dpomp: A Performance Monitoring Interface for OpenMP

Luiz DeRose
IBM Research
ACTC
Yorktown Heights, NY
USA
laderose@us.ibm.com

Bernd Mohr
Forschungszentrum Jülich
ZAM / NIC
52425 Jülich
Germany
b.mohr@fz-juelich.de

Seetharami Seelam
IBM Research
ACTC
Yorktown Heights, NY
USA
srseelam@us.ibm.com

Thomas J. Watson Research Center
PO Box 218
Yorktown Heights, NY 10598

Thomas J. Watson Research Center
PO Box 218
Yorktown Heights, NY 10598

What is POMP?

- Proposed standard for a performance monitoring interface for OpenMP
  - **Portable cross-platform/cross-language API to simplify the design and implementation of OpenMP tools**
  - **Three groups of events**
    - **OpenMP constructs and directives/pragmas**
      - **Enter/Exit around each OpenMP construct**
        - **Begin/End around associated body**
      - **Special case for parallel loops:**
        - **ChunkBegin/End, IterBegin/End, or IterEvent instead of Begin/End**
      - **"Single" events for small constructs like atomic or flush**
    - **OpenMP API calls**
      - **Enter/Exit events around omp_set_*_lock() functions**
      - **"single" events for all API functions**
    - **User functions and regions**
  - **Allows application programmers to specify and control amount of instrumentation**

laderose@us.ibm.com, b.mohr@fz-juelich.de, srseelam@us.ibm.com

Example: Standard Instrumentation

```
 1:  int main() {
 2:     int id;
***     POMP_Init();
 3:
***     { POMP_handle_t pomp_hd1 = 0;
***       int32 pomp_tid = omp_get_thread_num();
***       POMP_Parallel_enter(&pomp_hd1, pomp_tid, -1, 1,
***           "49*type=pregion*file=demo.c*slines=4,4*elines=8,8**");
 4:     #pragma omp parallel private(id)
 5:     {
***       int32 pomp_tid = omp_get_thread_num();
***       POMP_Parallel_begin(pomp_hd1, pomp_tid);
 6:       id = omp_get_thread_num();
 7:       printf("hello from %d\n", id);
***       POMP_Parallel_end(pomp_hd1, pomp_tid);
 8:     }
***       POMP_Parallel_exit(pomp_hd1, pomp_tid);
***     }
***     POMP_Finalize();
 9:  }
```


laderose@us.ibm.com, b.mohr@fz-juelich.de, srseelam@us.ibm.com

Example: Optimized Instrumentation

```
 1:  int main() {
 2:     int id;
***     POMP_handle_t pomp_hd1 = 0;
***     POMP_Init();
***     POMP_Get_handle(&pomp_hd1,
***          "49*type=pregion*file=demo.c*slines=4,4*elines=8,8**");
 3:
***     { int32 pomp_tid = omp_get_thread_num();
***       POMP_Parallel_enter(&pomp_hd1, pomp_tid, -1, 1, NULL);
 4:     #pragma omp parallel private(id)
 5:     {
***       int32 pomp_tid = omp_get_thread_num();
***       POMP_Parallel_begin(pomp_hd1, pomp_tid);
 6:       id = omp_get_thread_num();
 7:       printf("hello from %d\n", id);
***       POMP_Parallel_end(pomp_hd1, pomp_tid);
 8:     }
***       POMP_Parallel_exit(pomp_hd1, pomp_tid);
***     }
***     POMP_Finalize();
 9:  }
```


