UNIVERSITY OF MICHIGAN

Dynamic Detection and Mitigation of False Sharing with Intel Pin and LLVM

Brandon Kayes, Daniel Hoekwater, Thomas Smith, Tony Bai

EECS 583 Final Project

Fall 2021 - Group 21

What is false sharing?

- **False sharing:** The different pieces of data from different CPUs end up on the same cache line, so the CPUs must coordinate whenever either piece of data is changed
  - A CPU that writes to a falsely shared cache line must invalidate all other CPUs' caches
  - Other CPUs will incur cache misses when they try to read the falsely shared cache line
- **Our goal**: Identify false sharing through profiling, and eliminate it with compiler transformations
  - <u>References:</u> *Reducing False Sharing on Shared Memory Multiprocessors through Compile Time Data Transformations* by Tor E. Jeremiassen, Susan J. Eggers

Intel Pin

- To aid with profiling, we used a tool called **Intel Pin**
  - https://www.intel.com/content/www/us/en/developer/articles/tool/pin-a-dynamic-binary-instrumentation-tool.html
  - Pin is a dynamic binary instrumentation framework that enables the creation of dynamic program analysis tools.
- The tools only work with a single thread, so we had to modify them to work with multiple threads
- We extended the **pinatrace** tool to output a program's memory accesses along with which thread performed the access
- We extended the **dcache** tool to perform **m**ulticore L1 cache simulator (we named our version **mdcache**)
  - Used for both detection and evaluation

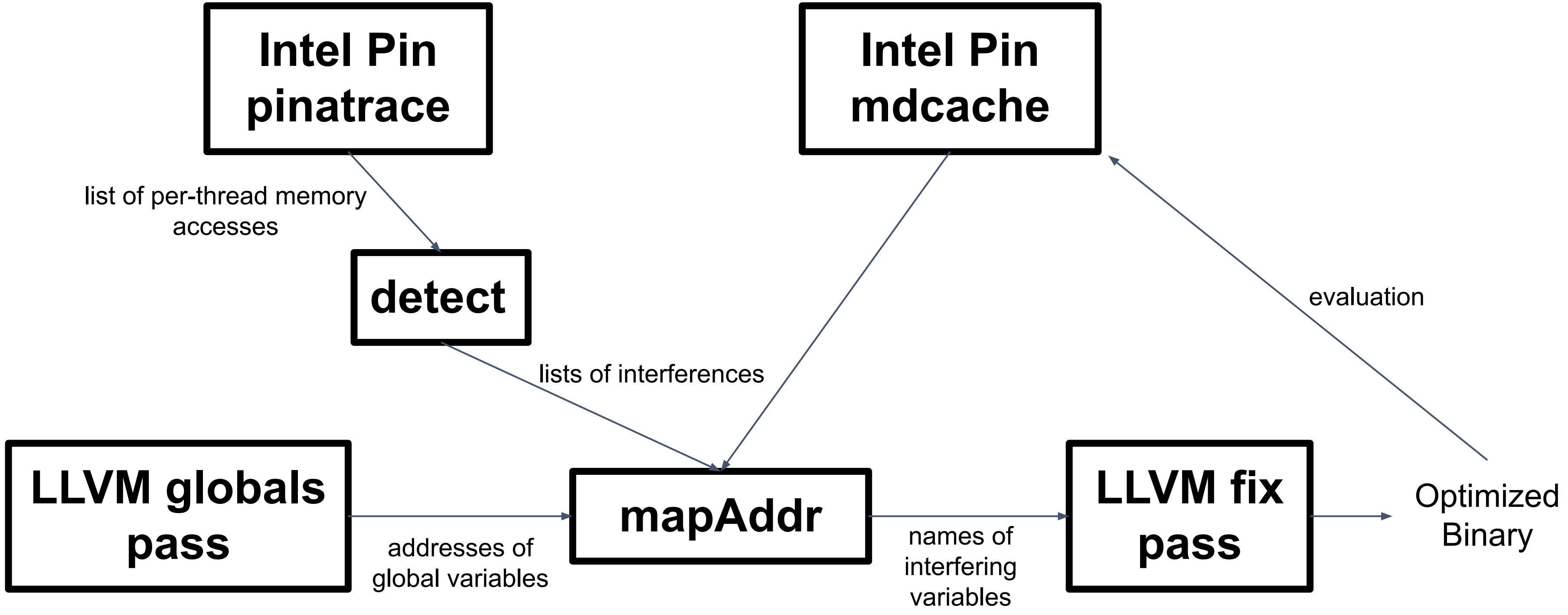
LLVM Passes

- We created two LLVM passes
- **globals pass**
    - A pass to print out the addresses of global variables, so that we can match up the memory accesses outputted by Intel Pin with named variables in our program
- **fix pass**
    - Given the names and offsets of variables which were detected to be false shared, this pass added padding between falsely shared variables to separate the variables out into different cache lines

