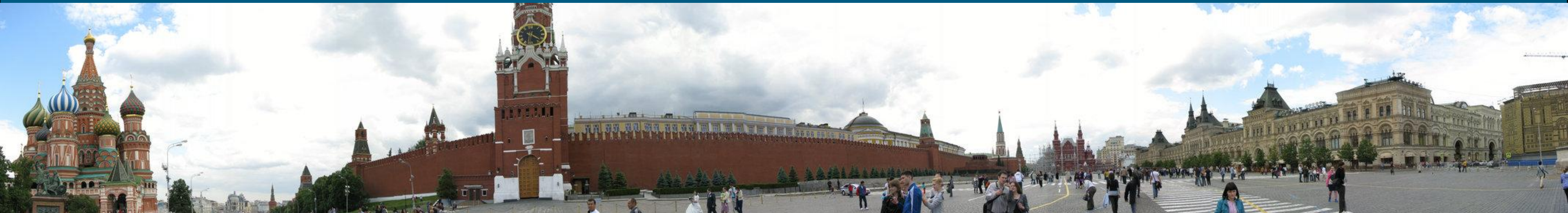
HOlistic Performance System Analysis

Bernd Mohr, Jülich Supercomputing Centre
Vladimir Voevodin, Moscow State University



Project Summary

Holistic Performance Analysis

=

integrated diagnostic infrastructure for combined

system-level performance analysis
+
application-level performance analysis

of parallel applications on an HPC system

Project Consortium

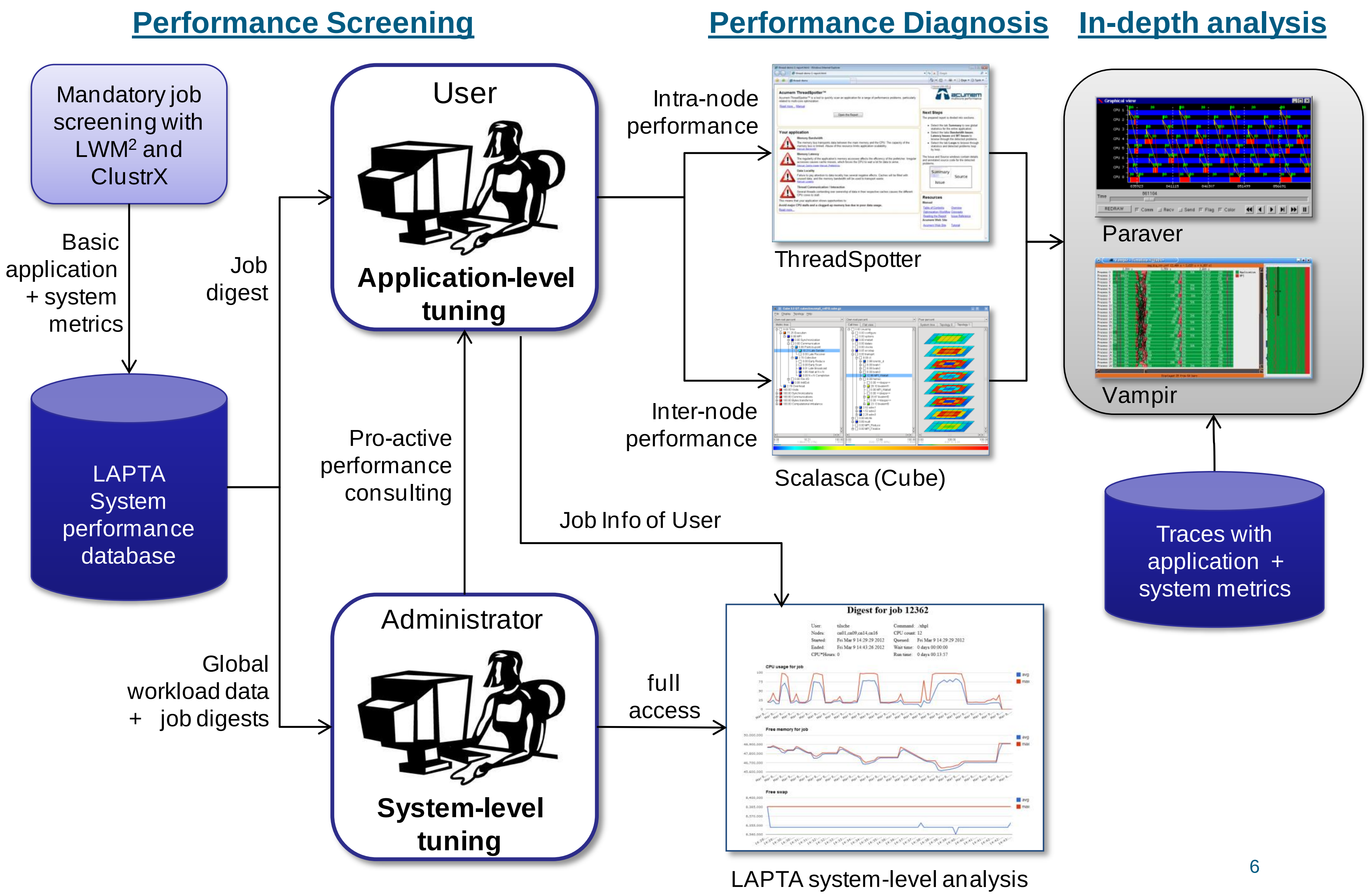
- **EU**
  - Forschungzentrum Jülich, JSC (EU Coordinator)
  - Barcelona Supercomputing Center
  - German Research School for Simulation Sciences
  - Rogue Wave Software AB
  - Technische Universität Dresden, ZIH
- **Russia**
  - Moscow State University, RCC (RU Coordinator)
  - T-Platforms
  - Russian Academy of Sciences,
    Joint Supercomputer Center
  - Southern Federal University, Taganrog

