Thread Safety for Third Party Libraries in Relaxed Memory Models

Ridhi Jain, IIIT-Delhi, India

Advisor:
Dr. Rahul Purandare, IIIT-Delhi, India
In Collaboration with:
Dr. Subodh Sharma, IIT-Delhi, India

INDRAPRASTHA INSTITUTE *of* INFORMATION TECHNOLOGY **DELHI**

Acknowledgements: Microsoft Research, FormaliSE

Motivation

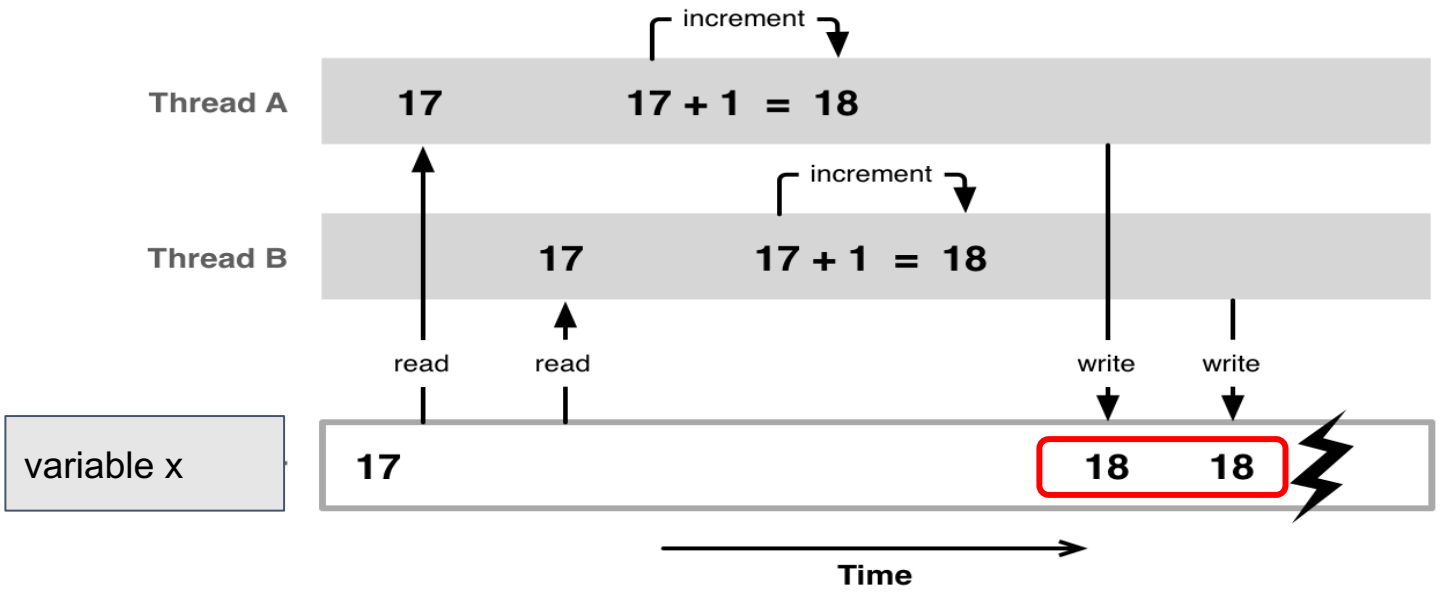
Software crashes can be very expensive

Failure of Mars Pathfinder

Medical Machine Kills, Therac-25