C++ Utilities

- We also created two C++ utility programs to process the output of one stage into a format that can be fed as input to the next stage
- **detect**
  - Given the memory accesses outputted by Intel Pin's pinatrace, identify which accesses are potentially false shared.
  - Criteria:
    - Accesses from different threads
    - Accesses in the same cache line
    - Accesses non-overlapping addresses
    - At least one of the accesses is a write
  - Output a list of potential interferences
- **mapAddr**
  - Given the lists of interferences outputted by Intel Pin as well as the output from the LLVM globals pass, identify the names of variables which are false shared

Putting it all together

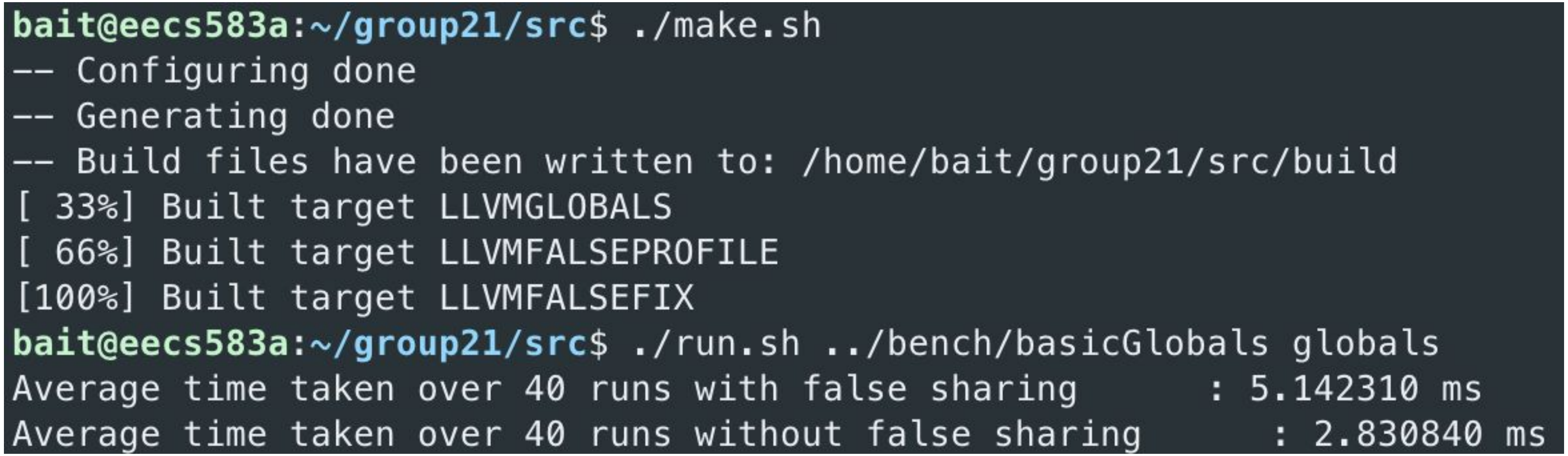


Demo

```
volatile int fsData1 = 0;
volatile int fsData2 = 0;

alignas(64) volatile int data1 = 0;
alignas(64) volatile int data2 = 0;

void runThread(volatile int *threadData)
{
 for (int i = 0; i < NUM_LOOPS; ++i) {
   ++(*threadData);
 }
}
```