THE HOPSA WORKFLOW AND PERFORMANCE TOOLS

HOPSA Tool Set for Parallel Programs

- BSC
  - **Extrae**: instrumentation + measurement system for Paraver
  - **Paraver**: trace visualization and analysis tool
  - **Dimemas**: performance modeling and prediction tool
- RW
  - **ThreadSpotter** : memory and threading analysis tool
- TUD
  - **Vampir**: trace visualization and analysis tool
- GRS/JSC
  - **LWM^2:** light-weight measurement module
  - **Scalasca**: instrumentation, measurement + analysis tool set
  - **CUBE**: Scalasca result browser
- GRS/JSC/TUD
  - **Score-P**: Instrumentation and measurement system

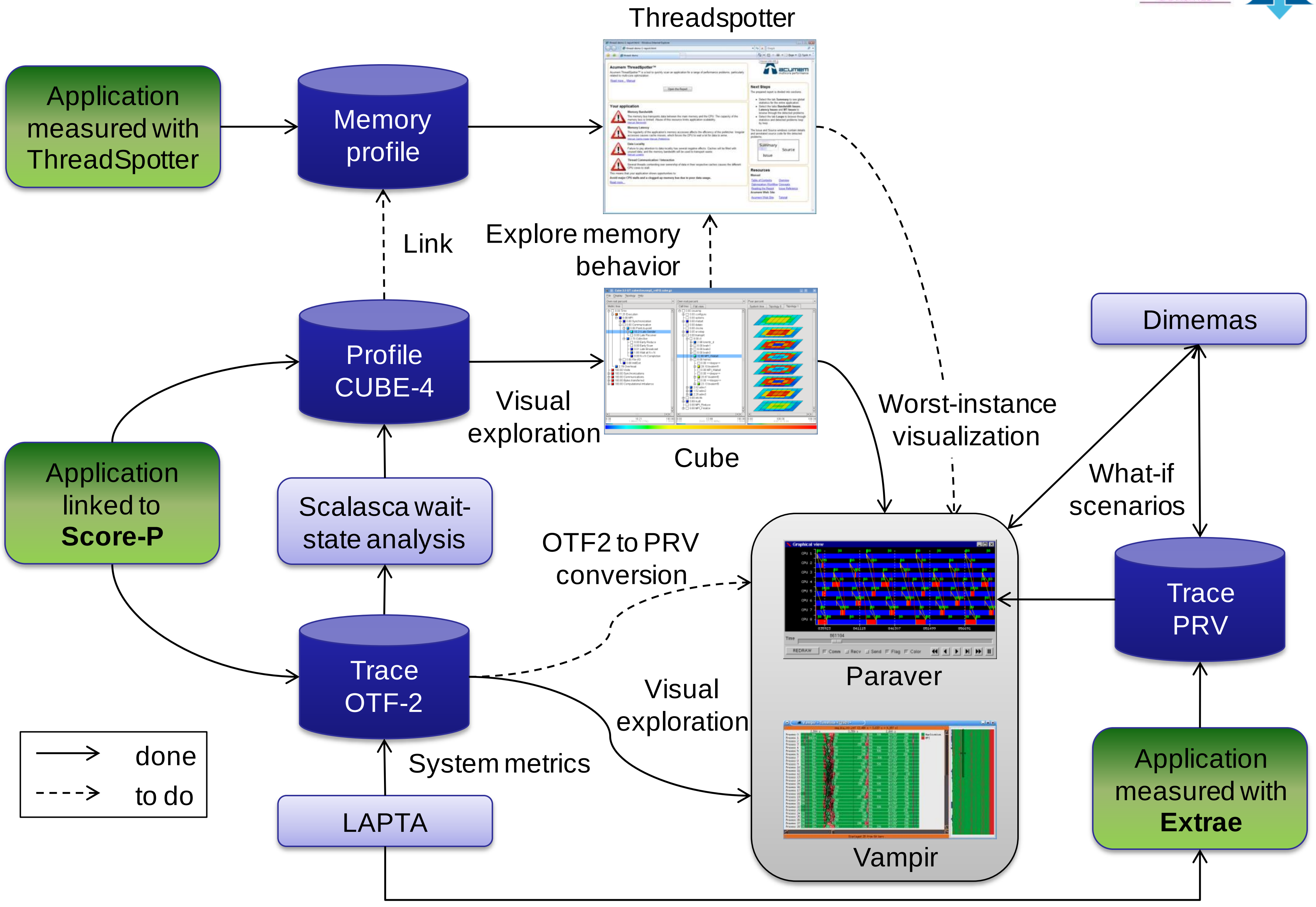
The HOPSA Performance Workflow



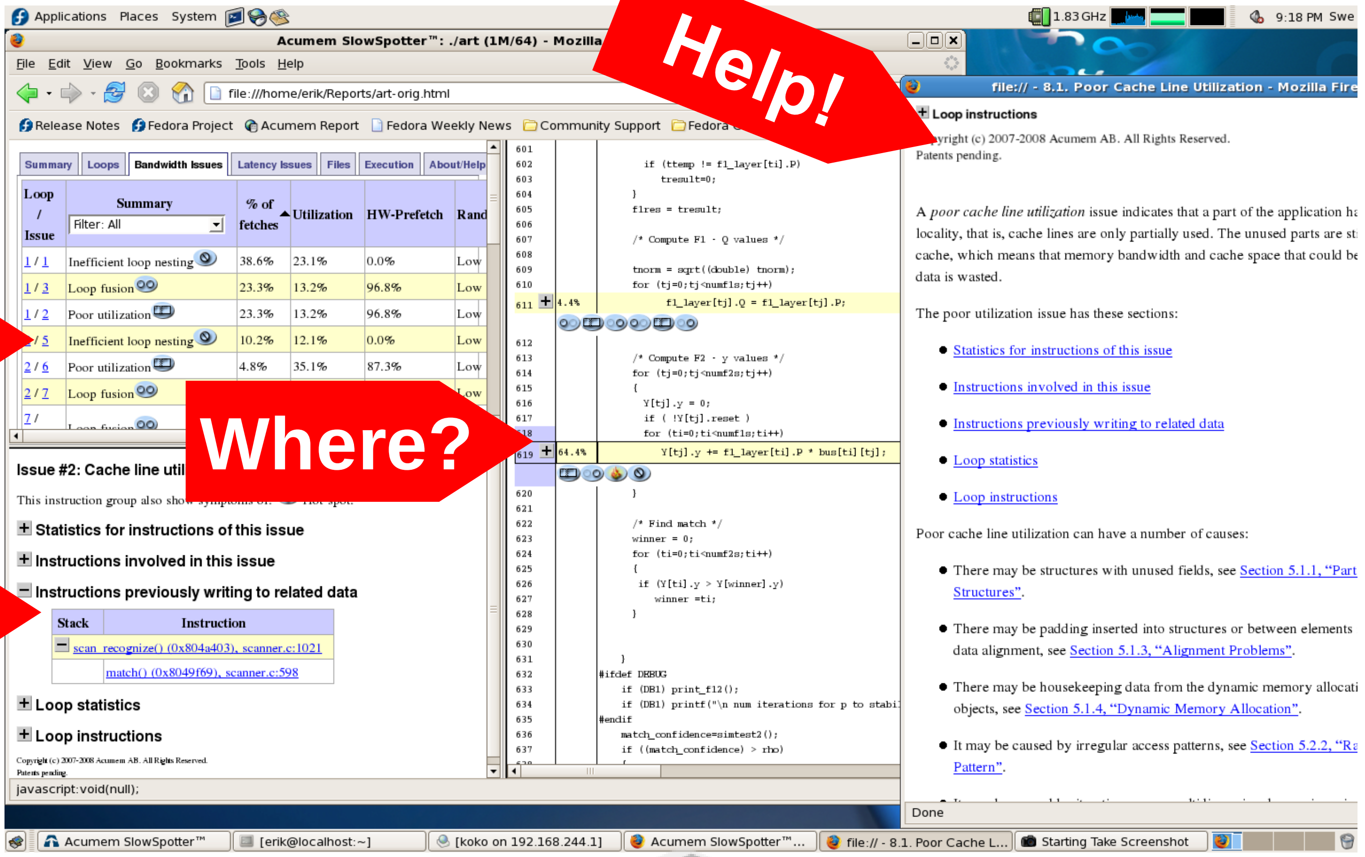
Digest for job 12362

User: tilsche
Nodes: cn01,cn09,cn14,cn16
Started: Fri Mar 9 14:29:29 2012
Ended: Fri Mar 9 14:43:26 2012
CPU*Hours: 0
Command: ./xhpl
CPU count: 12
Queued: Fri Mar 9 14:29:29 2012
Wait time: 0 days 00:00:00
Run time: 0 days 00:13:57

