Section Based Program Analysis to Reduce Overhead of Detecting Unsynchronized Thread Communication
Madan Das, Gabriel Southern, Jose Renau
Dept. of Computer Engineering
University of California, Santa Cruz

Features:
- Static analysis techniques
- Eliminates instrumentation at compile time
- Useful for race detection, deterministic execution engines and STMs
- Works with ThreadSanitizer
- Works for multi-threaded C and C++ programs
- Precise section based alias analysis
- Augmented with verifiable directives
- Validated with parsec, splash and phoenix suites
- Implemented as LLVM pass
- Holistic solution to detect data race issues
- Open source

http://masc.soec.ucsc.edu/sbpa

Programs have phases
- Identifying phases in parallel code can improve precision of alias analysis
- Most data accesses in parallel code are non-communicating (non-racy and independent in same phase)

Section Identification in SBPA
1. Build reduced inter-procedural CFG
2. Find multithreaded code sections (MTCS) enclosed by create/join
3. foreach MTCS section ts:
   .1 let b = beginning of ts
   .2 let e = end of ts
   .3 while b != e:
     .1 C = reachable barrier nodes starting from b.
     .2 if C has a single node
       .1 Code from b to C is a new thread section
       .2 b = C
     .3 else exit search

SBPA is effective!
- Eliminated 63% of total memory instrumentations

SBPA is accurate
- Validated with PARSEC, SPLASH and Phoenix suites

Integration with ThreadSanitizer
- ThreadSanitizer slows down 12.5 times
- SBPA speeds it up 2.74 times
- Still detects the same races

Per benchmark speed-ups with 2, 4 and 8 threads

References: