

MIT OpenCourseWare
<http://ocw.mit.edu>

6.189 Multicore Programming Primer, January (IAP) 2007

Please use the following citation format:

Saman Amarasinghe and Rodric Rabbah, *6.189 Multicore Programming Primer, January (IAP) 2007*. (Massachusetts Institute of Technology: MIT OpenCourseWare). <http://ocw.mit.edu> (accessed MM DD, YYYY).
License: Creative Commons Attribution-Noncommercial-Share Alike.

Note: Please use the actual date you accessed this material in your citation.

For more information about citing these materials or our Terms of Use, visit:
<http://ocw.mit.edu/terms>

It is not far fetched to imagine the “*print*” approach is the most widely used debugging utility. Novice and expert programmers alike use this approach to print out information as their buggy code executes in order to reveal clues about their code defects. This approach works reasonably well in sequential programs with a single thread of execution, but is not likely to be as useful for debugging parallel programs because of the multiple threads of execution.

What complicates the debugging process for parallel codes, and if you were to build a debugging tool, what features might you provide to help programmers track down their bugs productively?

The non-deterministic nature of parallel execution with multiple threads makes it difficult to diagnose problems because execution behavior may not be repeatable. Hence symptoms such as race conditions or deadlock may disappear from one run to another, or the addition of debugging code (e.g., print statement) affect timing in such a way that race conditions or deadlocks seemingly disappear.

Debugging tools for parallel programming have to provide functionality to diagnose behavior that may arise from non-deterministic execution. For example, address-value traces may be useful in diagnosing race conditions, and tracking communication patterns between threads can help to identify deadlock issues. Additionally checkpointing and replay features may also be useful, as are visualization facilities although scaling the visual debuggers to hundreds or thousands of threads may prove difficult.