Data race detection for large OpenMP applications

Ignacio Laguna, Harshitha Menon
Lawrence Livermore National Laboratory

Michael Bentley, Ian Briggs, Pavel Panchekha, Ganesh Gopalakrishnan
University of Utah

Hui Guo, Cindy Rubio González
University of California at Davis

Michael O. Lam
James Madison University

http://fpanalysistools.org/
- Shared memory parallel programming
- Standard for on node parallelism in HPC
Data Race

- When shared data is accessed by two or more threads with no synchronization and at least one access is a write
- Leads to non deterministic behavior, crashes, and incorrect results
- Races are undefined behavior in C/C++
- Even “benign” races can be transformed by compilers into harmful races

<table>
<thead>
<tr>
<th>Thread 1</th>
<th>Thread 2</th>
<th>Value of X</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>local = X</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>local++</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Local = X</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>X = local</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>local++</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>X = local</td>
<td>1</td>
</tr>
</tbody>
</table>

http://fpanalysistools.org/
Archer Tool Flow

Static Analysis (OpenMP C/C++ Clang/LLVM Compiler)

Call graph

Find Functions within OpenMP Regions (recursively locates functions within ompOutlined blocks)

Data Dependency Analysis Pass (obtains DD info through Polly; returns dependent loads and stores)

Sequential Code Detection Pass (returns all loads and stores not contained within parallel regions)

Loads / Stores Blacklist

TSan Instrumentation Pass (instrumented loads / stores not contained in Blacklist)

Dynamic Analysis

Annotated OpenMP Runtime

Executable

TSan Runtime

Data Race Report

http://fpanalysisistools.org/
Static Analysis

- Used to not instrument race free code
- Polly, an llvm polyhedral analysis
  - Used to exclude code with no data dependencies
- A custom llvm pass
  - Used to exclude serial code
  - Excludes loads/stores not reachable from OpenMP

```c
#pragma omp parallel for
for (int i=0; i<N; i++) {
    a[i] = a[i] + 1;
}
```

No data dependency code blacklisted

```c
#pragma omp parallel for
for (int i=0; i<N-1; i++) {
    a[i] = a[i+1];
}
```

Potentially racy code instrumented
Archer Tool Flow

Static Analysis (OpenMP C/C++ Clang/LLVM Compiler)

- Call graph
- Find Functions within OpenMP Regions (recursively locates functions within ompOutlined blocks)
- Data Dependency Analysis Pass (obtains DD info through Polly; returns dependent loads and stores)
- Sequential Code Detection Pass (returns all loads and stores not contained within parallel regions)
- TSan Instrumentation Pass (instrument loads / stores not contained in Blacklist)
- Loads / Stores Blacklist

Dynamic Analysis

- Annotated OpenMP Runtime
- Executable
- TSan Runtime

OpenMP Source Code

Data Race Report

http://fpanalysistools.org/
Dynamic Analysis

- ThreadSanitizer is used for dynamic analysis
  - Works well for Pthreads, but breaks on OpenMP constructs
- Annotated OpenMP runtime to highlight synchronization
Results

- AMG2013, algebraic multigrid solver for linear systems
  - Three unknown races found
- HYDRA, a simulator used at the National Ignition Facility
  - Found race which caused crashes when porting to Sequoia
  - First solution was to disable OpenMP
- Z-Pinch, a high-density physics code
  - Data race in macro extremely hard to pinpoint
  - Same value written in the same shared variable
  - IBM XL compiler transformations make the program crash
Examples
Exercise 1: Using Archer on an example from DataRaceBench

- OpenMP data race benchmark suite developed at LLNL
- Available at https://github.com/LLNL/dataracebench
Exercise 1

```c
const int len = 100;
int x[len];
int numNodes = len, numNodes2 = 0;
for (int i=0; i<len; i++) {
    if (i%2 == 0) {
        x[i] = 5;
    } else {
        x[i] = -5;
    }
}
#pragma omp parallel for
for (int i=numNodes-1; i > -1; --i) {
    if (x[i] <= 0) {
        numNodes2--;
    }
}
if (numNodes2 != -numNodes/2) {
    printf("numNodes2 = %d\n", numNodes2);
    //printf("A race occurred\n");
} else {
    //printf("No race found\n");
}
```

Just because no race was found doesn't mean that there is no race! It just means that we were “lucky”. Also these prints are only for illustration (not recommended) — Sutter’s “Pink Elephant” values!!
Exercise 1 - ./step-01.sh

```
exercise-1 $ clang -g -fopenmp minusminus-orig-yes.c \
    -o minusminus-without-archer
exercise-1 $ ./minusminus-without-archer
exercise-1 $ ./minusminus-without-archer
exercise-1 $ ./minusminus-without-archer
exercise-1 $ ./minusminus-without-archer
exercise-1 $ ./minusminus-without-archer
exercise-1 $ ./minusminus-without-archer
exercise-1 $ ./minusminus-without-archer
exercise-1 $ ./minusminus-without-archer
exercise-1 $ ./minusminus-without-archer
[...]
```
Exercise 1 - ./step-02.sh

```sh
exercise-1 $ clang-archer -g minusminus-orig-yes.c -o minusminus-with-archer
exercise-1 $ ./minusminus-with-archer

==================
WARNING: ThreadSanitizer: data race (pid=29573)
  Write of size 4 at 0x7ffdab010858 by thread T1:
    #0 .omp_outlined._debug__
    ~/Module-Archer/exercise-1/minusminus-orig-yes.c:72:16
    (minusminus-with-archer+0x4a7cb1)
    #1 .omp_outlined. ~/Module-Archer/exercise-1/minusminus-orig-yes.c:70:3
    (minusminus-with-archer+0x4a7e02)
    #2 __kmp_invoke_microtask <null> (libomp.so+0x89cc2)

Previous write of size 4 at 0x7ffdab010858 by main thread:
    #0 .omp_outlined._debug__
    ~/Module-Archer/exercise-1/minusminus-orig-yes.c:72:16
    (minusminus-with-archer+0x4a7cb1)
    #1 .omp_outlined. ~/Module-Archer/exercise-1/minusminus-orig-yes.c:70:3
    (minusminus-with-archer+0x4a7e02)
    [...]```
Exercise 1

```c
const int len = 100;
int x[len];
int numNodes = len, numNodes2 = 0;

for (int i=0; i<len; i++) {
    if (i%2 == 0) {
        x[i] = 5;
    } else {
        x[i] = -5;
    }
}

#pragma omp parallel for
for (int i=numNodes-1; i > -1; --i) {
    if (x[i] <= 0) {
        numNodes2-- ;
    }
}

if (numNodes2 != -numNodes/2) {
    printf ("numNodes2 = %d\n", numNodes2);
    //printf ("A race occured\n");
} else {
    //printf ("No race found\n");
}

// Again the prints are for illustration.  // Racy code has “catch-fire” semantics.  // So don’t rely upon printing after a race!  // Pink Elephant values!
```
Exercise 2 Application: LULESH

- Proxy application developed at LLNL
- Models a shock hydrodynamics problem
Exercise 2 - ./step-03.sh

```bash
exercise-2 $ mkdir build
exercise-2 $ cd build
exercise-2 $ cmake ../LULESH -DCMAKE_BUILD_TYPE=Release \
    -DCMAKE_CXX_COMPILER=`which clang-archer++` \
    -DWITH_MPI=Off \
    -DWITH_OPENMP=On
```

-- The CXX compiler identification is Clang 8.0.1
-- Check for working CXX compiler:
/home/ibriggs/ARCHER/archer_install/bin/clang-archer++
-- Check for working CXX compiler:
/home/ibriggs/ARCHER/archer_install/bin/clang-archer++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found OpenMP_CXX: -fopenmp=libomp (found version "3.1")
-- Found OpenMP: TRUE (found version "3.1")
[...]
Exercise 2 - ./step-03.sh

```
exercise-2 $ make
Scanning dependencies of target lulesh2.0
[ 16%] Building CXX object CMakeFiles/lulesh2.0.dir/lulesh-comm.cc.o
[ 33%] Building CXX object CMakeFiles/lulesh2.0.dir/lulesh-init.cc.o
[ 50%] Building CXX object CMakeFiles/lulesh2.0.dir/lulesh-util.cc.o
[ 66%] Building CXX object CMakeFiles/lulesh2.0.dir/lulesh-viz.cc.o
[ 83%] Building CXX object CMakeFiles/lulesh2.0.dir/lulesh.cc.o
[100%] Linking CXX executable lulesh2.0
[100%] Built target lulesh2.0
```
Running problem size $8^3$ per domain until completion

Num processors: 1
Num threads: 3
Total number of elements: 512

Run completed:

- Problem size = 8
- MPI tasks = 1
- Iteration count = 163
- Final Origin Energy = $1.788182e+04$
- Testing Plane 0 of Energy Array on rank 0:
  - MaxAbsDiff = $1.136868e-12$
  - TotalAbsDiff = $1.120390e-11$
  - MaxRelDiff = $5.521293e-14$

Elapsed time = 1.5 (s)
Grind time (us/z/c) = 17.881075 (per dom) (1.492283 overall)
FOM = 55.925049 (z/s)
Exercise 3 Application: LULESH

- Proxy application developed at LLNL
- Models a shock hydrodynamics problem

Goal: find a race condition in a modified Lulesh
Exercise 3 - ./step-04.sh

```bash
exercise-3 $ mkdir build
exercise-3 $ cd build
exercise-3 $ cmake ../LULESH -DCMAKE_BUILD_TYPE=Release \
    -DCMAKE_CXX_COMPILER=`which clang-archer++` \
    -DWITH_MPI=Off \n    -DWITH_OPENMP=On

-- The CXX compiler identification is Clang 8.0.1
-- Check for working CXX compiler:
/home/ibriggs/ARCHER/archer_install/bin/clang-archer++
-- Check for working CXX compiler:
/home/ibriggs/ARCHER/archer_install/bin/clang-archer++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found OpenMP_CXX: -fopenmp=libomp (found version "3.1")
-- Found OpenMP: TRUE (found version "3.1")
[...]
```
Exercise 2 - ./step-04.sh

```
exercise-3 $ make
Scanning dependencies of target lulesh2.0
[ 16%] Building CXX object CMakeFiles/lulesh2.0.dir/lulesh-comm.cc.o
[ 33%] Building CXX object CMakeFiles/lulesh2.0.dir/lulesh-init.cc.o
[ 50%] Building CXX object CMakeFiles/lulesh2.0.dir/lulesh-util.cc.o
[ 66%] Building CXX object CMakeFiles/lulesh2.0.dir/lulesh-viz.cc.o
[ 83%] Building CXX object CMakeFiles/lulesh2.0.dir/lulesh.cc.o
[100%] Linking CXX executable lulesh2.0
[100%] Built target lulesh2.0
```
Exercise 3 - ./step-04.sh

```bash
eExercise-3 $ ./lulesh2.0
Running problem size 8^3 per domain until completion
[...]

WARNING: ThreadSanitizer: data race (pid=28978)
Write of size 8 at 0x7b880000e730 by thread T1:
  #0 .omp_outlined._debug__.33
  ~/Module-Archer/exercise-3/LULESH/lulesh.cc:983:37 (lulesh2.0+0x4c9eea)
  #1 .omp_outlined..34 ~/Module-Archer/exercise-3/LULESH/lulesh.cc:970
   (lulesh2.0+0x4c9eea)
   #2 __kmp_invoke_microtask <null> (libomp.so+0x89cc2)

Previous write of size 8 at 0x7b880000e730 by thread T2:
  #0 .omp_outlined._debug__.33
  ~/Module-Archer/exercise-3/LULESH/lulesh.cc:983:37 (lulesh2.0+0x4c9eea)
  #1 .omp_outlined..34 ~/Module-Archer/exercise-3/LULESH/lulesh.cc:970
   (lulesh2.0+0x4c9eea)
   #2 __kmp_invoke_microtask <null> (libomp.so+0x89cc2)
[...]
```

http://fpanalysistools.org/
Exercise 3

```c
#pragma omp parallel for firstprivate(numNode)
for (Index_t gnode=0 ; gnode<numNode ; ++gnode )
{
    Index_t count = domain.nodeElemCount(gnode) ;
    Index_t *cornerList = domain.nodeElemCornerList(gnode) ;
    Real_t fx_tmp = Real_t(0.0) ;
    Real_t fy_tmp = Real_t(0.0) ;
    Real_t fz_tmp = Real_t(0.0) ;
    for (Index_t i=0 ; i < count ; ++i) {
        Index_t ielem = cornerList[i] ;
        fx_tmp += fx_elem[ielem] ;
        fy_tmp += fy_elem[ielem] ;
        fz_tmp += fz_elem[ielem] ;
    }
    domain.fx(gnode + gnode%2) += fx_tmp ;
    domain.fy(gnode) += fy_tmp ;
    domain fz(gnode) += fz_tmp ;
}
```
Thank You!

Questions?

pruners.github.io/archer