PayPal bug that made a $92 quadrillion deposit

Example



Data Race on variable x

Weak Memory Models

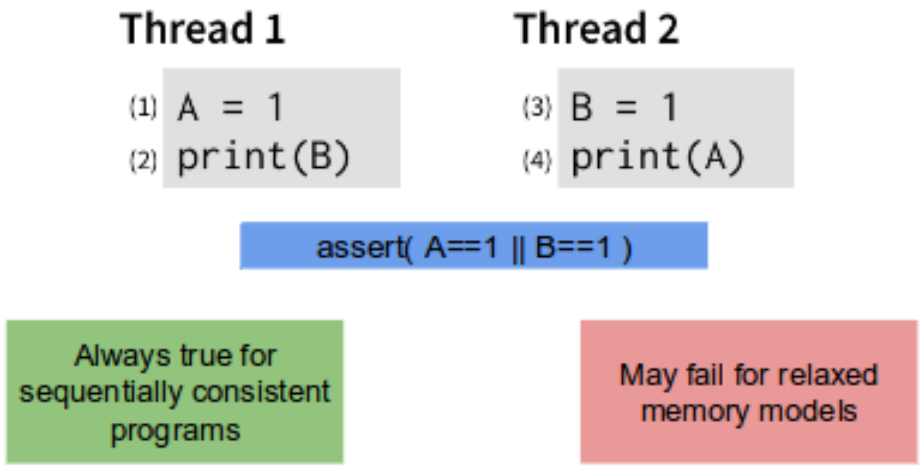
Weaker than Sequentially Consistent memory model

To enhance performance

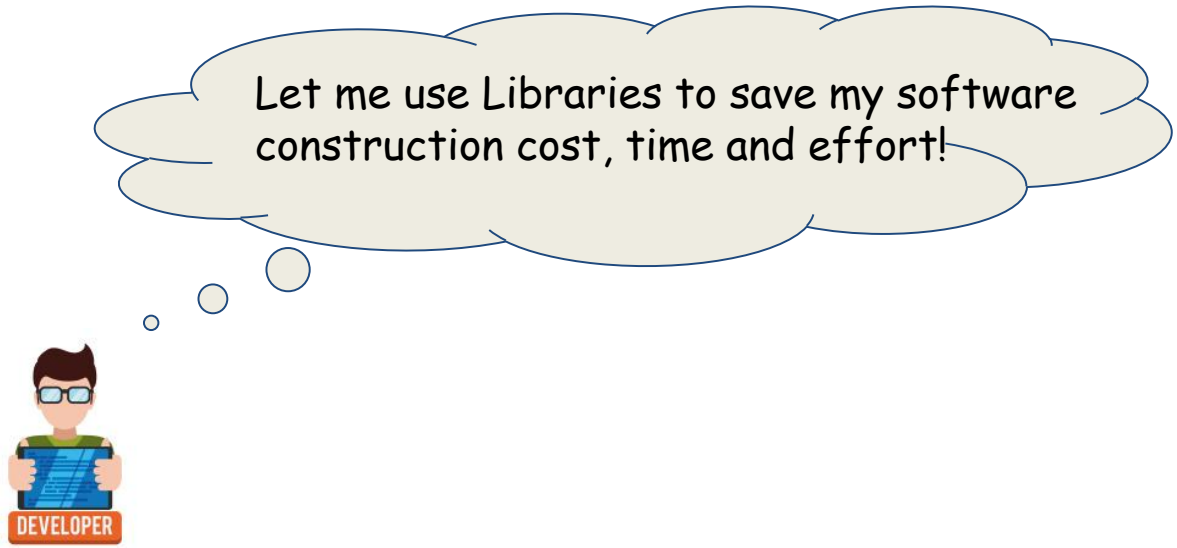
Total Store Order: A store followed by load in a same thread on two different memory locations can be reordered

Partial Store Order: A store followed by load or a store in a same thread on two different memory locations can be reordered

