Exploring Regression of Data Race Detection Tools Using DataRaceBench

—

3rd International Workshop on Software Correctness for HPC Applications, Nov 18, 2019, Denver, CO, USA

Pei-Hung Lin, Chunhua Liao, Markus Schordan, Ian Karlin

Lawrence Livermore National Laboratory, CA, USA

—

External Audience (Unlimited)
LLNL-PRES-796457

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and was supported by the LLNL-LDRD Program under Project No. 18-ERD-006. Lawrence Livermore National Security, LLC.

Overview

- Introduction
- Evaluation of 4 dynamic tools, 1 verification tool
- Evaluation with DataRaceBench
- Dynamic Analysis vs. Sound Static Analysis vs. Verification
- Regression of popular dynamic data race detection tools
- Conclusion

Introduction

Definition: What is a data race?

Data races occur when multiple threads perform simultaneous conflicting data accesses to the same memory location without proper synchronization and at least one is a write access.

Reasons for the Existence of Data Races

- A data race exists when synchronization between threads is missing.
- Additional synchronization slows down the execution of a parallel program.
- Data races can be dependent on the thread schedule and can be difficult to reproduce and to detect.

DataRaceBench 1.2.0

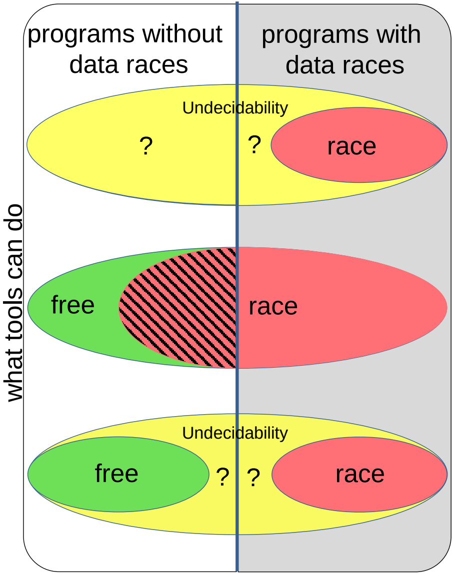
Benchmarks for data race detection tool evaluation

- 116 OpenMP micro-benchmarks
- Each benchmark has exactly one pair of program locations whose execution causes a data race
- Two groups of benchmarks: with and without data race
- Download: https://github.com/LLNL/dataracebench

DataRaceBench Evaluation Dashboard

- Latest evaluation results available on-line
- Regularly updated
- All evaluation data also available as database file for download
- https://github.com/LLNL/dataracebench/wiki/Regression-metrics

Testing, Static Analysis, Verification



- Software Testing tools
  - Google Thread Sanitizer (state-of-the-art according to [Effinger-Dean et al., 2012])
  - FastTrack(Java)/Aikido(C) (state-of-the-art according to [Effinger-Dean et al., 2012])
  - Archer - LLNL (based on Google's Thread Sanitizer, supports OpenMP) [Protze, Atzeni, Ahn, et al., 2014]
- Static Software Analyzer
  - LOCKSMITH (subset of C) [Pratikakis et al., 2006]
  - Relay (C code) [Voung et al., 2007]
- Software Verifier
  - BLAST: abstractions (nesC code) [Henzinger, 2003]
  - CHESS: **stateless** bounded MC [Musuvathi, 2008]
  - ompVerify (polyhedral loops) [Basupalli, 2011]
  - CIVL: symbolic execution, no abstraction [Zheng et al., 2015]

Factors Impacting Consistent Evaluation

Factor	Description	Solution
Benchmarks	Test programs and inputs used	DataRaceBench
Platforms	Hardware, OS and software	Standard VMs
Tools	Tool versions, configuration and installation	Docker Images
Process	The steps and settings used	Automated scripts
Results	Processing and interpreting results	Standard metrics
Randomness	Random results reported	Reporting ranges
Errors	Compile or runtime errors	Adding error rate metrics

Terminology and Metrics for Evaluation

Terminology

- True Positive (TP): data race exists and is reported
- False Negative (FN): data race exists, but is not reported
- False Positive (FP): no data race exists, but some data race is reported
- True Negative (TN): no data race exists and none is reported

Metrics

- Precision $P = \frac{TP}{TP+FP}$
- Recall $R = \frac{TP}{TP+FN}$
- Accuracy $A = \frac{TP+TN}{TP+TN+FP+FN}$

Dynamic Tools - Evaluation Statistics

Tool-Compiler	Tests	Test Results				Metrics			Testing Error				Test Time (hh:mm:ss)
		TP	FN	TN	FP	Prec.	Recall	Acc.	CSF	CUN	RSF	RTO	
Archer1.0-Clang3.9.1	376	187	24	145	0	1.00	0.89	0.93	5	5	10	0	00:06:11
Archer2.0-Clang6.0.0	386	202	20	156	3	0.99	0.91	0.94	0	5	0	0	00:06:17
Inspector2018-Intel17.0.2	392	195	30	156	9	0.96	0.87	0.90	2	0	0	0	01:32:50
Inspector2018-Intel18.0.2	396	198	27	160	8	0.96	0.88	0.91	0	0	0	3	02:04:34
Inspector2018-Intel19.0.0	396	213	12	60	108	0.66	0.95	0.69	0	0	0	3	03:41:17
Inspector2018-Intel19.0.4	396	198	27	160	11	0.95	0.88	0.90	0	0	0	0	01:33:54
Inspector2019-Intel17.0.2	392	195	30	159	6	0.97	0.87	0.91	2	0	0	0	01:37:08
Inspector2019-Intel18.0.2	396	195	30	162	6	0.97	0.87	0.91	0	0	0	3	02:04:49
Inspector2019-Intel19.0.0	396	214	11	61	107	0.67	0.95	0.70	0	0	0	3	03:32:55
Inspector2019-Intel19.0.4	396	195	30	164	7	0.97	0.87	0.91	0	0	0	0	01:37:27
ROMP-Clang8.0.0	384	198	18	144	6	0.97	0.92	0.93	0	6	9	3	00:59:20
Tsan5.0.2-Clang5.0.2	386	192	30	153	3	0.98	0.86	0.91	0	5	0	3	00:36:28
Tsan6.0.1-Clang6.0.1	386	195	27	156	3	0.98	0.88	0.92	0	5	0	0	00:07:34
Tsan7.1.0-Clang7.1.0	386	193	29	154	5	0.97	0.87	0.91	0	5	0	0	00:07:19
Tsan8.0.1-Clang8.0.1	384	184	38	152	4	0.98	0.83	0.89	0	6	0	0	00:07:03

- (C|R)SF: segmentation fault; CUN: compiler unsupported; RTO: timeout
- Archer 2.0 with `export TSAN_OPTIONS="ignore_noninstrumented_modules=1"` (by default not set!)
- ThreadSanitizer with LLVM OpenMP runtime with `LIBOMP_TSAN_SUPPORT` turned on (by default off!)

Metrics for the Tools

Tool	Corr. Success Rate	Precision Range	Recall Range	Accuracy Range
Archer1.0-Clang3.9.1	0.80	1.00-1.00	0.85-0.89	0.92-0.94
Archer2.0-Clang6.0.0	0.90	0.98-0.98	0.90-0.91	0.94-0.95
Inspector2018-Intel17.0.2	0.87	0.93-0.96	0.85-0.88	0.89-0.92
Inspector2018-Intel18.0.2	0.90	0.94-0.96	0.86-0.90	0.90-0.93
Inspector2018-Intel19.0.0	0.66	0.61-0.61	0.95-0.95	0.66-0.66
Inspector2018-Intel19.0.4	0.90	0.93-0.95	0.86-0.92	0.90-0.93
Inspector2019-Intel17.0.2	0.90	0.96-0.96	0.86-0.86	0.91-0.91
Inspector2019-Intel18.0.2	0.91	0.96-0.96	0.86-0.86	0.91-0.91
Inspector2019-Intel19.0.0	0.66	0.61-0.62	0.95-0.97	0.66-0.68
Inspector2019-Intel19.0.4	0.91	0.94-0.96	0.86-0.86	0.91-0.91
ROMP-Clang8.0.0	0.85	0.96-0.96	0.91-0.91	0.93-0.93
Tsan5.0.2-Clang5.0.2	0.84	0.98-0.98	0.79-0.91	0.88-0.95
Tsan6.0.1-Clang6.0.1	0.86	0.98-0.98	0.83-0.91	0.90-0.95
Tsan7.1.0-Clang7.1.0	0.84	0.96-0.98	0.79-0.91	0.87-0.95
Tsan8.0.1-Clang8.0.1	0.81	0.96-0.98	0.76-0.86	0.85-0.92

Ranges because multiple runs for each benchmark

Results showing Compiler/Tool Regression or Wrong

ID	R	Tool-Compiler														
		Arch.1-Cl.391	Arch.2-Cl.600	Ins.18-In.1702	Ins.18-In.1802	Ins.18-In.1900	Ins.18-In.1904	Ins.19-In.1702	Ins.19-In.1802	Ins.19-In.1900	Ins.19-In.1904	ROMP-Cl.800	Tsan5-Cl.502	Tsan6-Cl.601	Tsan7-Cl.710	Tsan8-Cl.801
5	Y	✓	✓	✗	✗	✓	✓/✗	✗	✗	✓	✗	✓	✓	✓	✓	✗
6	Y	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓/✗	✓/✗	✓/✗	✗
7	Y	✗	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
8	Y	✗	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
13	Y	✓	✗	✗	✓/✗	✗	✓/✗	✗	✗	✓/✗	✗	✓	✓/✗	✓	✓/✗	✗
23	Y	✓/✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗
24	Y	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
25	Y	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
27	Y	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
29	Y	✓	✓	✓/✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
34	Y	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓/✗	✓/✗	✓/✗	✓
37	Y	RSF	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
39	Y	✓	✓	✗	✓/✗	✓	✗	✗	✗	✓	✗	✓	✓	✓	✓	✓
40	Y	✓	✓	✓/✗	✗	✓	✓/✗	✗	✗	✓	✗	✓	✓/✗	✓/✗	✓/✗	✓/✗
41	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
42	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
43	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
44	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
45	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
46	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
47	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✗	✓	✓	✓	✓
48	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
49	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
50	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
52	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
53	N	RSF	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
54	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
55	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
56	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
57	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
59	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
60	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
61	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓

IDs not shown are benchmarks that are correctly evaluated with every tool.

Results showing Compiler/Tool Regression or Wrong

ID	R	Tool-Compiler														
		Arch.1-Cl.391	Arch.2-Cl.600	Ins.18-In.1702	Ins.18-In.1802	Ins.18-In.1900	Ins.18-In.1904	Ins.19-In.1702	Ins.19-In.1802	Ins.19-In.1900	Ins.19-In.1904	ROMP-Cl.800	Tsan5-Cl.502	Tsan6-Cl.601	Tsan7-Cl.710	Tsan8-Cl.801
62	N	RSF	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
63	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
64	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
65	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	RTO	✓	✓	✓	✓
66	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
67	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
68	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
71	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
72	N	✓	✓	✓/✗	✓/✗	✓	✓/✗	✓	✓	✓	✓/✗	✓	✓	✓	✓/✗	✓/✗
75	Y	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓/✗	✓	✓	✓
80	Y	✓/✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
82	Y	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓/✗	✓/✗	✓/✗	✓/✗
85	N	CSF	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓
86	Y	CSF	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓/✗	✓/✗
87	Y	CSF	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓/✗	✓/✗	✓/✗	✓/✗
91	N	CSF	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓
93	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
94	N	CUN	CUN	CSF	RSF	RSF	✓	CSF	RSF	RSF	✓	CUN	CUN	CUN	CUN	CUN
95	Y	CUN	CUN	✓	✓	✓	✓	✓	✓	✓	✓	CUN	CUN	CUN	CUN	CUN
96	N	CUN	CUN	✓/✗	✓	✗	✓	✓	✓	✗	✓	CUN	CUN	CUN	CUN	CUN
97	N	RSF	✗	✓	✓	✗	✓	✓	✓	✗	✓	CUN	RTO	✓	✓	CUN
99	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
100	N	CUN	CUN	✓	✓	✓	✓	✓	✓	✓	✓	CUN	CUN	CUN	CUN	CUN
102	N	CSF	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
105	N	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓	RSF	✓	✓	✓	✓
106	Y	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	RSF	✓	✓	✓	✓
107	N	✓	✓	✗	✗	✗	✓	✗	✗	✓/✗	✓	✓	✓	✓	✓	✓
108	N	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
110	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✗	✗	✗	✗
112	N	CUN	CUN	CSF	✓	✗	✓	CSF	✓	✗	✓	CUN	CUN	CUN	CUN	CUN
113	N	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
114	Y	✓	✓/✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓/✗	✓	✓	✓/✗
116	Y	RSF	✓	✓	✓	✓	✓	✓	✓	✓	✓	RSF	✓	✓	✓	✓

Dynamic Tools - Configuration Pitfalls

Configuration Pitfalls

1. Archer 2.0:

TP	FN	TN	FP	Configuration
202	20	156	3	`TSAN_OPTIONS` set
213	9	6	153	when environment var. not set (!)

2. Thread Sanitizer:

TP	FN	TN	FP	Configuration
44	13	52	1	with `TSAN_SUPPORT` option
53	5	3	49	when environment var. not set (!)

   - Must use LLVM OpenMP runtime with TSAN-Support option

3. Thread Sanitizer: Gives more FPs with gcc OpenMP library

DRACO: Static Verification Results

Total	TP	TN	FP	FN	TU	FU	Error	Verification Time
116	26	20	0	0	28	33	9	00:14:17

Comparison with Dynamic Tools

- Analysis time
- Unknown vs. multiple test runs
- DRB: contains test case causing the data race
  - already set up for dynamic tools
  - static tools could use this information as well (in our evaluation we did not)
  - more difficult for tools: no input data provided
- DRB 024, 025: benchmarks with SIMD directives - DRACO only!

Conclusion

- Dyn. Tools: Multiple runs necessary for DR detection (increased runtime!)
- Verification Tool: only tool to detect DR in two benchmarks (with SIMD)
- Evaluation with DataRaceBench shows regression of popular tools
- Configuration pitfalls (Archer, ThreadSanitizer)
- Representation of errors/unknown in metrics?

End of Talk

Questions?