```

June 7th 2008

Improvements of new version

- More configuration options
  - Order of trees
  - Color spectrum
  - Format and precision of numbers
  - Fonts
- Optimized to handle large data sets
  - Fast parser
  - No 3rd-party XML library
  - Dynamic loading of individual metrics
  - Faster aggregation algorithms
- More flexible and user-friendly topology widget
  - E.g., rotation of topology via mouse



Cube 3.0 QT: xns/xns_bgl_4096_ts.cube

```
0.00 Time
  38.36 Execution
    0.00 MPI
      0.00 Communication
        9.12 Collective
          0.00 Early Reduce
          0.00 Late Broadcast
          1.43 Wait at N x N
        11.92 Point-to-point
          0.16 Late Receiver
          29.39 Late Sender
      0.00 MPI I/O
      0.00 Init/Exit
      0.00 Synchronization
        0.07 Barrier
          0.01 Barrier Completion
          9.54 Wait at Barrier
  0.00 Overhead
100 Visits
- Communication Matrix
```

```
0.00 <<time step loop>>
  0.00 updatedt
  0.06 updatex
  8.67 updateien
  0.00 gene
  0.00 <<iteration loop>>
    2.28 genbc
    0.00 setd
    0.00 newx
    0.00 genf
    0.00 genu
    0.00 genh
    0.00 newd
      0.00 ewdtimerstamp
      0.00 ewdmoreloc_
      0.00 blkins3dst
      0.00 blkrhs
      0.00 ewdscatter2
        0.00 ewdscatter2i
          0.00 ewdmalloc_
          0.00 MPI_Send
          11.19 MPI_Recv
          0.00 ewdfree_
      0.00 ewdstatrhs
        0.00 ewddot
        0.00 ewdreducefloat
          0.00 MPI_Allreduce
      0.00 ewdbsrgetdiag
```

```
0.000000    29.386941    100.000000
0.000000    11.191419    100.000000
0.000000                 100.000000
```

Ongoing and planned collaborations

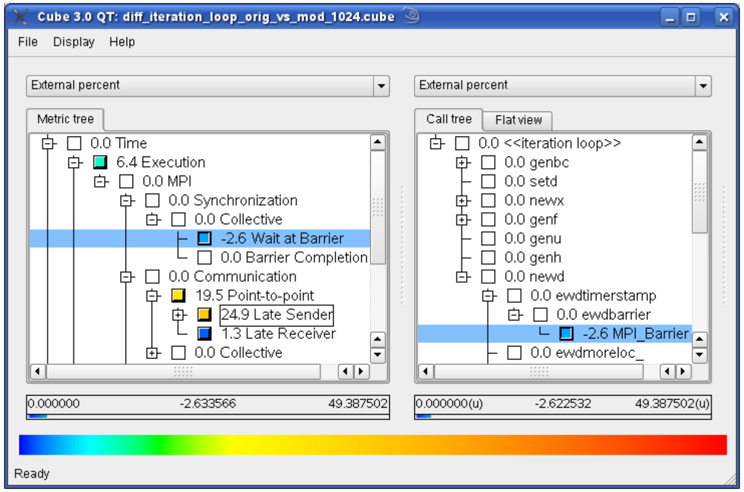
- Vampir & Scalasca
  - Unified parallel read interface for OTF & EPILOG traces
  - Unified tracing library (planned)
- TAU & Scalasca
  - Unified instrumentation facilities
  - Unified profiling runtime (planned)

Thank you!

scalasca

www.scalasca.org

Please download and try
Version 1.0



Cube 3.0 QT: diff_iteration_loop_orig_vs_mod_1024.cube

```
Metric tree                          Call tree
⊟ □ 0.0 Time                         ⊟ □ 0.0 <<iteration loop>>
  ⊟ ■ 6.4 Execution                    ⊞ □ 0.0 genbc
    ⊟ □ 0.0 MPI                          □ 0.0 setd
      ⊟ □ 0.0 Synchronization          ⊞ □ 0.0 newx
        ⊟ □ 0.0 Collective             ⊞ □ 0.0 genf
            ■ -2.6 Wait at Barrier       □ 0.0 genu
            □ 0.0 Barrier Completion     □ 0.0 genh
      ⊟ □ 0.0 Communication            ⊟ □ 0.0 newd
        ⊟ ■ 19.5 Point-to-point          ⊟ □ 0.0 ewdtimerstamp
          ⊞ ■ 24.9 Late Sender             ⊟ □ 0.0 ewdbarrier
            ■ 1.3 Late Receiver                ■ -2.6 MPI_Barrier
        ⊞ □ 0.0 Collective                 □ 0.0 ewdmoreloc_

0.000000   -2.633566   49.387502     0.000000(u)   -2.622532   49.387502(u)

Ready
```