laderose@us.ibm.com, b.mohr@fz-juelich.de, srseelam@us.ibm.com

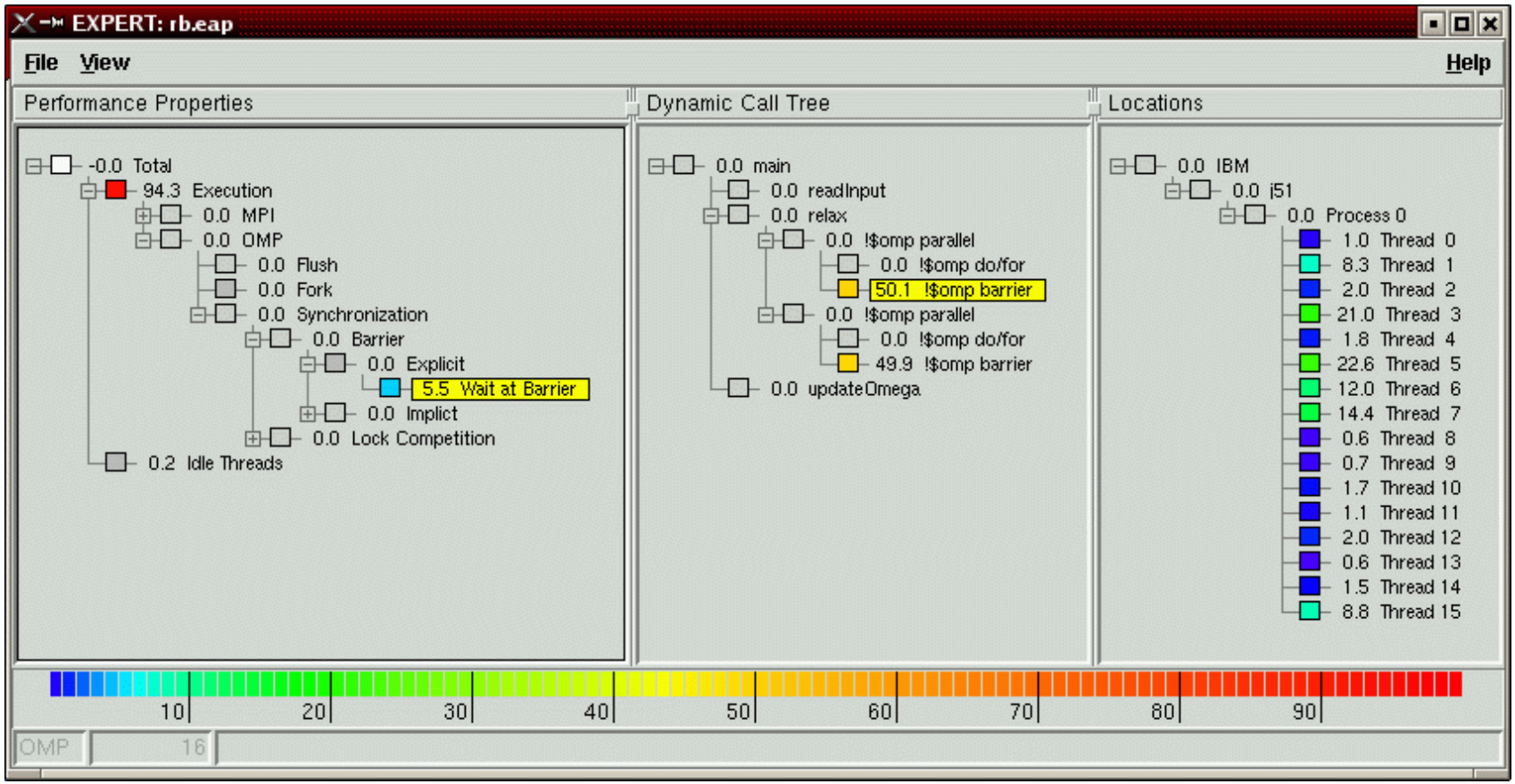
Using dpomp on JUMP

- module load dpomp
- To generate a function list
  - **llrun -D -s a.out**
  - **llrun -D "-l <function.lst> " a.out**
  - **dpomp [-s | -l <function.lst>] a.out –llfile <loadleveler file>**
- Running applications with dpomp
  - **llrun –D <pomp lib> -t <#threads> a.out**
  - **llrun -D "-f <function.lst> <pomp lib>" -t <#threads> a.out**
  - **dpomp <pomp lib> a.out -llfile <loadleveler file>**
  - **dpomp -f <function.lst> <pomp lib> a.out -llfile <ll file>**
- POMP Libraries (probes):
  - **test_probe**
  - **elg_probe (to generate epilog trace files: *.elg)**
  - **pomprof_probe (to generate OpenMP profile files: *.viz)**

