OMPD and a Case Study with STAT

Scalable Tools Workshop

Ignacio Laguna and Gregory L. Lee

August 2, 2016

LLNL-PRES-699267
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

Debugging OpenMP Programs

Original code

```
#pragma omp parallel
{
   a[i] = ...
}
```

Translated code

```
void parallel_region_block()
{
   a[i] = ...
}
...
omprt_run_parallel(parallel_region_block);
// code after parallel region
```

Breakpoint

What programmers see

Stack trace of **team member** thread

```
in clone () from libc
in start_thread () from libpthread
in omprt_internal () from libopenmp
in parallel_region_block ()
```

Problems

- No history information of the parallel region
- Programmers don't want to see runtime information

What programmers would like to see

```
in block ()
in #omp parallel from file:X
```

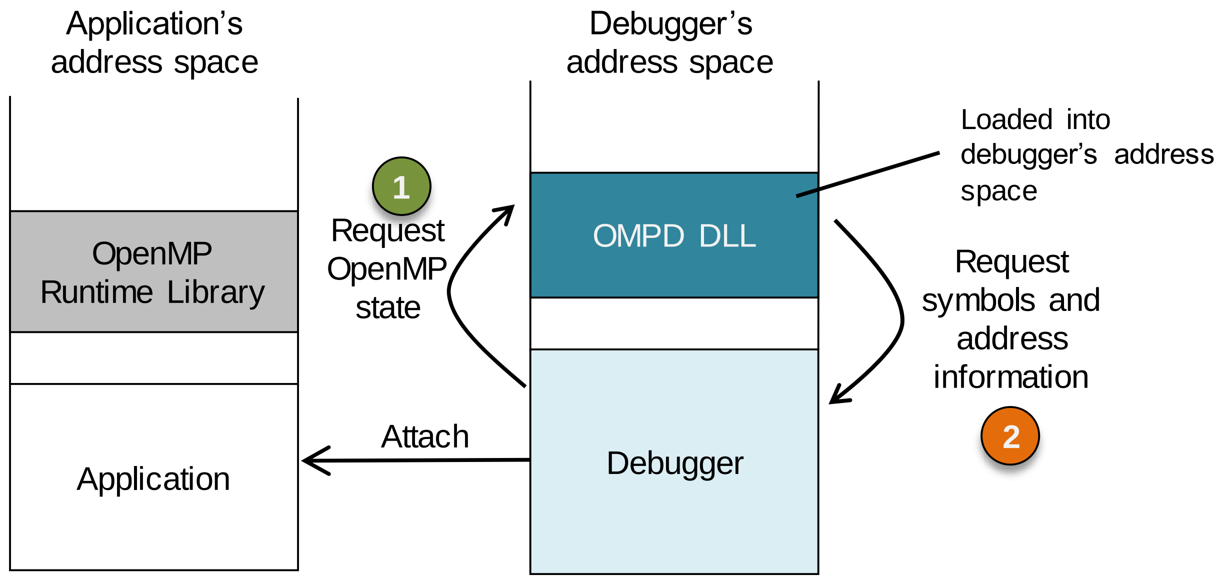
OMPD: *OpenMP Debugging Interface*

- API to allow debuggers understand state of OpenMP runtime

- Cross-runtime solution to debug OpenMP programs
  - — Currently each parallel debugger has its own solution

- Many use cases:
  - Place breakpoints in parallel regions
  - Check state of threads
  - Tasks parent/child relationships
  - Others…. see STAT use case

Workflow of OMPD


Application's address space
Debugger's address space
Loaded into debugger's address space
1
Request OpenMP state
OMPD DLL
OpenMP Runtime Library
Request symbols and address information
2
Attach
Debugger
Application

1 • Handles for threads, parallel regions, tasks

2 • Find symbols and addresses in target process

Status Update of OMPD

- We have a prototype of an OMPD library
  - Intel / Clang OpenMP Runtime
  - OpenMP 3.x only
- We are testing OMPD in multiple debuggers
  - GDB (callbacks using GDB)
  - STAT (callbacks using DynInst)
  - TotalView
- OMPD technical specification has been extended
  - RogueWave, RWTH Aachen, LLNL
- Specification document has been made public
  - https://github.com/OpenMPToolsInterface/OMPD-Technical-Report

Implemented functions

```
ompd_finalize
ompd_get_active_level
ompd_get_ancestor_task_region
ompd_get_display_control_vars
ompd_get_dynamic
ompd_get_enclosing_parallel_handle
ompd_get_implicit_task_in_parallel
ompd_get_level
ompd_get_master_thread_in_parallel
ompd_get_max_active_levels
ompd_get_max_threads
ompd_get_nested
ompd_get_num_procs
ompd_get_num_threads
ompd_get_osthread
ompd_get_parallel_function
ompd_get_parallel_handle_string_id
ompd_get_parallel_id
ompd_get_proc_bind
ompd_get_schedule
ompd_get_state
ompd_get_task_enclosing_parallel_handle
ompd_get_task_frame
ompd_get_task_function
ompd_get_task_handle_string_id
ompd_get_task_id
ompd_get_thread_handle
ompd_get_thread_handle_string_id
ompd_get_thread_in_parallel
ompd_get_thread_limit
ompd_get_thread_num
ompd_get_threads
ompd_get_top_parallel_region
ompd_get_top_task_region
ompd_get_version
ompd_get_version_string
ompd_in_final
ompd_in_parallel
ompd_initialize
ompd_is_implicit
ompd_parallel_handle_compare
ompd_process_initialize
ompd_release_address_space_handle
ompd_release_display_control_vars
ompd_release_parallel_handle
ompd_release_task_handle
ompd_release_thread_handle
ompd_task_handle_compare
ompd_thread_handle_compare
```

OMPD Project Contributors

LLNL

- Ignacio Laguna
- Dong Ahn
- Martin Schulz
- Marty Mcfadden

Rogue Wave Software

- Ariel Burton
- John DelSignore

RWTH Aachen University

- Joachim Protze

Rice University

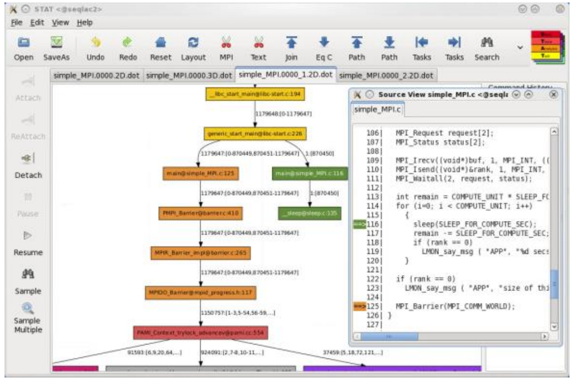
- John Mellor-Crummey
- Lai Wei

IBM

- Alexandre Eichenberger

The Stack Trace Analysis Tool (STAT) is a major success story for scalable tools development and deployment

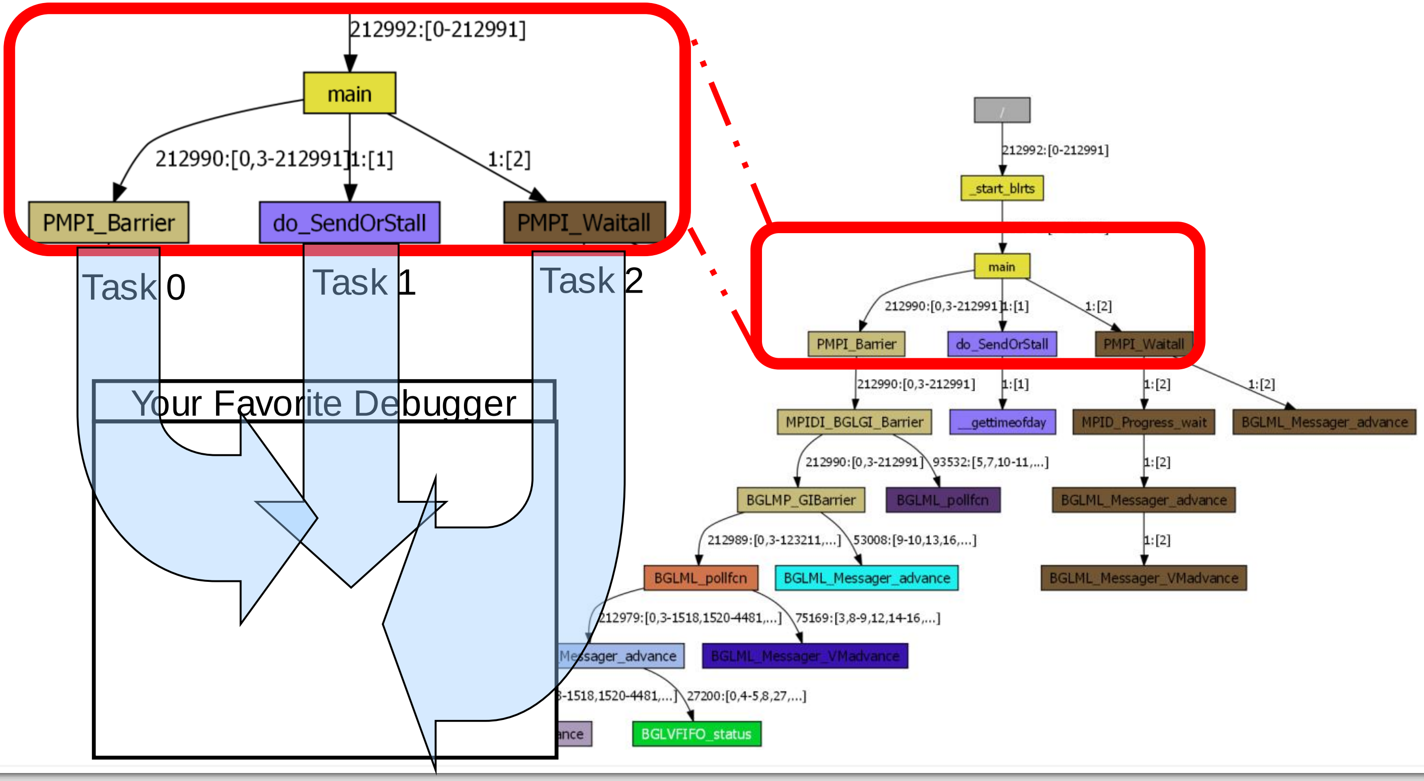
- STAT enables debugging millions of processes
  - Modular and highly scalable software architecture
  - Lightweight analysis and concise user display
- STAT has been crucial to fix production bugs
  - Identified 3 million task hang of pf3d on Sequoia
  - Widely used on LC HPC systems
  - Deployed and used at other sites, including DOE labs
    - Packaged in Cray Linux Environment

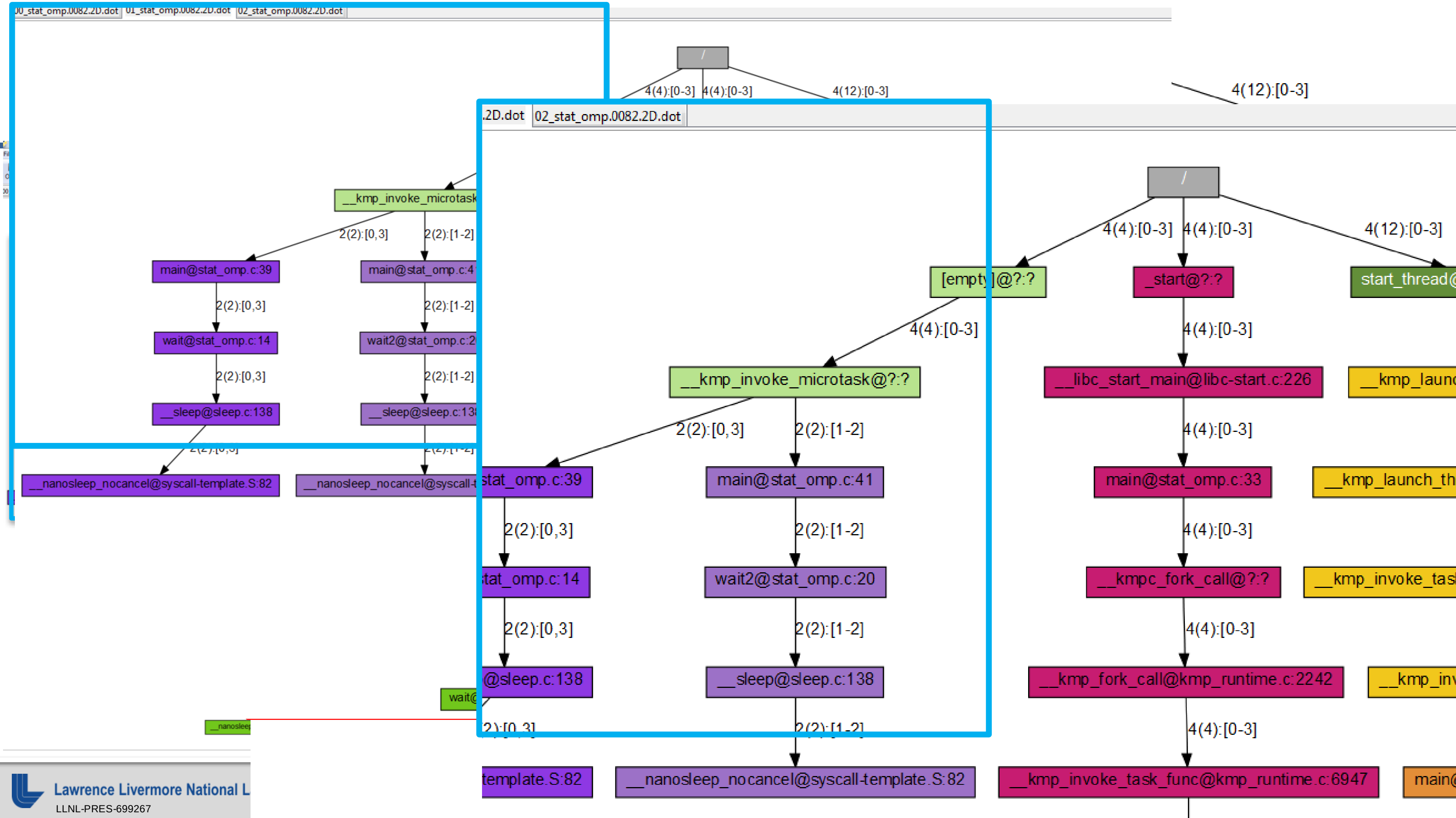


```
  MPI_Request request[2];
  MPI_Status status[2];

  MPI_Irecv((void*)buf, 1, MPI_INT, ((
  MPI_Isend((void*)&rank, 1, MPI_INT,
  MPI_Waitall(2, request, status);

  int remain = COMPUTE_UNIT * SLEEP_FO
  for (i=0; i < COMPUTE_UNIT; i++)
    {
      sleep(SLEEP_FOR_COMPUTE_SEC);
      remain -= SLEEP_FOR_COMPUTE_SEC;
      if (rank == 0)
        LMON_say_msg ( "APP", "%d secs
    }

  if (rank == 0)
    LMON_say_msg ( "APP", "size of thi

  MPI_Barrier(MPI_COMM_WORLD);
}
```