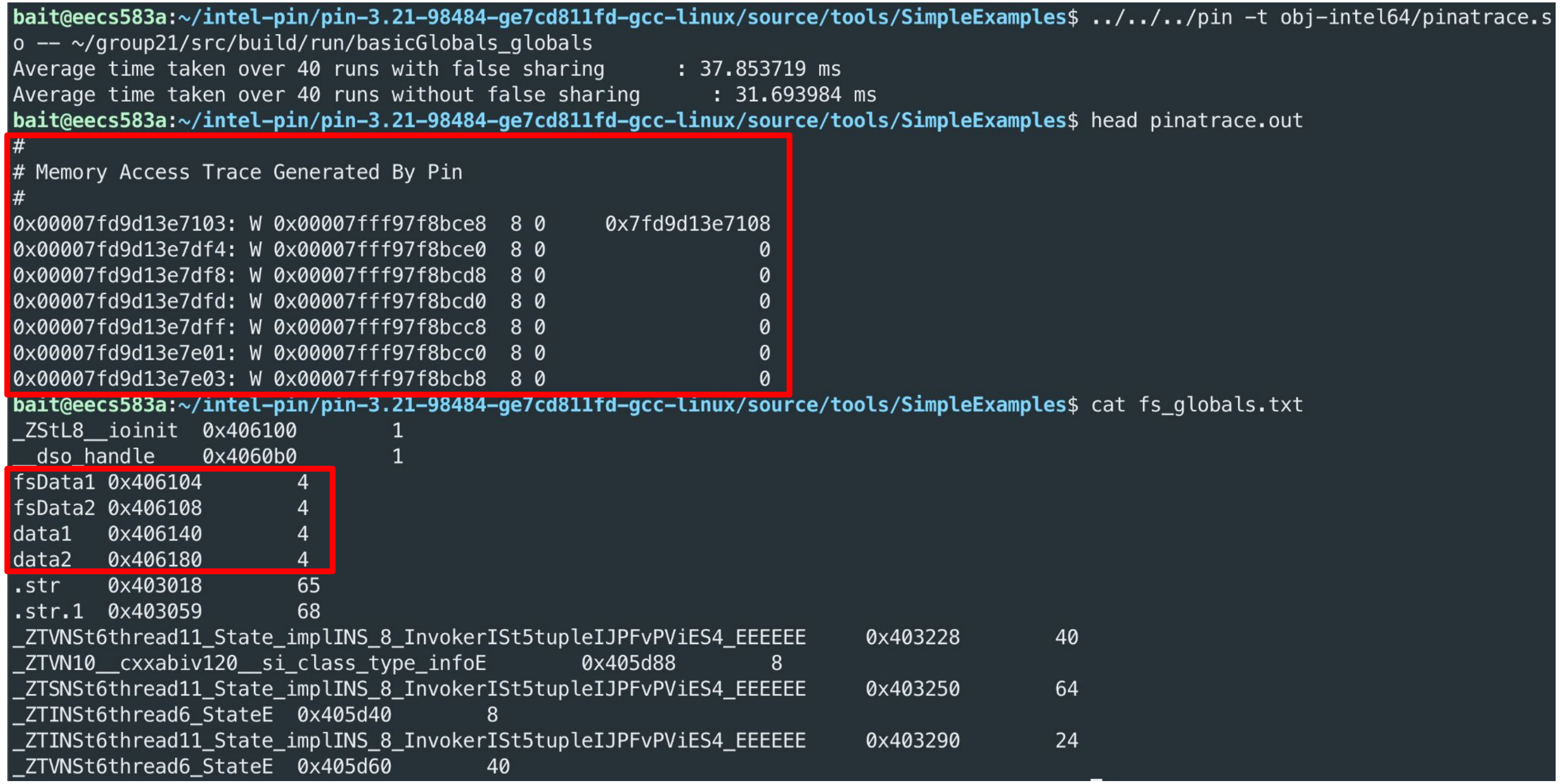
```
int main() {
 for (int i = 0; i < NUM_RUNS; i++) {
   std::thread thread1{runThread, &fsData1};
   std::thread thread2{runThread, &fsData2};
   thread1.join();
   thread2.join();
 }

 for (int i = 0; i < NUM_RUNS; i++) {
   std::thread thread1{runThread, &data1};
   std::thread thread2{runThread, &data2};
   thread1.join();
   thread2.join();
 }
}
```

Finding Potential Interferences



```
bait@eecs583a:~/group21/src$ ./make.sh
-- Configuring done
-- Generating done
-- Build files have been written to: /home/bait/group21/src/build
[ 33%] Built target LLVMGLOBALS
[ 66%] Built target LLVMFALSEPROFILE
[100%] Built target LLVMFALSEFIX
bait@eecs583a:~/group21/src$ ./run.sh ../bench/basicGlobals globals
Average time taken over 40 runs with false sharing      : 5.142310 ms
Average time taken over 40 runs without false sharing      : 2.830840 ms
```

