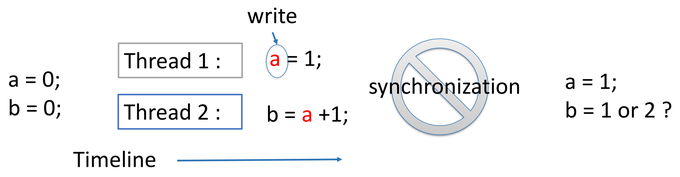
DataRaceBench: A Benchmark Suite for Systematic Evaluation of Data Race Detection Tools

Chunhua "Leo" Liao, Pei-Hung Lin, Joshua Asplund, Markus Schordan and Ian Karlin

Center for Applied Scientific Computing, Lawrence Livermore National Laboratory

Motivation

A data race occurs when two concurrent threads access a shared variable and at least one access is a write, and the threads use no explicit mechanism to prevent the accesses from being simultaneous.



Huge Threat to correctness of all multithreaded applications: already caused disastrous damages in medical equipment and electrical grids.

Radiation therapy machine accidents in 1980s

Northeast blackout in 2003

Self-driving cars?

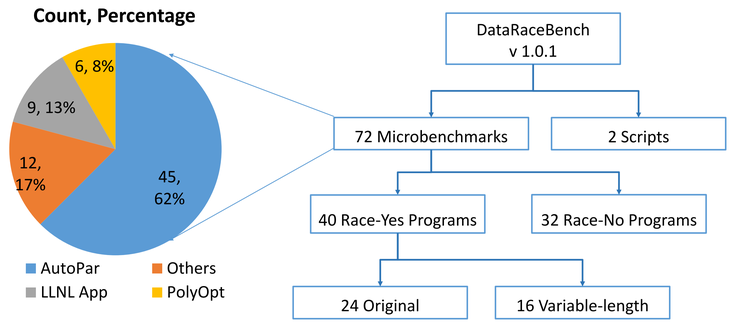
Problem: many tools have been developed. But no dedicated benchmark suites to evaluate data race detection tools.

Data Race Detection Tools	Benchmarks
Archer '16	Kernels (OmpSCR), Real/proxy apps (AMG2013, HYDRA)
PolyOMP '16	Kernels (OmpSCR), Perf. bench.(PolyBench-ACC)
Versatile On-the-fly Race Detection'14	Perf. bench. (ECPP, NPB)
OpenMP Analysis Toolkit (OAT)'13	Perf. bench. (NPB)
RaceMob '13	Real apps. (Apache httpd, SQLite, Memcached)
ompVerify '11	Kernels (Stencil, matrix transpose, sort)
Intel Thread Checker'03	Kernels (Histogram)

Solution: a dedicated data race detection benchmark suite

- Focus on OpenMP: the most popular programming model for writing multithreaded programs in high-performance computing (HPC)
- Capture representative code patterns with and without data races
- Generate quantitative metrics to enable apple-to-apple comparison
- Discover strengths and limitations of data race detection tools
- Accessible through open-source release: https://github.com/LLNL/dataracebench
- Best paper finalist of SuperComputing'17: https://dl.acm.org/citation.cfm?doid=3126908.3126958

Details of DataRaceBench



Multiple sources to ensure representativeness of microbenchmarks:
- tests from two OpenMP code optimization tools : AutoPar and PolyOpt
- LLNL applications, and
- Others , including literature

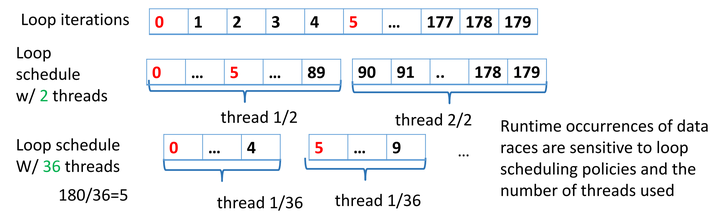
When applicable, each race-yes program has at most a single pair of source locations causing runtime data races pairs up with a race-no program

