

OpenMP Runtime Error Detection with ARCHER

At the 22nd VI-HPS Tuning Workshop

Joachim Protze, Simone Atzeni
RWTH Aachen University, University of Utah
July 2016



Data race example in OpenMP

```
static double farg1, farg2;  
#define FMAX(a, b) (farg1=(a), farg2=(b), farg1>farg2?farg1:farg2)
```

What could possibly go wrong?

To avoid side effects, the arguments are copied to temporary storage

Double checked scoping of variables: everything seems to be fine

```
1619: #pragma omp parallel for ordered(bar, foo, THRESH)  
1620: for (x=0; x<1000; x++)  
1621:   T = FMAX(0.1111*foo*bar[x], THRESH);
```

Tool flags a write-write race in line 1621

What could possibly go wrong?

Threaded Applications (OpenMP)

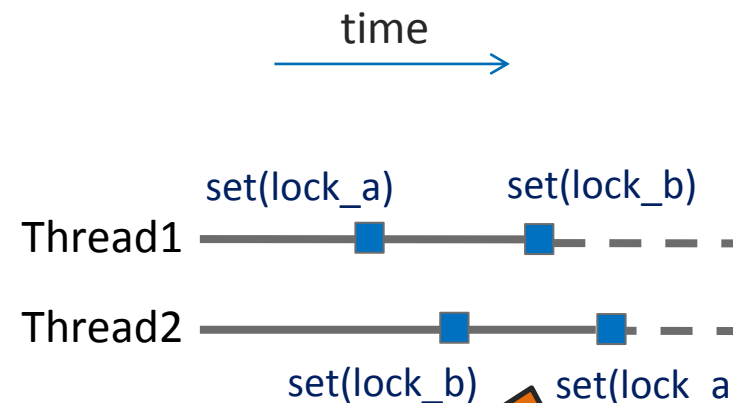
Threaded Defects



Threaded Applications (OpenMP) Threaded Defects – Deadlock

A circular wait condition exists in the system that causes two or more parallel units to wait indefinitely

```
#pragma omp parallel sections
{
    #pragma omp section
    {
        omp_set_lock(&lock_a);
        omp_set_lock(&lock_b);
        omp_unset_lock(&lock_b);
        omp_unset_lock(&lock_a);
    }
    #pragma omp section
    {
        omp_set_lock(&lock_b);
        omp_set_lock(&lock_a);
        omp_unset_lock(&lock_a);
        omp_unset_lock(&lock_b);
    }
}
```



- Thread 1 waits for lock_b owned by thread 2
- Thread 2 waits for lock_a, owned by Thread 1.
- Neither thread can free a lock and both threads wait indefinitely.

Threaded Applications (OpenMP) Threaded Defects – Data Race

Program behavior dependent on execution order of threads/processes

```
int x,y;
#pragma omp parallel
{
    x = omp_get_thread_num ();
    #pragma omp barrier
    #pragma omp master
    printf ("Master is:%d" ,x);
}
```

A write-write race on x

```
int x,y;
#pragma omp parallel
{
    #pragma omp master
    sleep(5);
    x = omp_get_thread_num ();
    #pragma omp barrier
    #pragma omp master
    printf ("Master is:%d" ,x);
}
```

If the master thread is intended to write x, it will usually do so, due to the sleep; But sometimes it may not ...

Threaded Applications (OpenMP)

Definitions

Data race

- Two threads access the same shared variable
 - at least one thread modifies the variable
 - the accesses are concurrent, i.e. unsynchronized
- Leads to non-deterministic behavior
- Hard to find with traditional debugging tools

Deadlock

- Two or more threads are waiting for each other to release locks while holding the lock the other leads to non-deterministic behavior
- Program hangs
- May be non-deterministic

Threaded Applications (OpenMP)

Archer

- Error checking tool for
 - Memory errors
 - **Threading errors**
(OpenMP, Pthreads)
- Based on ThreadSanitizer (runtime check)
- Available for Linux, Windows and Mac
- Supports C, C++ (Fortran in work)
- Modified OpenMP runtime improved for data race detection
- More info: <https://github.com/PRUNER/archer>

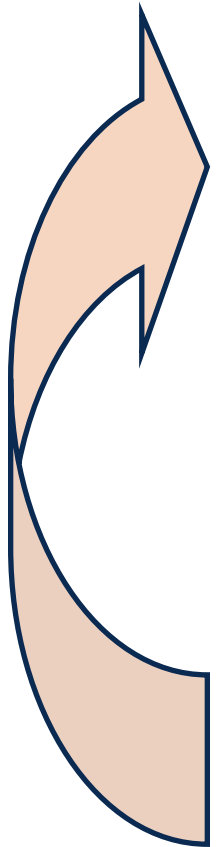


Threaded Applications (OpenMP)