Relaxed Memory

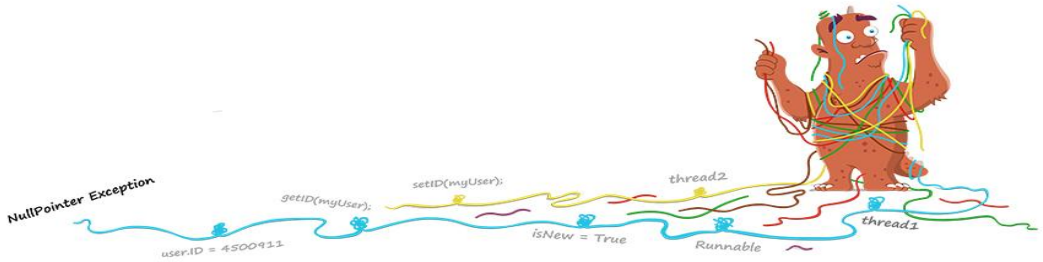


Problem Statement

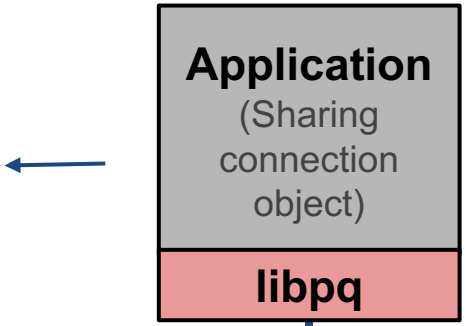


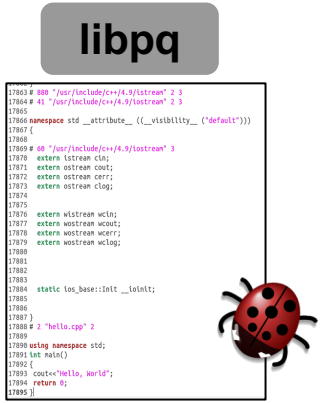
Problem Statement

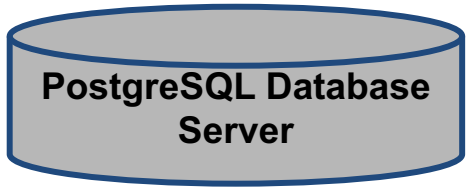
- These third party libraries can be faulty.
- The bugs can be even subtle in a multithreaded environment.
- May be even worse in Relaxed Memory models.



Example


Application
(Sharing
connection
object)
libpq


libpq


PostgreSQL Database
Server

Motivation

libpq: Manipulation of same connection object, leads to a crash

libcurl: Has no internal thread synchronization

libpng: Using same instance of a structure might lead to a crash.

Motivation

Library	Version	Thread Safe?
libcrypt		?
Expat		Yes
FreeTDS		?
FreeType		?
GD 1.8.x		?
GD 2.0.x		?
gdbm		No
ImageMagick	5.2.2	Yes
libjpeg	v6b	?
OpenSSL	0.9.6g	Yes

Figure 1: Thread safety status of sample libraries from Apache HTTP Server 2.x Thread Safety Issues

Research Question

Can we detect the harmful data races in a third party library in weaker memory models?

Questions that Follow

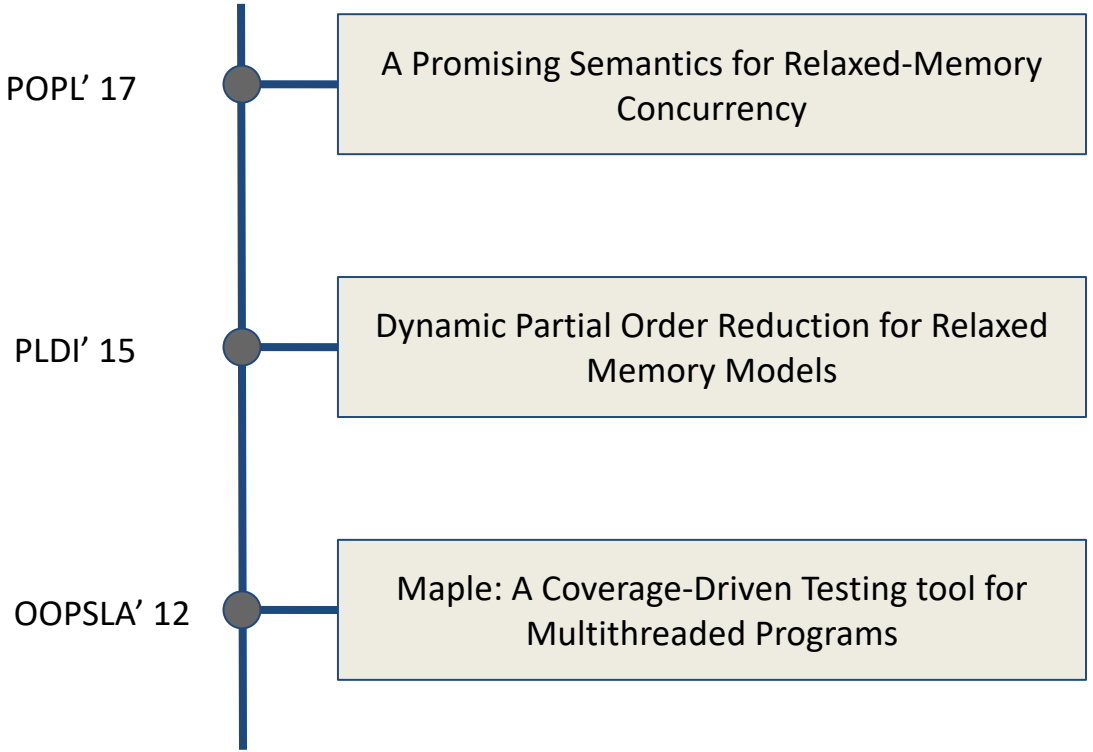
Are all the detected races harmful?

