The Case for Region Serializability

Jessica Ouyang, Peter Chen, Jason Flinn, & Satish Narayanasamy
University of Michigan

MICHIGAN

Semantics for Racy Programs

No clean semantics for programs with data races running on multicore

Problems

- Weak shared memory model
- Difficult to write, test, and debug multithreaded code

Solution

- Provide ***sequential*** semantics to ***all*** code
- Guarantee serializability at ***larger*** granularity of code

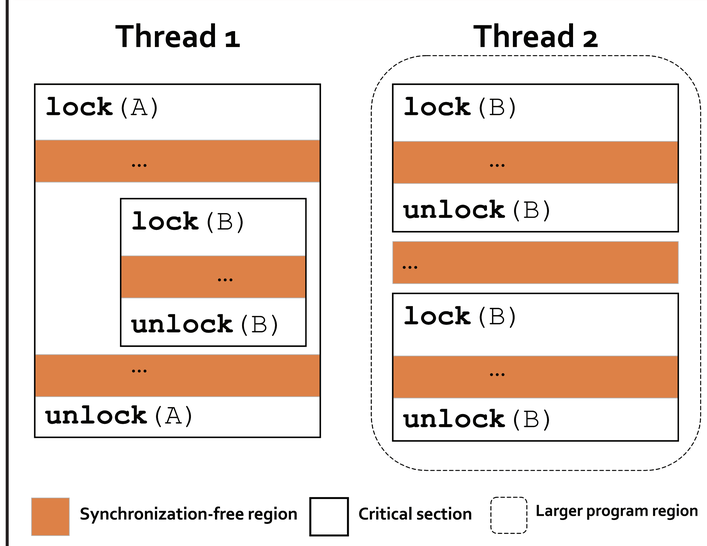
Benefits

- Easier for ***programmers*** to understand
- Strengthens guarantees of existing tools
- Allows ***static & dynamic program analysis, model checking, and testing*** to scale

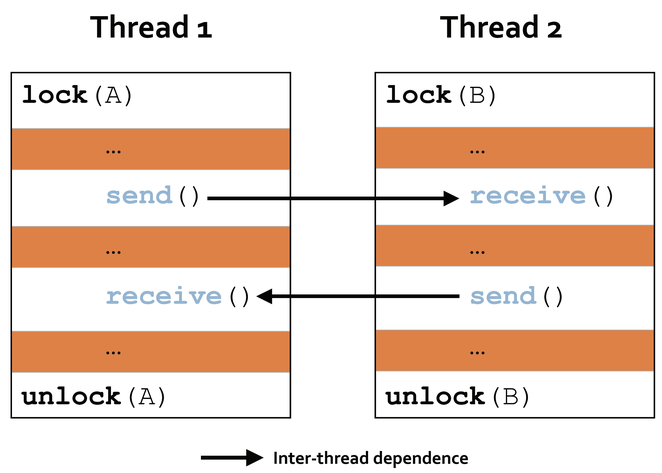
Region Serializability

Guarantee sequential consistency at larger granularity of code

Regions in Source Code



Avoiding Potential Deadlocks

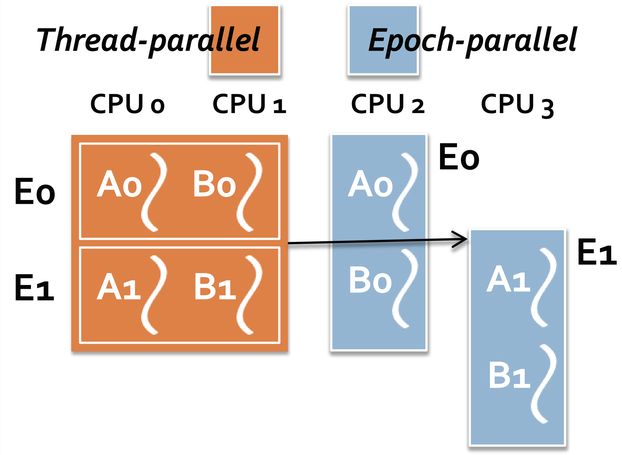


Compiler must not ***optimize*** across region boundaries

Runtime only ***preempts*** at beginning and end of regions

Uniparallel Execution

Serialize regions while ***scaling*** with cores



Uniprocessor epoch-parallel execution guarantees serializability

Multiprocessor thread-parallel accelerates uniprocessor execution