laderose@us.ibm.com, b.mohr@fz-juelich.de, srseelam@us.ibm.com

Visualizing Epilog Traces with KOJAK

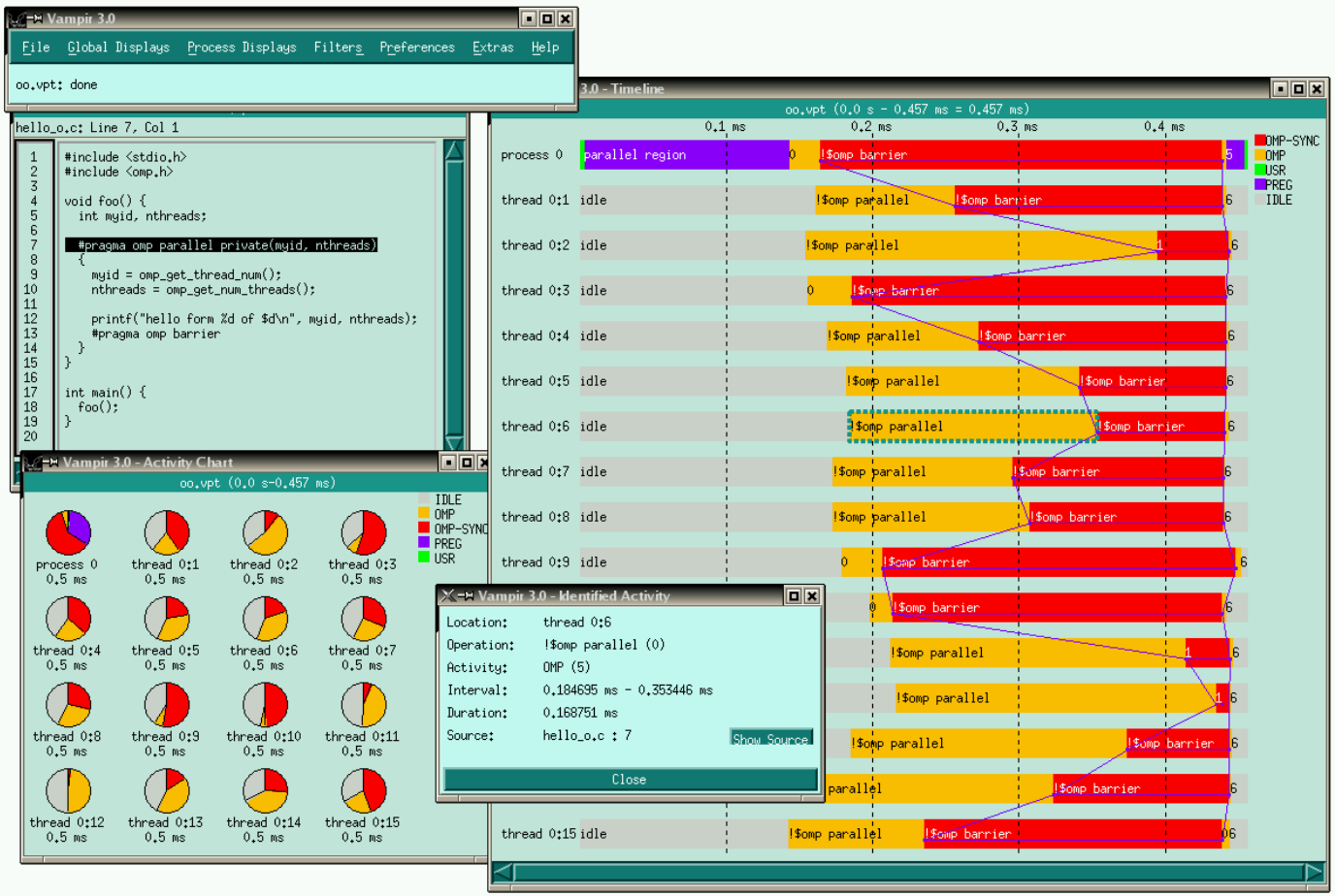
% cl-analyzer a.elg
% Presenter a.eap



laderose@us.ibm.com, b.mohr@fz-juelich.de, srseelam@us.ibm.com

Converting Epilog Trace to VTF3 for VAMPIR

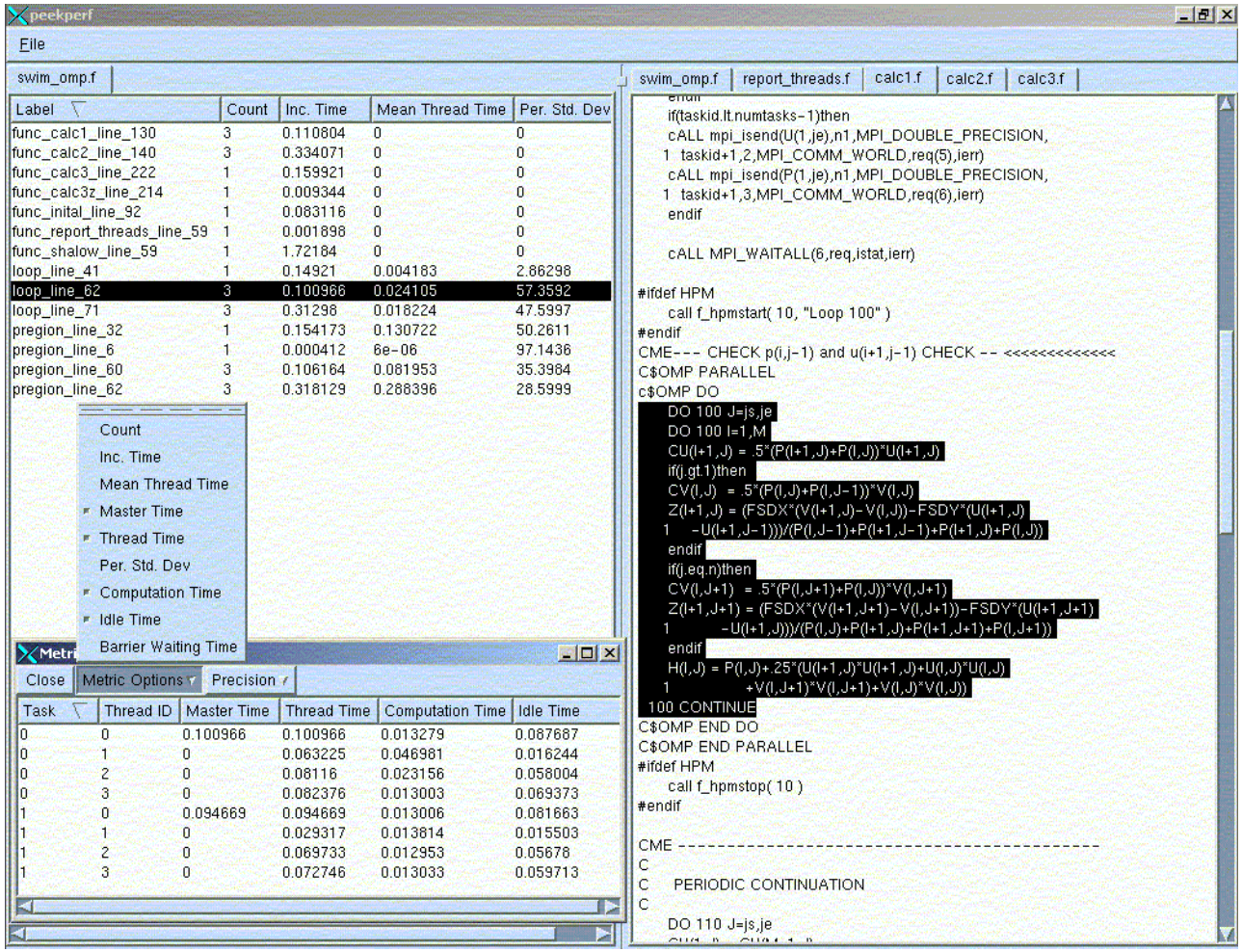
```
% elg2vtf3 a.elg
% vampir a.vpt
```



laderose@us.ibm.com, b.mohr@fz-juelich.de, srseelam@us.ibm.com

Viewing OpenMP Profiler Traces with PeekPerf

% **peekperf *.viz**



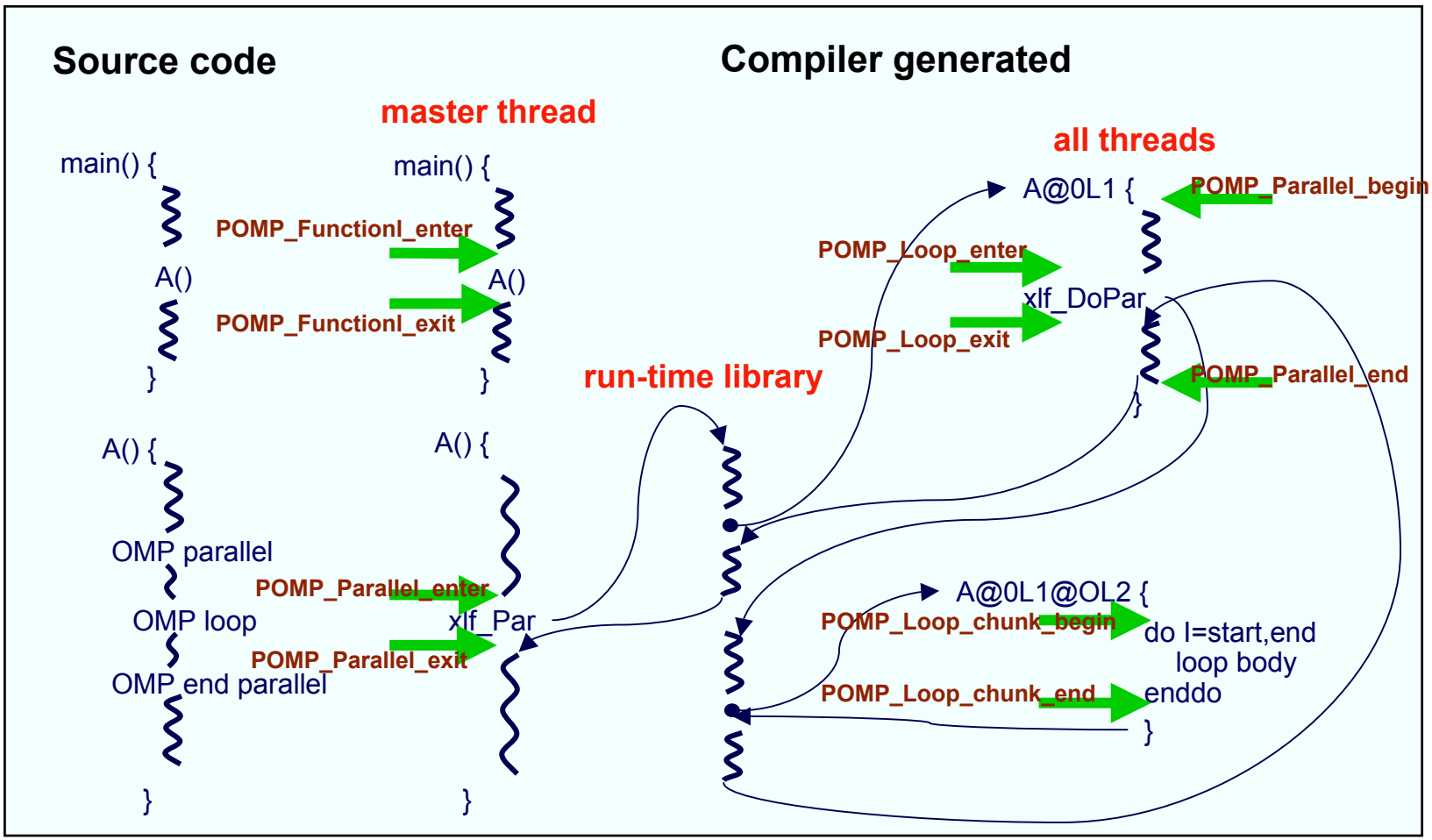
laderose@us.ibm.com, b.mohr@fz-juelich.de, srseelam@us.ibm.com

dpomp: A Performance Monitoring Interface for OpenMP

QUESTIONS?

dPOMP: DPCL based POMP

- The IBM compiler and run-time library



laderose@us.ibm.com, b.mohr@fz-juelich.de, srseelam@us.ibm.com