A Deadlock Detection Concept for OpenMP Tasks and Fully-Hybrid MPI-OpenMP Applications

Tobias Hilbrich
Technische Universität Dresden, Germany

Bronis R. de Supinski
Lawrence Livermore National Laboratory, USA

Andreas Knüpfer
Technisch Universität Dresden, Germany

Robert Dietrich
Technisch Universität Dresden, Germany

Christian Terboven
RWTH Aachen, Germany

Felix Münchhalfen
RWTH Aachen, Germany

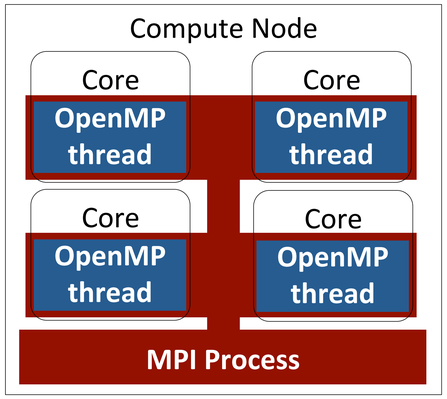
Wolfgang E. Nagel
Technische Universität Dresden, Germany

http://bit.ly/sc15poster

Current high performance computing applications often combine the Message Passing Interface (MPI) with threaded parallel programming paradigms, e.g., OpenMP. MPI allows fully hybrid applications in which multiple threads of a process issue MPI operations concurrently. Little study on deadlock conditions for this combined use exists. We propose a wait-for graph approach to understand and detect deadlock for such fully hybrid applications. It specifically considers OpenMP 3.0 tasking support to incorporate OpenMP's task-based execution model. Our model creates dependencies with deadlock criteria that can be visualized to support comprehensive deadlock reports. We use a model checking approach to investigate wide ranges of valid execution states of example programs to verify the soundness of our wait-for graph construction.

MPI, OpenMP, Tasks, and Hardware

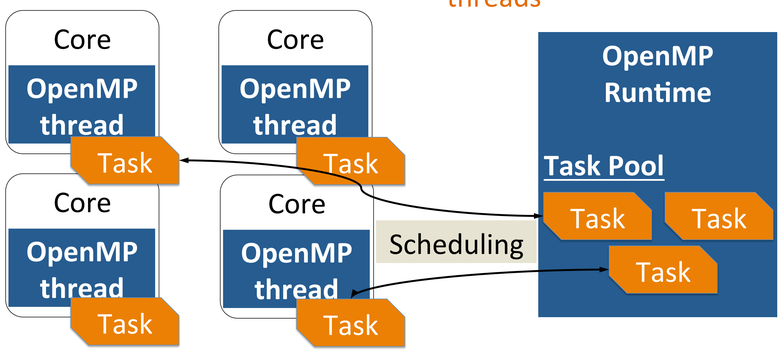
- Increasing core counts per compute node



⇒ **Process**:
From MPI, execute the same program, distinct virtual memory, message communication

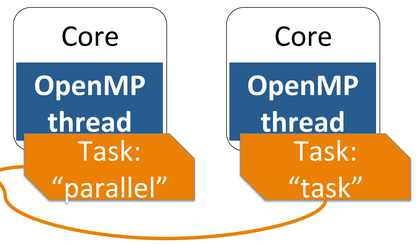
⇒ **Thread**:
From OpenMP, share memory within their process

⇒ **Task**:
Managed by OpenMP runtime, executes on threads

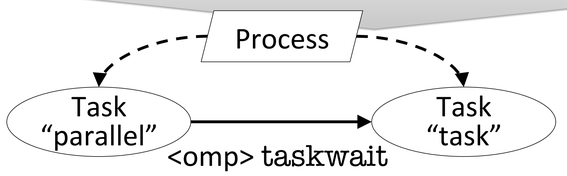


OpenMP Tasks and Deadlocks

```
1 #pragma omp parallel
2 {
3 #pragma omp task
4   {
5     ... /*Do something*/
6   } /*end: omp task*/
7 #pragma omp taskwait
8 } /*end: omp parallel*/
```



Wait-For Graph Construction Rules



Deadlock?
⇒No

Is my MPI-OpenMP application deadlocked?

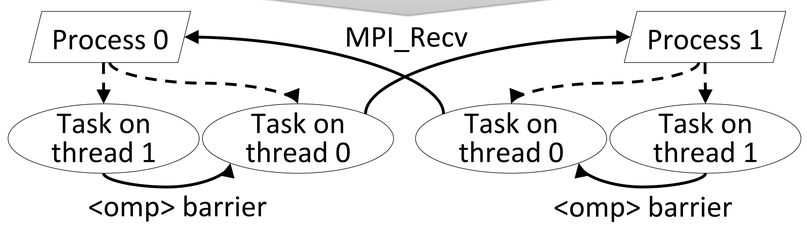
Wait-For Graph (WFG) Construction Rules

I. Each MPI process i get one node in the WFG
II. Each task t_i of a process gets one node in the WFG
III. A node for process i waits for all of its task nodes t_i (OR semantic)
IV. An unscheduled task t_i waits for its parent process i (with AND semantic)
V. If any task t_i is at a task scheduling point, ignore rule IV. for all tasks that could replace t_i
VI. Model individual wait-for dependencies for MPI/OpenMP directives active on tasks
VII. MPI Dependencies target process nodes, not task nodes

OpenMP-MPI Deadlock

Process 0		Process 1	
Thread 0	Thread 1	Thread 0	Thread 1
MPI_Recv(from:1)		MPI_Recv(from:0)	
#pragma omp barrier		#pragma omp barrier	
	MPI_Send(to:1)		MPI_Send(to:0)

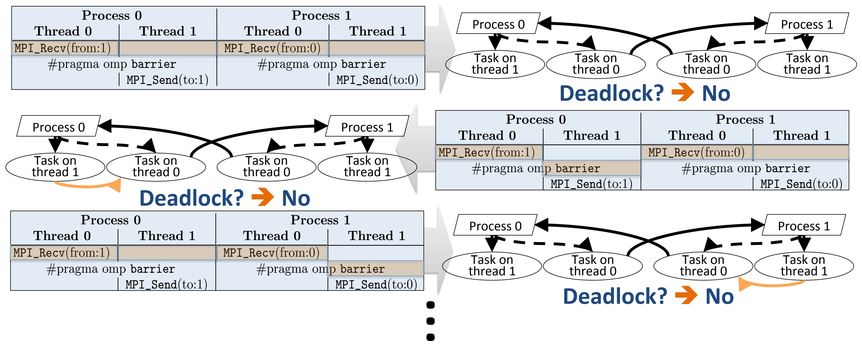
Wait-For Graph Construction Rules



Deadlock?
⇒Yes

Soundness

- Are the construction rules sound ?
- Approach: Enumerate a large set of
  - ⇒Evaluate large numbers of application execution states
  - ⇒Apply construction rules to these states
  - ⇒Test WFG for deadlock
  - ⇒Compare result to expectation => Should this state have a deadlock or not?
  - ⇒We demonstrate no false positives/negatives for 13 examples



Contact: tobias.hilbrich@tu-dresden.de