Property labels for race-yes set	#	Property labels for race-no set	#
Y1: Unresolvable dependences	23	N1: Embarrassingly parallel	7
Y2: Missing data sharing clauses	6	N2: Use of data sharing clauses	9
Y3: Missing synchronization	4	N3: Use of explicit synchronization	2
Y4: SIMD data races	2	N4: Use of SIMD directives	4
Y5: Accelerator data races	1	N5: Use of accelerator directives	1
Y6: Undefined behaviors	2	N6: Use of special language features	5
Y7: Numerical kernel data races	4	N7: Numerical kernels	9
Total #	42		37

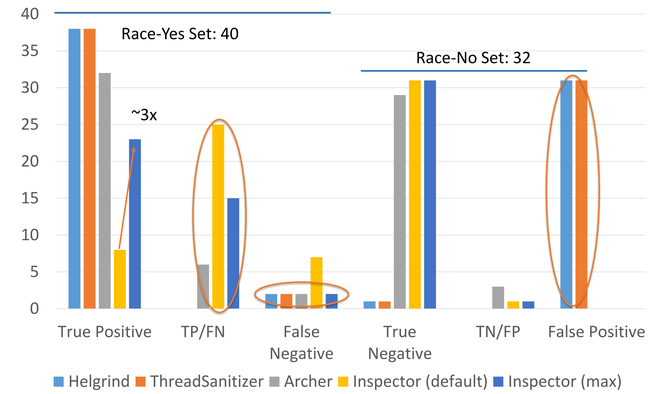
```
1.  int indexSet[180] = {
2.   521, 523, 525, 527, 529, 533,
3.   547, 549, 551, 553, 555, 557,...
4.   };
5.  double * xa1, *xa2; ...
6.  xa2 = xa1 + 12 ;
7.  #pragma omp parallel for
8.  for(int i=0; i< 180; ++i)
9.  {
10.   int idx=indexSet[i];
11.   xa1[idx]+=1.0;
12.   xa2[idx]+=3.0;
13. }
```

An example microbenchmark for a numerical computation pattern
indirectaccess2-orig-yes.c
Data race pair:
xa1[idx]@11 vs. xa2[idx]@12

iteration i	0	1	...	5
indexSet[i]	521	523	...	533
xa1[indexSet[i]]	base+521		...	base+533
xa2[indexSet[i]]	base+12+521		...	



Evaluating Data Race Detection Tools Using DataRaceBench



Evaluating four popular data race detection tools
- Pthreads-based: Helgrind, ThreadSanitizer
- OpenMP-aware: Archer, Intel Inspector(two configurations: default and maximum resources)

Hardware: Quartz cluster@LLNL supporting 72 threads per node
Execution configuration: Tools x microbenchmarks x OpenMP threads x Array Sizes x Repeats =(4+1) x 5320

Term	Meaning in our context
True Positive (TP)	Detecting data races in a race-yes program
False Positive (FP)	Detecting data races in a race-no program
True Negative (TN)	Not detecting data races in a race-no program
False Negative (FN)	Not detecting data races in a race-yes program

Metric	Formula
Precision	Confidence of true positive P = TP/(TP + FP)
Recall	Completeness of true positive R = TP/(TP + FN)
Accuracy	Chance of having a correct report A = (TP +TN)/(TP + FP +TN + FN)

Tool	Precision		Recall		Accuracy	
	min	max	min	max	min	max
Helgrind	0.551	0.551	0.950	0.950	0.542	0.542
ThreadSanitizer	0.551	0.551	0.950	0.950	0.542	0.542
Archer	0.914	1.000	0.800	0.950	0.847	0.972
Intel Inspector(default)	0.889	1.000	0.200	0.825	0.542	0.903
Intel Inspector(max)	0.958	1.000	0.575	0.950	0.750	0.972

Conclusion

DataRaceBench: the first dedicated OpenMP benchmark suite to enable systematic and quantitative evaluation of data race detection tools
- Configurations of dynamic tools matter: Intel default vs. max resources, number of threads and scheduling policies
- Multiple runs: necessary to increase probability of finding data races
- Precision/Accuracy: Archer and Intel Inspector win over Helgrind and ThreadSanitizer due to OpenMP awareness
- User friendliness: Only Intel inspector consolidates multiple data race instances of same origin
- SIMD loops with data races: compilers do no generate SIMD instructions for our race-yes SIMD benchmarks

LLNL-POST-744937