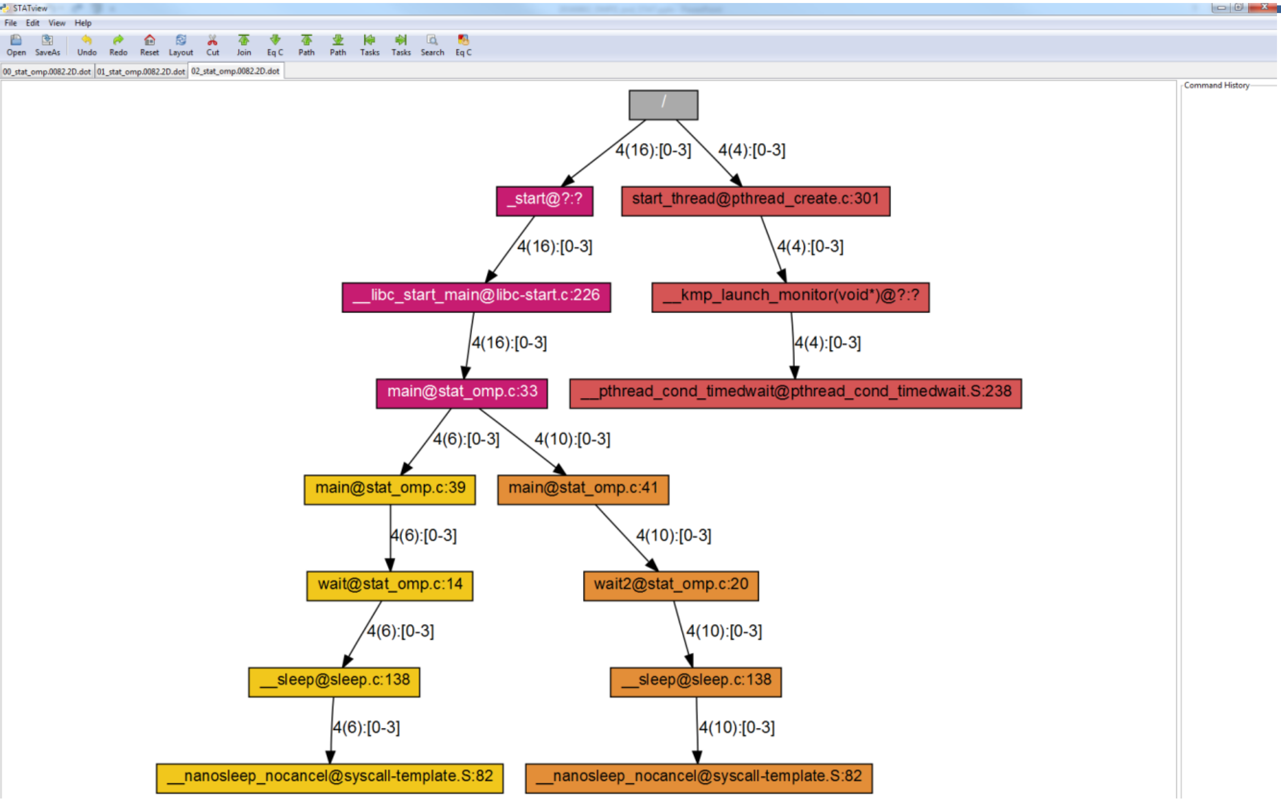
- Collaborative project between LLNL and university partners
  - Prototyped by student during a summer internship
  - Development continues with University of Wisconsin, University of New Mexico, and Denmark Technical University
- Winner of a 2011 R&D 100 award

STAT merges stack traces to identify similarities and differences




00_stat_omp.0082.2D.dot
01_stat_omp.0082.2D.dot
02_stat_omp.0082.2D.dot
/
4(4):[0-3]
4(4):[0-3]
4(12):[0-3]
[empty]@?:?
_start@?:?
start_thread@
__kmp_invoke_microtask@?:?
__libc_start_main@libc-start.c:226
__kmp_launc
2(2):[0,3]
2(2):[1-2]
main@stat_omp.c:39
main@stat_omp.c:41
main@stat_omp.c:33
__kmp_launch_th
wait@stat_omp.c:14
wait2@stat_omp.c:20
__kmpc_fork_call@?:?
__kmp_invoke_tas
__sleep@sleep.c:138
__kmp_fork_call@kmp_runtime.c:2242
__kmp_inv
__nanosleep_nocancel@syscall-template.S:82
__kmp_invoke_task_func@kmp_runtime.c:6947
main@

OMPD provides an application-oriented view



- OpenMP runtime frames filtered out
- Worker threads grafted to spawn location

More Information

- OMPD
  - http://openmp.org/mp-documents/ompt-tr.pdf
- STAT
  - http://www.paradyn.org/STAT/STAT.html
  - https://github.com/LLNL/STAT
- Contact Info
  - Ignacio Laguna lagunaperalt1@llnl.gov
  - Greg Lee lee218@llnl.gov

Lawrence Livermore National Laboratory