Overcoming Distributed Debugging Challenges in the MPI+OpenMP Programming Model

Lai Wei², Ignacio Laguna¹, Dong H. Ahn¹, Matthew P. LeGendre¹, Gregory L. Lee¹
¹ Lawrence Livermore National Laboratory, ² Department of Computer Science, Rice University

Motivation
- Exascale computing will embrace a wide range of programming models.
- Programming models, such as OpenMP, harness the many levels of architectural parallelism (CPUs and devices).
- Debugging tools must help programmers identify errors at the level of the programming model.
- Question: What are the effective levels for debugging in hybrid programming models such as MPI+OpenMP?

Contributions
- Our framework reconstructs user-level call stacks for OpenMP threads.
- We share lessons learned from restructuring OpenMP call stacks.
- Our extension to STAT provides an easy-to-understand view of a stack trace to help programmers debug MPI+OpenMP programs at the user code level.

Challenges
1. Call stacks of OpenMP worker threads provide only partial calling context.
2. Stack frames from routines inside an OpenMP library implementation are confusing to debugger users.

Background on OMPD
OMPD provides debuggers access to OpenMP runtime constructs. Debuggers interact with OMPD to get information, such as a thread’s current status and a thread’s current parallel region.

Approach
We built an OpenMP stack walker that uses OMPD in conjunction with Dyninst to rebuild call stacks for OpenMP threads. We updated STAT so that it uses the OpenMP stack walker to collect call stacks of threads in MPI+OpenMP programs.

Accomplishments
- Reconstructing full calling context for all OpenMP threads.
- Eliminating internal frames of OpenMP runtime library.
- Presenting OpenMP library entry frames upon request.
- Incorporating these capabilities into STAT to support MPI+OpenMP debugging.

We leverage OMPD to reconstruct call stacks for OpenMP threads in STAT. This provides better support for debugging MPI+OpenMP programs.

Our results show that we are able to present users with intuitive stack trace views for MPI+OpenMP programs.

Future Work
1. Refine STAT call stack views for MPI+OpenMP programs and evaluate STAT on large applications.
2. Support debugging of OpenMP 4.0 programs, which can include omp target constructs.
3. Present views of OpenMP tasks as well as identify master and worker threads in MPI+OpenMP programs.

References

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.