Interoperability between analysis tools



ThreadSpotter Memory + Threading Analysis


Applications Places System
1.83 GHz 9:18 PM Swe
Acumem SlowSpotter™: ./art (1M/64) - Mozilla
File Edit View Go Bookmarks Tools Help
file:///home/erik/Reports/art-orig.html
Release Notes Fedora Project Acumem Report Fedora Weekly News Community Support Fedora
Summary Loops Bandwidth Issues Latency Issues Files Execution About/Help
file:// - 8.1. Poor Cache Line Utilization - Mozilla Fire
Acumem SlowSpotter™ [erik@localhost:~] [koko on 192.168.244.1] Acumem SlowSpotter™... file:// - 8.1. Poor Cache L... Starting Take Screenshot
Done
What?
Where?
How?
Help!

Loop / Issue	Summary (Filter: All)	% of fetches	Utilization	HW-Prefetch	Rand
1 / 1	Inefficient loop nesting	38.6%	23.1%	0.0%	Low
1 / 3	Loop fusion	23.3%	13.2%	96.8%	Low
1 / 2	Poor utilization	23.3%	13.2%	96.8%	Low
/ 5	Inefficient loop nesting	10.2%	12.1%	0.0%	Low
2 / 6	Poor utilization	4.8%	35.1%	87.3%	Low
2 / 7	Loop fusion				Low
7 /	Loop fusion				

Issue #2: Cache line util

This instruction group also show

+ Statistics for instructions of this issue

+ Instructions involved in this issue

− Instructions previously writing to related data

Stack	Instruction
− scan_recognize() (0x804a403), scanner.c:1021	
	match() (0x8049f69), scanner.c:598

+ Loop statistics

+ Loop instructions

Copyright (c) 2007-2008 Acumem AB. All Rights Reserved.
Patents pending.

javascript:void(null);

```
                    if (ttemp != f1_layer[ti].P)
                       tresult=0;
                 }
                 f1res = tresult;

                 /* Compute F1 - Q values */

                 tnorm = sqrt((double) tnorm);
                 for (tj=0;tj<numf1s;tj++)
+ 4.4%                f1_layer[tj].Q = f1_layer[tj].P;


                 /* Compute F2 - y values */
                 for (tj=0;tj<numf2s;tj++)
                 {
                   Y[tj].y = 0;
                   if ( !Y[tj].reset )
                   for (ti=0;ti<numf1s;ti++)
+ 64.4%               Y[tj].y += f1_layer[ti].P * bus[ti][tj];

                 }

                 /* Find match */
                 winner = 0;
                 for (ti=0;ti<numf2s;ti++)
                 {
                  if (Y[ti].y > Y[winner].y)
                     winner =ti;
                 }


              }
#ifdef DEBUG
              if (DB1) print_f12();
              if (DB1) printf("\n num iterations for p to stabi
#endif
              match_confidence=simtest2();
              if ((match_confidence) > rho)
```

+ Loop instructions

yright (c) 2007-2008 Acumem AB. All Rights Reserved.
Patents pending.

