DeCoP: Deterministic Cooperative Parallelism

Section Based Program Analysis to Reduce Overhead of Detecting Unsynchronized Thread Communication

Madan Das, Gabriel Southern, Cormac Flanagan, Jose Renau
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.soe.ucsc.edu/sbpa

SBPA is effective!

➔ Eliminated 63% of total memory instrumentations

SBPA is accurate

➔ Validated with PARSEC, SPLASH and Phoenix suites

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:
   .4 if C is reachable barrier nodes starting from b.
   .5 Code from b to C is a new thread section
   .6 b = C
   .7 else exit search

Integration with ThreadSanitizer

➔ ThreadSanitizer slows down 12.5 times
➔ SBPA speeds it up 2.74 times
➔ Still detects the same races

Compensation time normalized to Baseline

68% of loads are proven non-communicating. Baseline is Coredet.

Geometric means of speed-ups in different modes

References: