Programming-Model Centric Debugging for OpenMP

Kevin Pouget, Jean-François Méhaut and Miguel Santana

Context Developing applications that should run efficiently on multiple cores in parallel is significantly harder than writing sequential code. To simplify the application developer's job, many parallel programming runtime frameworks have been developed during the last decades. From Posix thread standard to OpenMPparallel computing, from MPI message passing interface to dataflow scheduling, these tools try to raise the abstraction level used during application development.

However, current debugging tools and techniques do not usually take into account these higher levels of abstraction, and only consider the underlying programming languages. This is problematic, firstly because these tools do not exploit the high-level information that could be useful to developers; and secondly because it impairs their debugging activity, as developers have to manually cope with the interactions between the application, the runtime libraries and the operating system that are involved in the programming model implementation.

Contribution In this presentation, we introduce a new approach for debugging parallel applications based on the OpenMPprogramming standard. This approach is centered on the programming model, by opposition with more traditionnal debugging methods focused on the programming language and machine code.

We detail the main axes of programming-model centric debugging: providing a structural representation of the application, following the execution dynamic behaviors and allowing interactions with the abstract and physical machines; and we explain how they apply to OpenMPfork-join and task-based programming.

Implementation Then, we introduce key insights related to the implementation of such a debugger. We explain how we dynamically capture the state of the OpenMPabstract machine using GDB internal breakpoints, and discuss the trade-offs of this implementation choice in terms of ease of implementation and efficiency. We propose two alternative techniques with different benefits.

We implemented our debugger support for Gnu Gomp and Intel OpenMPto emphasize that only the top layer of the code is implementation specific. We also mention how OpenMPTrace and Debugging interfaces (OMPT/OMPD)

appear promising for our tool to become independent of the actual OpenMPimplementation, although the interfaces are not mature enough in their current form.

Finally, we focus on the data-dependent tasks of OpenMP4.0 and detail how we dynamically reconstruct the task graph and present it through Temanejo graphical interface. We also extended GDB command set to include catchpoints on task creation, execution, etc. We finish the presentation with an illustration of our debugger usage through different use-case situations. We highlight how it helps controlling the application execution, and how its high-level representation of the OpenMPabstract machine facilitates the detection of bugs related to the data-dependency graph.

Future work Programming-model centric debugging is a new approach of source-level debugging, that aims at providing tools more adapted to high-level programming environments such as OpenMP. In the following of this work, we would like to extend the idea to performance debugging. By leveraging the debugger ability to control the execution and its knowledge of the runtime environments, we would like to open it up towards interactive execution profiling. Such a profiler would operate both at language and model level, to offer different granularity of control.