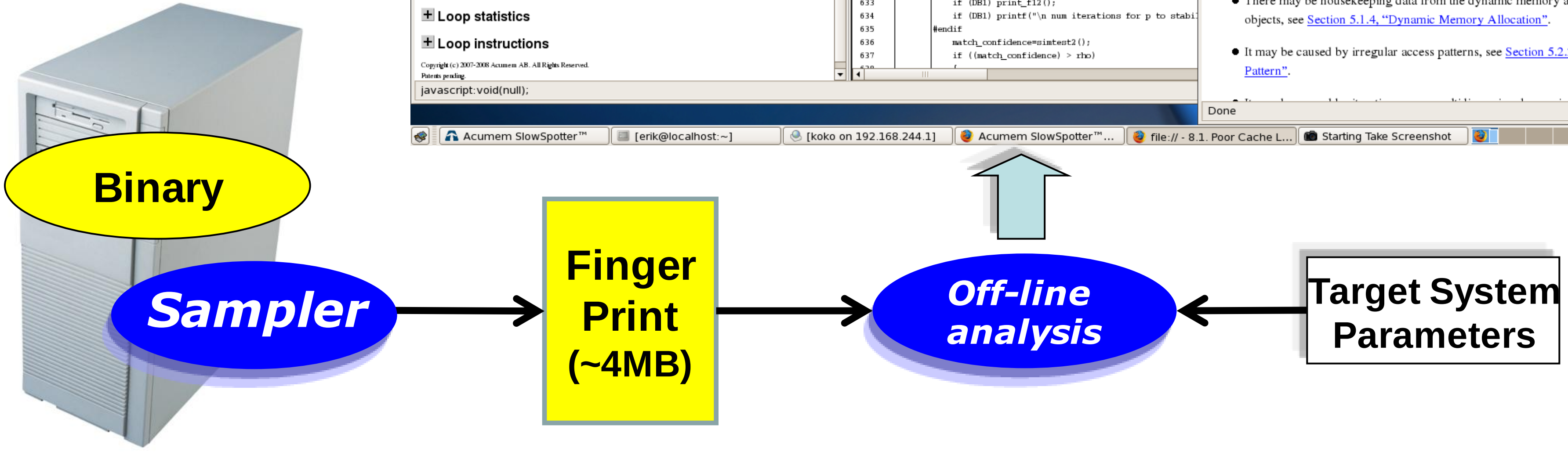
A *poor cache line utilization* issue indicates that a part of the application ha locality, that is, cache lines are only partially used. The unused parts are st cache, which means that memory bandwidth and cache space that could be data is wasted.

The poor utilization issue has these sections:

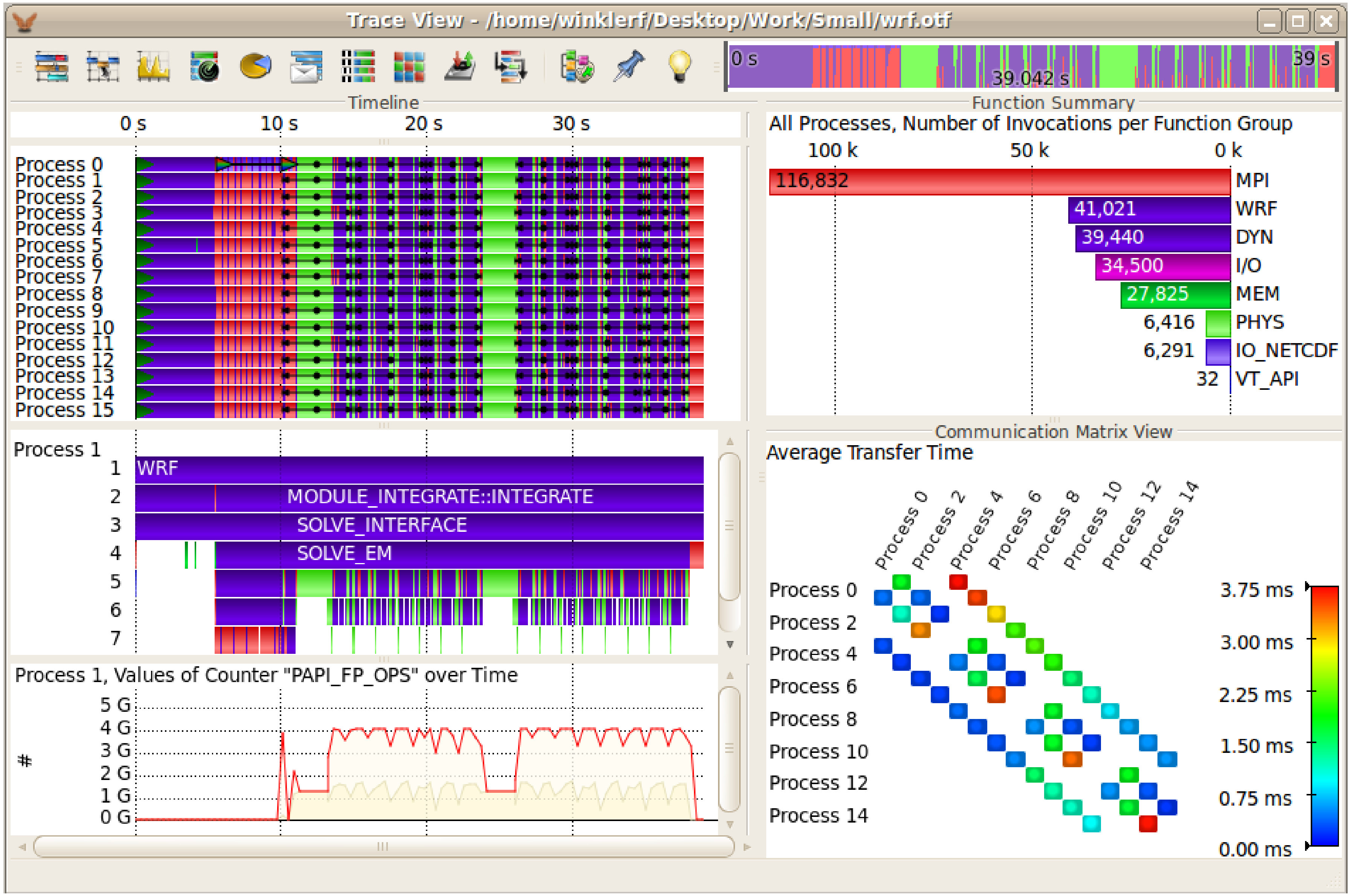
- Statistics for instructions of this issue
- Instructions involved in this issue
- Instructions previously writing to related data
- Loop statistics
- Loop instructions

Poor cache line utilization can have a number of causes:

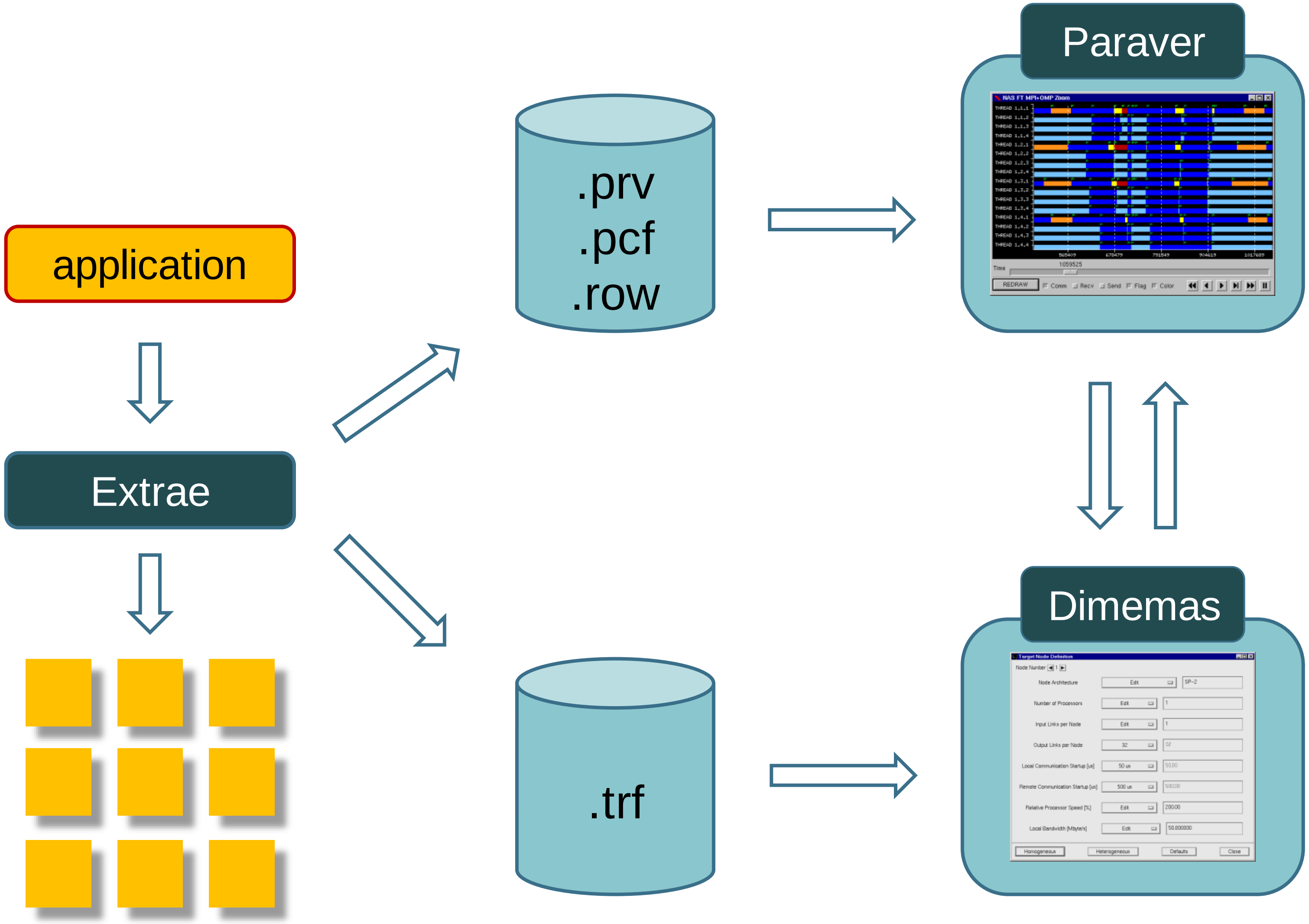
- There may be structures with unused fields, see Section 5.1.1, "Part Structures".
- There may be padding inserted into structures or between elements data alignment, see Section 5.1.3, "Alignment Problems".
- There may be housekeeping data from the dynamic memory allocati objects, see Section 5.1.4, "Dynamic Memory Allocation".
- It may be caused by irregular access patterns, see Section 5.2.2, "Ra Pattern".



Vampir Event-Trace Visualization



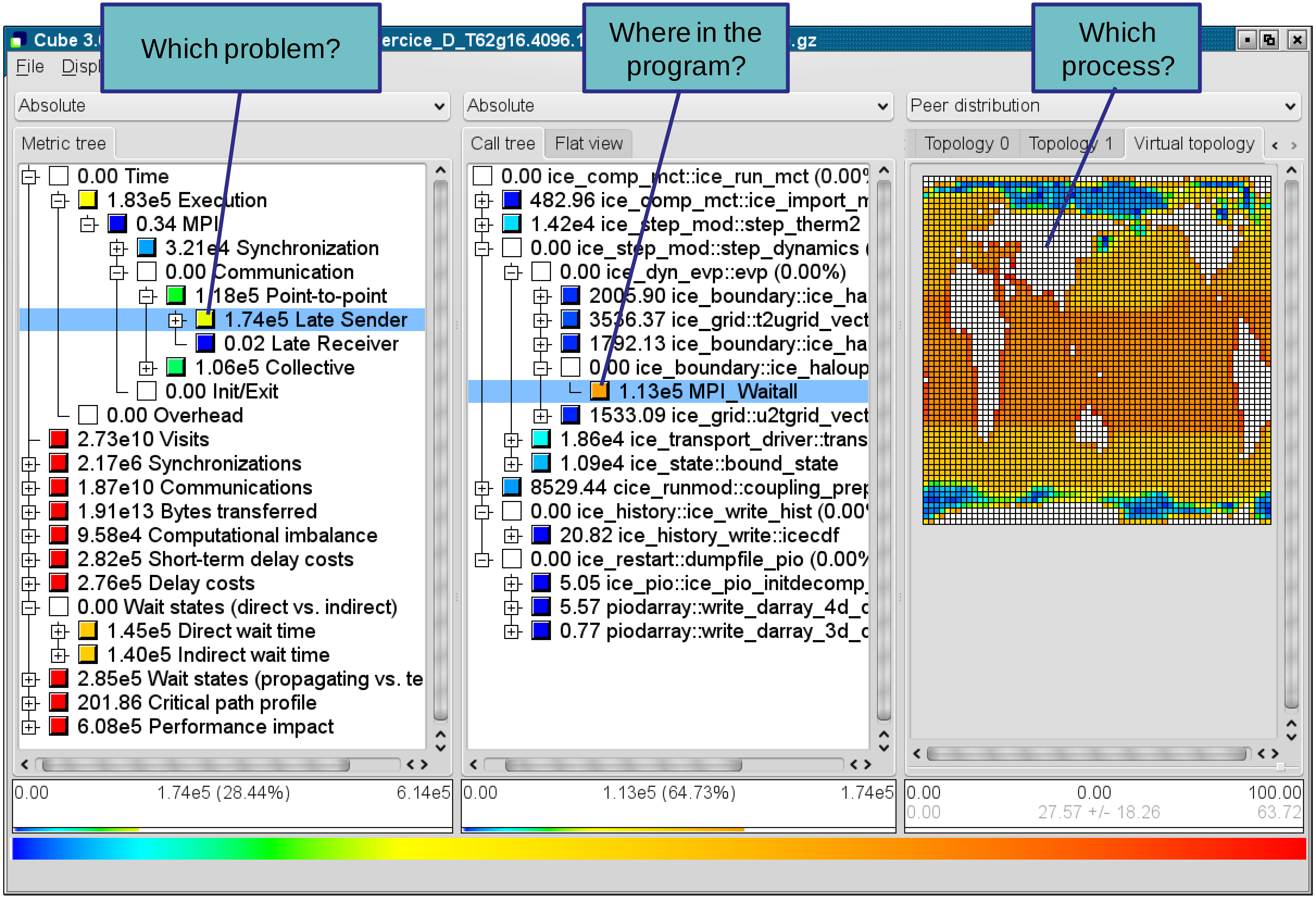
BSC Event Trace Performance Tools



Node Number 1

Node Architecture	Edit	SP–2
Number of Processors	Edit	1
Input Links per Node	Edit	1
Output Links per Node	32	32
Local Communication Startup [us]	50 us	50.00
Remote Communication Startup [us]	500 us	500.00
Relative Processor Speed [%]	Edit	200.00
Local Bandwidth [Mbyte/s]	Edit	50.000000

Scalasca Callpath Profiler + Trace Analyzer



```
0.00 Time
  1.83e5 Execution
    0.34 MPI
      3.21e4 Synchronization
      0.00 Communication
        1.18e5 Point-to-point
          1.74e5 Late Sender
          0.02 Late Receiver
        1.06e5 Collective
      0.00 Init/Exit
  0.00 Overhead
2.73e10 Visits
2.17e6 Synchronizations
1.87e10 Communications
1.91e13 Bytes transferred
9.58e4 Computational imbalance
2.82e5 Short-term delay costs
2.76e5 Delay costs
0.00 Wait states (direct vs. indirect)
  1.45e5 Direct wait time
  1.40e5 Indirect wait time
2.85e5 Wait states (propagating vs. te
201.86 Critical path profile
6.08e5 Performance impact
```

```
0.00 ice_comp_mct::ice_run_mct (0.00%
482.96 ice_comp_mct::ice_import_m
1.42e4 ice_step_mod::step_therm2
0.00 ice_step_mod::step_dynamics (
  0.00 ice_dyn_evp::evp (0.00%)
    2005.90 ice_boundary::ice_ha
    3536.37 ice_grid::t2ugrid_vect
    1792.13 ice_boundary::ice_ha
    0.00 ice_boundary::ice_haloup
      1.13e5 MPI_Waitall
    1533.09 ice_grid::u2tgrid_vect
  1.86e4 ice_transport_driver::trans
  1.09e4 ice_state::bound_state
8529.44 cice_runmod::coupling_prep
0.00 ice_history::ice_write_hist (0.00
  20.82 ice_history_write::icecdf
0.00 ice_restart::dumpfile_pio (0.00%
  5.05 ice_pio::ice_pio_initdecomp_
  5.57 piodarray::write_darray_4d_d
  0.77 piodarray::write_darray_3d_d
```

```
0.00        1.74e5 (28.44%)        6.14e5
0.00        1.13e5 (64.73%)        1.74e5
0.00             0.00              100.00
0.00        27.57 +/- 18.26         63.72
```

The “module” Command

- Software which allows to **easily manage** different versions
  - of a product (e.g., vampir 7.0 ⇔ vampir 8.0)
  - of the same product (e.g., 32-bit ⇔ 64bit)
  - of execution modes of a product (e.g., parallel ⇔ sequential)

  without the need to adapt setups or makefiles
  of the user of the product!!!

- Works by **dynamic** modification of a user's environment
  ⇨ Only applies to **calling** shell / window / session!

- Modules used for
  - UNITE standard tool enviroment

Most Important Module Commands

`module`

- `avail` # show all available products
- `list` # list loaded products

- `load` *`product(s)`* # setup access to product
- `unload` *`product(s)`* # release access

- `whatis` *`product(s)`* # print short description
- `help` *`product(s)`* # print longer description
- `show` *`product(s)`* # show what “settings” are
  # performed for product

UNITE

- **UNIform Integrated Tool Environment**
- Standardizes tool access and documentation
  - Currently in use at JSC, RWTH, ZIH
- Based on “module” command
  - Standardized tool and version identification
    - <tool>/<version>-<special>
    - <special>: optional indicator if tool is specific for a MPI library, compiler, or 32/64 bit mode
- **Tools only visible after**
  - **`module load UNITE`** **# once per session**
- Basic usage and pointer to tool documentation via
  - `module help <tool>`

Example

```
% module load UNITE
UNITE loaded
% module help scalasca
Module Specific Help for scalasca/1.2-parastation-intel:

Scalasca: Scalable Performance Analysis of Large-Scale
          Parallel Applications
Version 1.2 (for parastation, Intel Compiler)

Basic usage:
1. Instrument application with skin
2. Collect & analyze execution measurement with scan
3. Examine analysis results with square

For more information:
- See ${SCALASCA_ROOT}/doc/manuals/quickref.pdf
  or type "scalasca -h"
- http://www.scalasca.org
- mailto:scalasca@fz-juelich.de
```

Schedule

- **Tuesday, Nov 27**
  - Introduction to HOPSA performance workflow
  - Memory and Treading analysis with ThreadSpotter
  - Performance screening with LWM^2
  - Profile analysis with Score-P and CUBE
- **Wednesday, Nov 28**
  - Trace analysis with Score-P, Vampir, and Scalasca
  - Trace analysis with Extrae/Paraver
- **Thursday, Nov 29**
  - Trace analysis with Extrae/Paraver
  - Performance prediction with Dimemas
  - Use all tools on your code
- **Friday, Nov 30**
  - Use all tools on your code

Questions?