Archer – Background

- Static Analysis
 - Only for OpenMP programs
 - Exclude race free regions and sequential code from runtime analysis to reduce overhead
- Runtime check
 - Error detection only in software branches that are executed
- Low runtime overhead
 - Roughly 2x - 20x
 - Detect races in large OpenMP applications
 - No false positives
- Compiler instrumentation
 - Slower compilation process (apply different passes on the source code to identify race free regions of code, instruments only the rest)

Threaded Applications (OpenMP) Archer – Usage



- Compile the program with the `-g` compiler flag
 - `clang-archer myprog.c -o myprog`
- Run the program under control of ARCHER Runtime
 - `export OMP_NUM_THREADS=...`
`./myprog`
 - Detects problems only in software branches that are executed
- Understand and correct the threading errors detected
- Edit the source code
- Repeat until no errors reported

Threaded Applications (OpenMP)

Archer – Result Summary

```
1  #include <stdio.h>
2
3  int main(int argc, char **argv) {
4      int a = 0;
5      #pragma omp parallel
6      {
7          if (a < 100) {
8              #pragma omp critical
9              a++;
10         }
11     }
12 }
```

WARNING: ThreadSanitizer: data race

Read of size 4 at 0x7fffffffddcd by thread T2:

```
#0 .omp_outlined. race.c:7
(race+0x0000004a6dce)
#1 __kmp_invoke_microtask <null>
(libomp_tsan.so)
```

Previous write of size 4 at 0x7fffffffddcd by main thread:

```
#0 .omp_outlined. race.c:9
(race+0x0000004a6e2c)
#1 __kmp_invoke_microtask <null>
(libomp_tsan.so)
```

Using Archer on darwin during the workshop

```
$ module use /home/hpcprotz1/.modules  
$ module load archer
```

Use NPB-OMP, modify config/make.def to use clang-archer:

Line 78: CC = clang-archer

Build IS or DC:

```
$ make dc CLASS=W  
$ OMP_NUM_THREADS=8 bin/dc.W.x
```

No report means no threading-issue detected 😊

Hands-on

```
$ module use /home/hpcprot1/.modules
$ module load archer

$ cp -r $ARCHER_EXAMPLES archer-examples
$ cd archer-examples
$ clang -fopenmp -g prime_omp.c -lm
```

Try:

```
$ OMP_NUM_THREADS=2 ./a.out
$ OMP_NUM_THREADS=4 ./a.out
$ OMP_NUM_THREADS=8 ./a.out
```

Hands-on 2

- Now compile with data race detection:

```
$ clang-archer -g prime_omp.c
```

```
$ OMP_NUM_THREADS=2 ./a.out
```

Fix the issues, recompile, test again

Better: do this using the batch system

(see provided batch file `$ARCHER_EXAMPLES/archer.batch`)

Fallback

- In cases, where compilation with clang-archer fails:

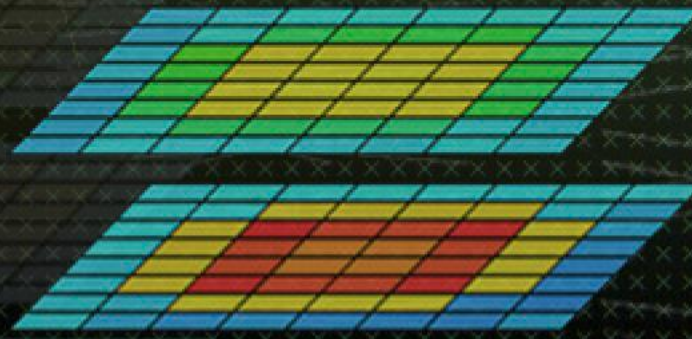
```
$ clang -sanitize=thread -fopenmp -g prime_omp.c
```

```
$ OMP_NUM_THREADS=2 ./a.out
```

For OpenMP programs, use the clang delivered with ARCHER to avoid

Conclusions

- Deadlocks:
 - Avoid locks when possible
 - Prefer critical/master/...
- Races:
 - Often hard to detect, in many cases only visible from time to time
 - Races manifesting only at large scale are often detectable by ARCHER at small scale
 - (Fortran) consider: default(private)
- Use tools to detect defects as early as possible:
 - During development + unit testing
 - Development of ARCHER is ongoing effort, also porting to more architectures and OpenMP runtimes.



Thank You