Can we detect data races on stripped binaries?

Can we detect the reorderable instructions in binaries?

Can we reorder stores and loads?

Related Work

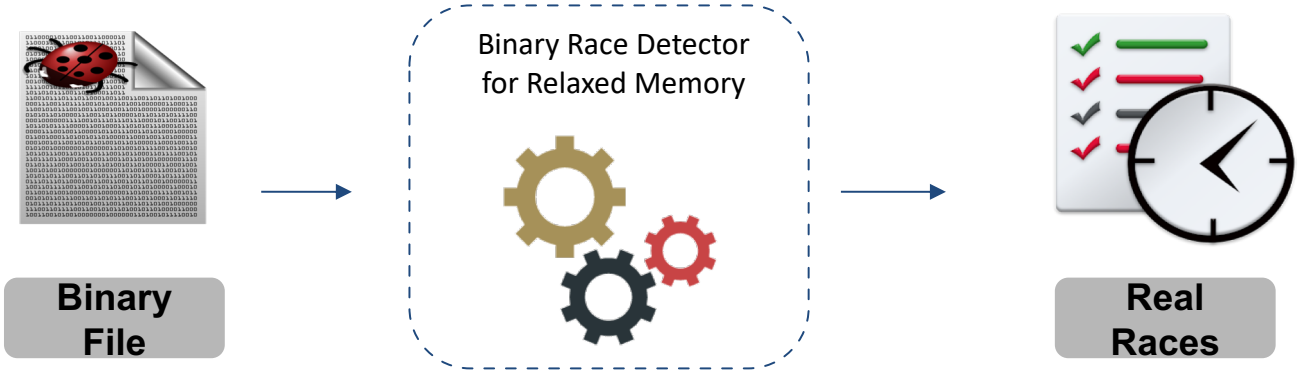


Kang, Jeehoon, et al. "A promising semantics for relaxed-memory concurrency." *ACM SIGPLAN Notices*. Vol. 52. No. 1. ACM, 2017.

Zhang, Naling, Markus Kusano, and Chao Wang. "Dynamic partial order reduction for relaxed memory models." *ACM SIGPLAN Notices*. Vol. 50. No. 6. ACM, 2015.

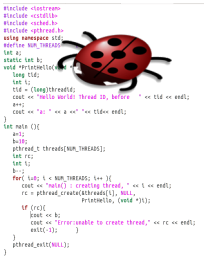
Yu, Jie, et al. "Maple: a coverage-driven testing tool for multithreaded programs." *Acm Sigplan Notices*. Vol. 47. No. 10. ACM, 2012.