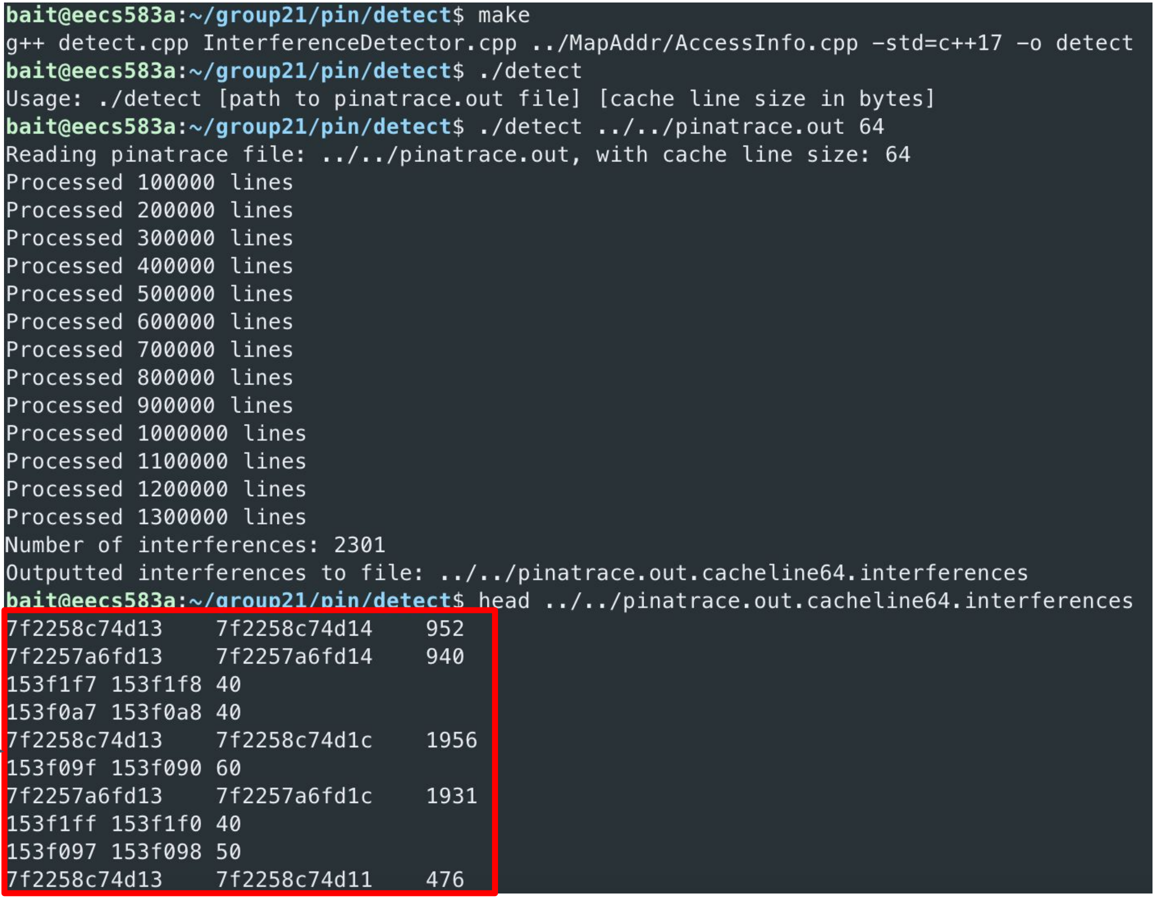
Intel Pin pinatrace



```
bait@eecs583a:~/intel-pin/pin-3.21-98484-ge7cd811fd-gcc-linux/source/tools/SimpleExamples$ ../../../pin -t obj-intel64/pinatrace.s
o -- ~/group21/src/build/run/basicGlobals_globals
Average time taken over 40 runs with false sharing      : 37.853719 ms
Average time taken over 40 runs without false sharing      : 31.693984 ms
bait@eecs583a:~/intel-pin/pin-3.21-98484-ge7cd811fd-gcc-linux/source/tools/SimpleExamples$ head pinatrace.out
#
# Memory Access Trace Generated By Pin
#
0x00007fd9d13e7103: W 0x00007fff97f8bce8  8 0     0x7fd9d13e7108
0x00007fd9d13e7df4: W 0x00007fff97f8bce0  8 0                  0
0x00007fd9d13e7df8: W 0x00007fff97f8bcd8  8 0                  0
0x00007fd9d13e7dfd: W 0x00007fff97f8bcd0  8 0                  0
0x00007fd9d13e7dff: W 0x00007fff97f8bcc8  8 0                  0
0x00007fd9d13e7e01: W 0x00007fff97f8bcc0  8 0                  0
0x00007fd9d13e7e03: W 0x00007fff97f8bcb8  8 0                  0
bait@eecs583a:~/intel-pin/pin-3.21-98484-ge7cd811fd-gcc-linux/source/tools/SimpleExamples$ cat fs_globals.txt
_ZStL8__ioinit  0x406100        1
__dso_handle    0x4060b0        1
fsData1 0x406104        4
fsData2 0x406108        4
data1   0x406140        4
data2   0x406180        4
.str    0x403018        65
.str.1  0x403059        68
_ZTVNSt6thread11_State_implINS_8_InvokerISt5tupleIJPFvPViES4_EEEEEE     0x403228        40
_ZTVN10__cxxabiv120__si_class_type_infoE        0x405d88        8
_ZTSNSt6thread11_State_implINS_8_InvokerISt5tupleIJPFvPViES4_EEEEEE     0x403250        64
_ZTINSt6thread6_StateE  0x405d40        8
_ZTINSt6thread11_State_implINS_8_InvokerISt5tupleIJPFvPViES4_EEEEEE     0x403290        24
_ZTVNSt6thread6_StateE  0x405d60        40
```

Detect

- Given the memory accesses outputted by Intel Pin's pinatrace, identify which accesses are potentially false shared.
- Criteria:
    - **Accesses from different threads**
    - **Accesses in the same cache line**
    - **Accesses non-overlapping addresses**
    - **At least one of the accesses is a write**
- Output a list of potential interferences



```
bait@eecs583a:~/group21/pin/detect$ make
g++ detect.cpp InterferenceDetector.cpp ../MapAddr/AccessInfo.cpp -std=c++17 -o detect
bait@eecs583a:~/group21/pin/detect$ ./detect
Usage: ./detect [path to pinatrace.out file] [cache line size in bytes]
bait@eecs583a:~/group21/pin/detect$ ./detect ../../pinatrace.out 64
Reading pinatrace file: ../../pinatrace.out, with cache line size: 64
Processed 100000 lines
Processed 200000 lines
Processed 300000 lines
Processed 400000 lines
Processed 500000 lines
Processed 600000 lines
Processed 700000 lines
Processed 800000 lines
Processed 900000 lines
Processed 1000000 lines
Processed 1100000 lines
Processed 1200000 lines
Processed 1300000 lines
Number of interferences: 2301
Outputted interferences to file: ../../pinatrace.out.cacheline64.interferences
bait@eecs583a:~/group21/pin/detect$ head ../../pinatrace.out.cacheline64.interferences
7f2258c74d13    7f2258c74d14    952
7f2257a6fd13    7f2257a6fd14    940
153f1f7 153f1f8 40
153f0a7 153f0a8 40
7f2258c74d13    7f2258c74d1c    1956
153f09f 153f090 60
7f2257a6fd13    7f2257a6fd1c    1931
153f1ff 153f1f0 40
153f097 153f098 50
7f2258c74d13    7f2258c74d11    476
```

Intel Pin mdcache: Hit-rate Statistics

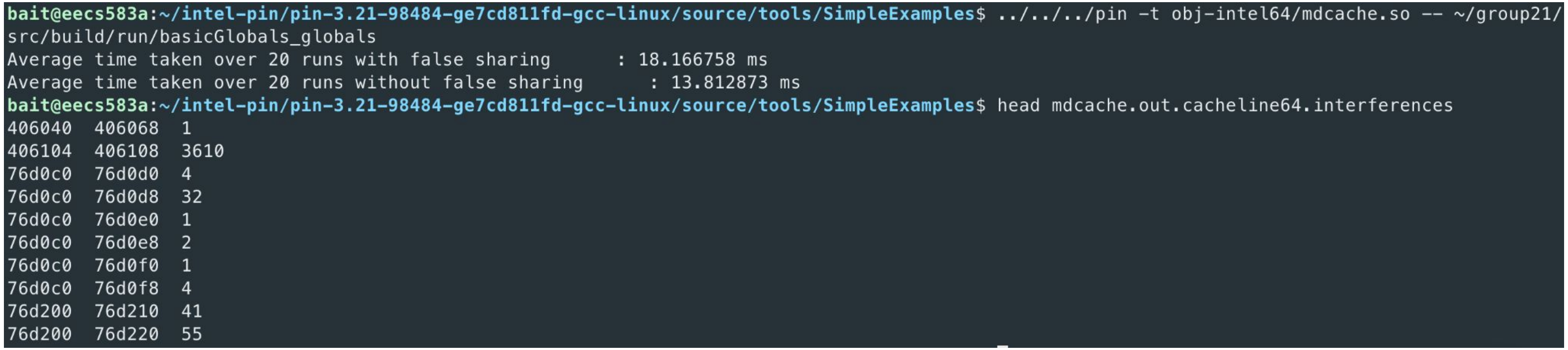
CPU 0 L1 Cache Statistics

Total-Hits	1177097	26.29%
Total-Misses	3297634	73.65%
Total-Tombstones	2439	**0.05%**

CPU 1 L1 Cache Statistics

Total-Hits	75308	2.19%
Total-Misses	2557302	74.44%
Total-Tombstones	802804	**23.37%**

Intel Pin mdcache: False Sharing Detection



```
bait@eecs583a:~/intel-pin/pin-3.21-98484-ge7cd811fd-gcc-linux/source/tools/SimpleExamples$ ../../../pin -t obj-intel64/mdcache.so -- ~/group21/
src/build/run/basicGlobals_globals
Average time taken over 20 runs with false sharing      : 18.166758 ms
Average time taken over 20 runs without false sharing      : 13.812873 ms
bait@eecs583a:~/intel-pin/pin-3.21-98484-ge7cd811fd-gcc-linux/source/tools/SimpleExamples$ head mdcache.out.cacheline64.interferences
406040  406068  1
406104  406108  3610
76d0c0  76d0d0  4
76d0c0  76d0d8  32
76d0c0  76d0e0  1
76d0c0  76d0e8  2
76d0c0  76d0f0  1
76d0c0  76d0f8  4
76d200  76d210  41
76d200  76d220  55
```

pinatrace + Detect vs mdcache

pinatrace + Detect

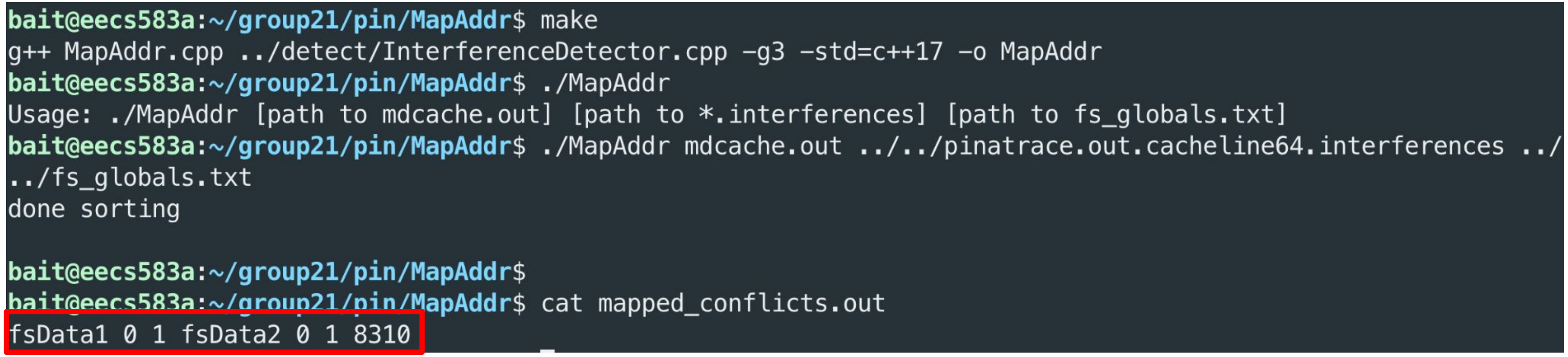
```
priority    source          destination
      34    7f3ac0fb6b88    7f3ac0fb6b90
      30    7f3ac0fdb2c8    7f3ac0fdb2e0
      24    7f3ac0fb6b90    7f3ac0fb6b80
      24    7f3ac0fb6b88    7f3ac0fb6b80
      18    b0b098          b0b0b0
      18    b0b090          b0b0b0
      18    7f3ac0fb9b68    7f3ac0fb9b6c
      18    7f3ac0fb6b90    7f3ac0fb6b88
      18    406104          406108
      14    b0b1e8          b0b1e0
```

mdcache

```
      source   destination   priority
      406104   406108        7
7fe0bb5a8b80   7fe0bb5a8b88  3
7fe0b9c73680   7fe0b9c73698  2
7fe0b9c72e40   7fe0b9c72e4c  2
7fe0b8699a00   7fe0b8699a08  2
7fe0b8698e40   7fe0b8698e4c  2
7fe0bb5a8b88   7fe0bb5a8bb8  1
7fe0bb5a8b88   7fe0bb5a8bb0  1
7fe0bb5a8b88   7fe0bb5a8ba8  1
7fe0bb5a8b88   7fe0bb5a8ba0  1
```

MapAddr

Used to match interfering accesses with global variable names



```
bait@eecs583a:~/group21/pin/MapAddr$ make
g++ MapAddr.cpp ../detect/InterferenceDetector.cpp -g3 -std=c++17 -o MapAddr
bait@eecs583a:~/group21/pin/MapAddr$ ./MapAddr
Usage: ./MapAddr [path to mdcache.out] [path to *.interferences] [path to fs_globals.txt]
bait@eecs583a:~/group21/pin/MapAddr$ ./MapAddr mdcache.out ../../pinatrace.out.cacheline64.interferences ../
../fs_globals.txt
done sorting

bait@eecs583a:~/group21/pin/MapAddr$
bait@eecs583a:~/group21/pin/MapAddr$ cat mapped_conflicts.out
fsData1 0 1 fsData2 0 1 8310
```

