Abstract

Task based programming models (e.g., Cilk, Intel TBB, X10, Java Fork-Join tasks) simplify multicore programming in contrast to programming with threads. In a task-based model, the programmer specifies the parallel tasks and the runtime maps these tasks to hardware threads. The runtime automatically balances the load using a work-stealing runtime and provides performance portability. However, interference between parallel tasks can result in concurrency errors such as data races, atomicity violations, and deadlocks. This paper proposes a dynamic analysis technique to detect all atomicity violations for a given input for a class of task parallel programs without requiring interleaving exploration. The proposed technique maintains asymptotically constant amount of metadata with each shared memory location, which is independent of the number of dynamic tasks, and leverages the series-parallel structure of a task parallel program execution to detect such errors. Our prototype tool for Intel Threading Building Blocks (TBB) detects atomicity violations with performance overheads similar to techniques that detect atomicity violation in a given schedule. Our approach provides a stronger guarantee of detecting atomicity violations that occur in any schedule for a given input in contrast to prior approaches that detect atomicity violations in a given interleaving.