System Overview



An Overview of the System

Detection of Shared Variables



Sample Program

Binary File

Variable Addresses

Local Variables and Shared Library Address Filter

Shared Variable Addresses

Address Filter

Detection of Data Races

Software Binary

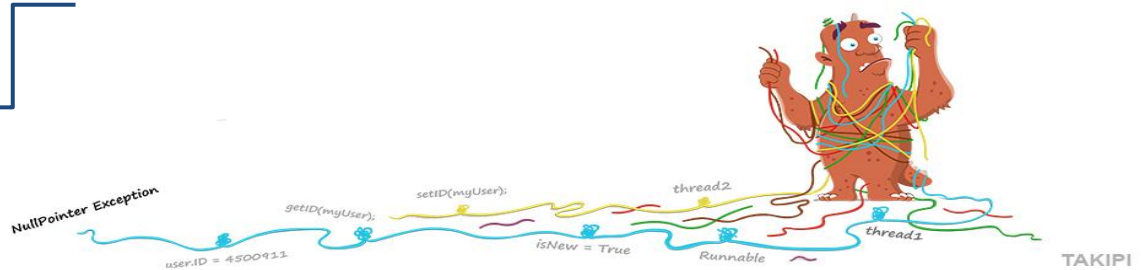
Shared Variable Addresses

Happens Before Relation and Store Buffers

Dynamic Binary Analysis

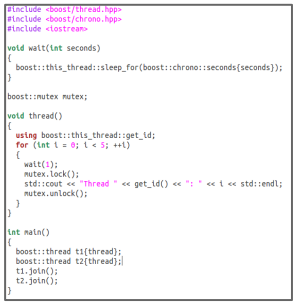
```
1_8_r_{1}_{1}_[]_{2_7_w}
1_9_w_{1}_{1}_[1_10_mov eax, dword ptr [rip+0x2007da]]_{2_9_w}
1_10_r_{1}_{1}_[]_{2_7_w}
2_5_r_{2}_{2}_[]_{3_7_w}
2_7_w_{2}_{2}_[]_{3_5_r,3_7_w,3_8_r,3_10_r}
2_8_r_{2}_{2}_[]_{3_7_w}
2_9_w_{2}_{2}_[]_{3_9_w}
```

Racing Instructions



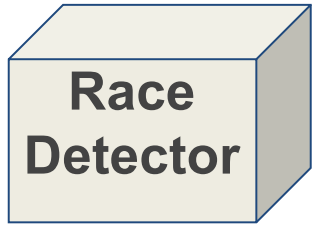
Intermediate Results

- Tested on 16 libraries including boost, libcurl and libgpg.
- Detected data-races where **Intel Thread Checker** detected races, with additional reorderable instructions.



```
#include <boost/thread.hpp>
#include <boost/chrono.hpp>
#include <iostream>

void wait(int seconds)
{
  boost::this_thread::sleep_for(boost::chrono::seconds{seconds});
}

boost::mutex mutex;

void thread()
{
  using boost::this_thread::get_id;
  for (int i = 0; i < 5; ++i)
  {
    wait(1);
    mutex.lock();
    std::cout << "Thread " << get_id() << ": " << i << std::endl;
    mutex.unlock();
  }
}

int main()
{
  boost::thread t1{thread};
  boost::thread t2{thread};
  t1.join();
  t2.join();
}
```

Lib boost example




```
1_8_r_{1}_{1}_[]_{2_7_w}
1_9_w_{1}_{1}_[1_10_mov eax, dword ptr [rip+0x2007da]]_{2_9_w}
1_10_r_{1}_{1}_[]_{2_7_w}
2_5_r_{2}_{2}_[]_{3_7_w}
2_7_w_{2}_{2}_[]_{3_5_r,3_7_w,3_8_r,3_10_r}
2_8_r_{2}_{2}_[]_{3_7_w}
2_9_w_{2}_{2}_[]_{3_9_w}
```

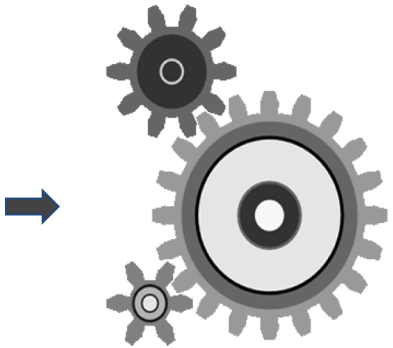
Racing Instructions

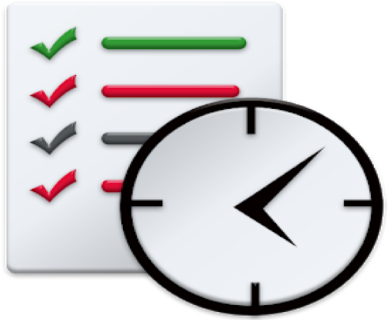
Not all these races are harmful!!