Fix

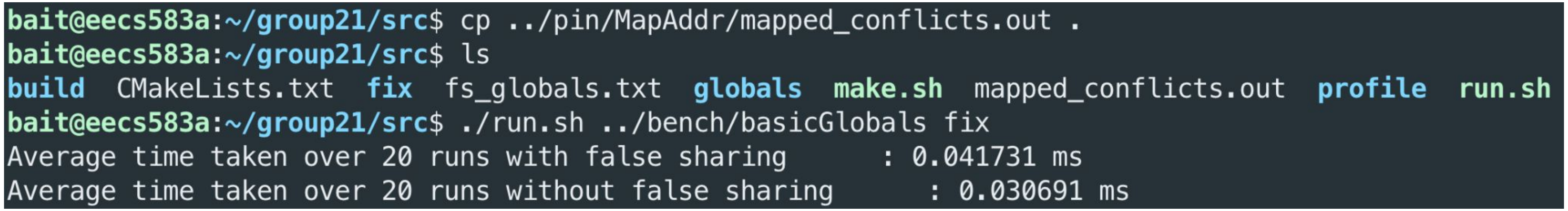
Before:

```
@myGlobalVar =
    dso_local global i32 0,
    align 4


%"struct.MyStruct" =
    type { i32, i32 }


getelementptr inbounds (
    %"struct.MyStruct",
    %"struct.MyStruct"* @data,
    i32 0,
    i32 1
)
```

After:

```
@myGlobalVar =
    dso_local global i32 0,
    align 64


%"struct.MyStruct.0" =
    type { i32, [60 x i8], i32 }


getelementptr inbounds (
    %"struct.MyStruct.0",
    %"struct.MyStruct.0"* @data.1,
    i32 0,
    i32 2
)
```

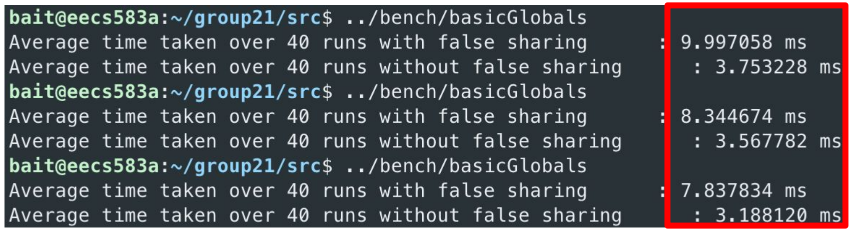
Fix



```
bait@eecs583a:~/group21/src$ cp ../pin/MapAddr/mapped_conflicts.out .
bait@eecs583a:~/group21/src$ ls
build  CMakeLists.txt  fix  fs_globals.txt  globals  make.sh  mapped_conflicts.out  profile  run.sh
bait@eecs583a:~/group21/src$ ./run.sh ../bench/basicGlobals fix
Average time taken over 20 runs with false sharing      : 0.041731 ms
Average time taken over 20 runs without false sharing      : 0.030691 ms
```

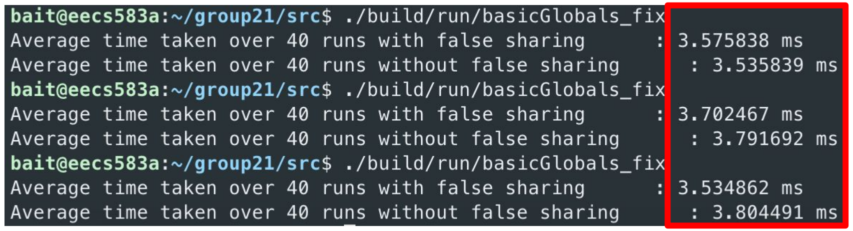
For evaluation purposes, we cache mapped_conflicts.out and ran the program with a higher loop iteration count:

Before:



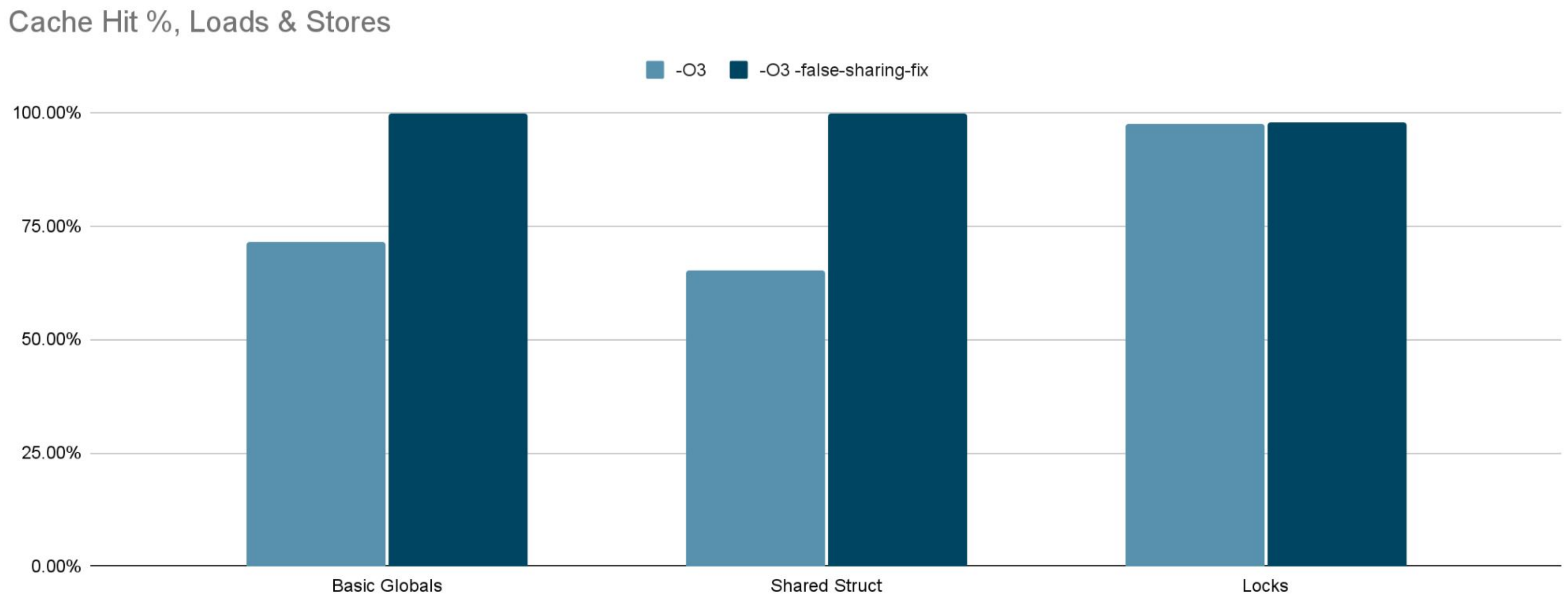
```
bait@eecs583a:~/group21/src$ ../bench/basicGlobals
Average time taken over 40 runs with false sharing      : 9.997058 ms
Average time taken over 40 runs without false sharing      : 3.753228 ms
bait@eecs583a:~/group21/src$ ../bench/basicGlobals
Average time taken over 40 runs with false sharing      : 8.344674 ms
Average time taken over 40 runs without false sharing      : 3.567782 ms
bait@eecs583a:~/group21/src$ ../bench/basicGlobals
Average time taken over 40 runs with false sharing      : 7.837834 ms
Average time taken over 40 runs without false sharing      : 3.188120 ms
```

After:



```
bait@eecs583a:~/group21/src$ ./build/run/basicGlobals_fix
Average time taken over 40 runs with false sharing      : 3.575838 ms
Average time taken over 40 runs without false sharing      : 3.535839 ms
bait@eecs583a:~/group21/src$ ./build/run/basicGlobals_fix
Average time taken over 40 runs with false sharing      : 3.702467 ms
Average time taken over 40 runs without false sharing      : 3.791692 ms
bait@eecs583a:~/group21/src$ ./build/run/basicGlobals_fix
Average time taken over 40 runs with false sharing      : 3.534862 ms
Average time taken over 40 runs without false sharing      : 3.804491 ms
```

Fix


Cache Hit %, Loads & Stores
-O3
-O3 -false-sharing-fix
100.00%
75.00%
50.00%
25.00%
0.00%
Basic Globals
Shared Struct
Locks

Future Improvements

- Broaden the fix pass to work with nested structs, structs passed to functions, etc.
- Add other mitigation techniques (e.g. the group and transpose technique)
- Fix false sharing in other contexts like the stack and the heap
- Generalize all our tools to work with processors other than Intel x86_64 CPUs (different cache line sizes, different word sizes, etc.)
- Optimize our tools to work with larger benchmarks