Real Races (Scheduler)

Dynamic Partial Order Reduction is used to to replay the schedules

```
1_8_r_{1}_{1}_[]_{2_7_w}
1_9_w_{1}_{1}_[1_10_mov eax, dword ptr [rip+0x2007da]]_{2_9_w}
1_10_r_{1}_{1}_[]_{2_7_w}
2_5_r_{2}_{2}_[]_{3_7_w}
2_7_w_{2}_{2}_[]_{3_5_r,3_7_w,3_8_r,3_10_r}
2_8_r_{2}_{2}_[]_{3_7_w}
2_9_w_{2}_{2}_[]_{3_9_w}
```





Racing Instructions

Scheduler

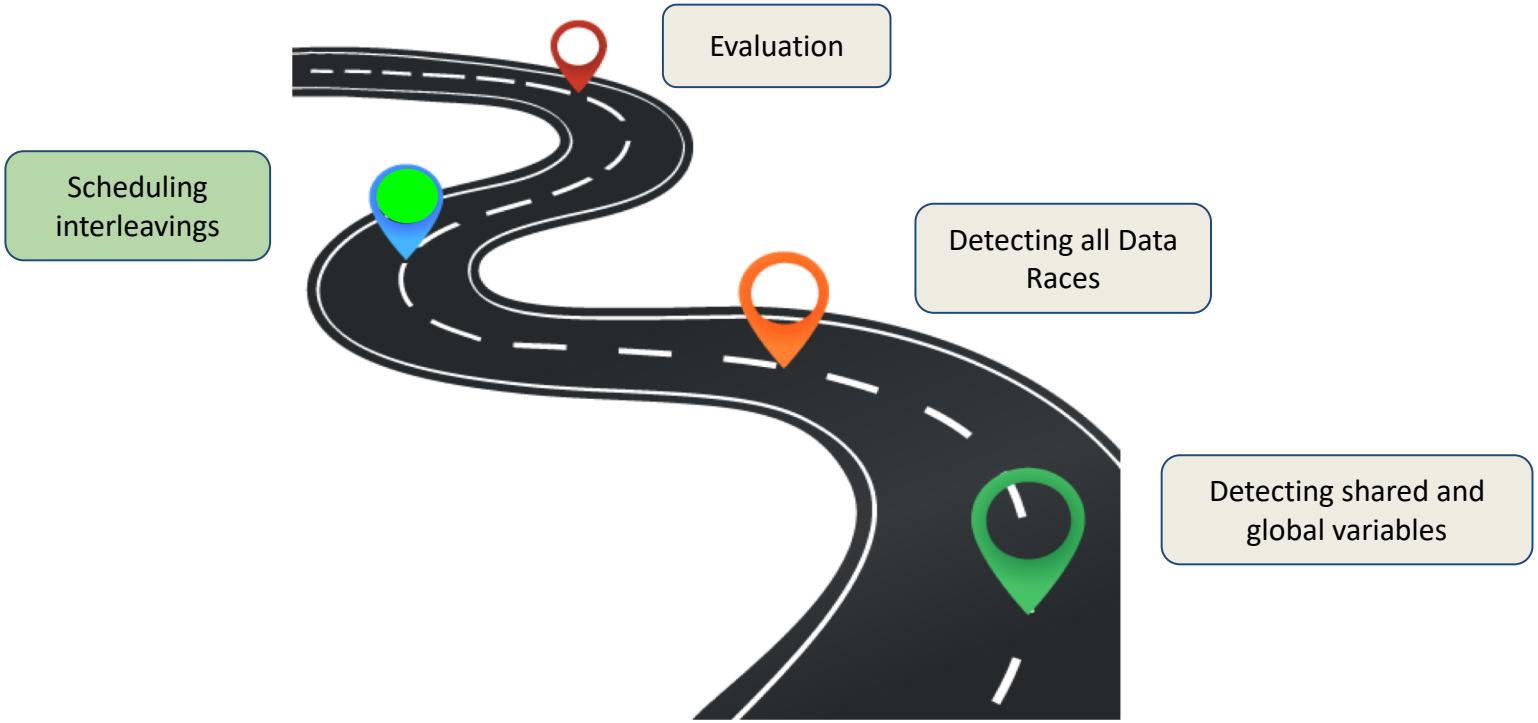
Real Races

Challenges

Binaries are to be analysed and dynamically controlled.

Debug information might not be available.

The implementation of application invoking library may be faulty.


Evaluation
Scheduling interleavings
Detecting all Data Races
Detecting shared and global variables

Thank You