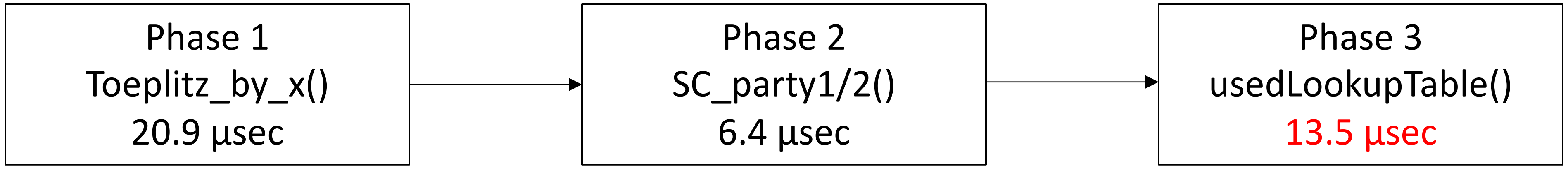
Project Buggy Snail

Problem

- Phase 3(using lookup) took 4.6 µsec in new protocol and OPRF but it takes 13 µsec in Dark matter PRF.
- The following was microbenchmark on dark matter PRF.



Possible Problems

- Something from previous phase/round takes more time and the timing is leaked.
- Phase 1 and Phase 2 runs nRuns(1000) number of times internally* instead of nTimes(=1) times.
- * = This internal run is different from 1000 runs used in timing.

Analysis: Project Buggy Snail

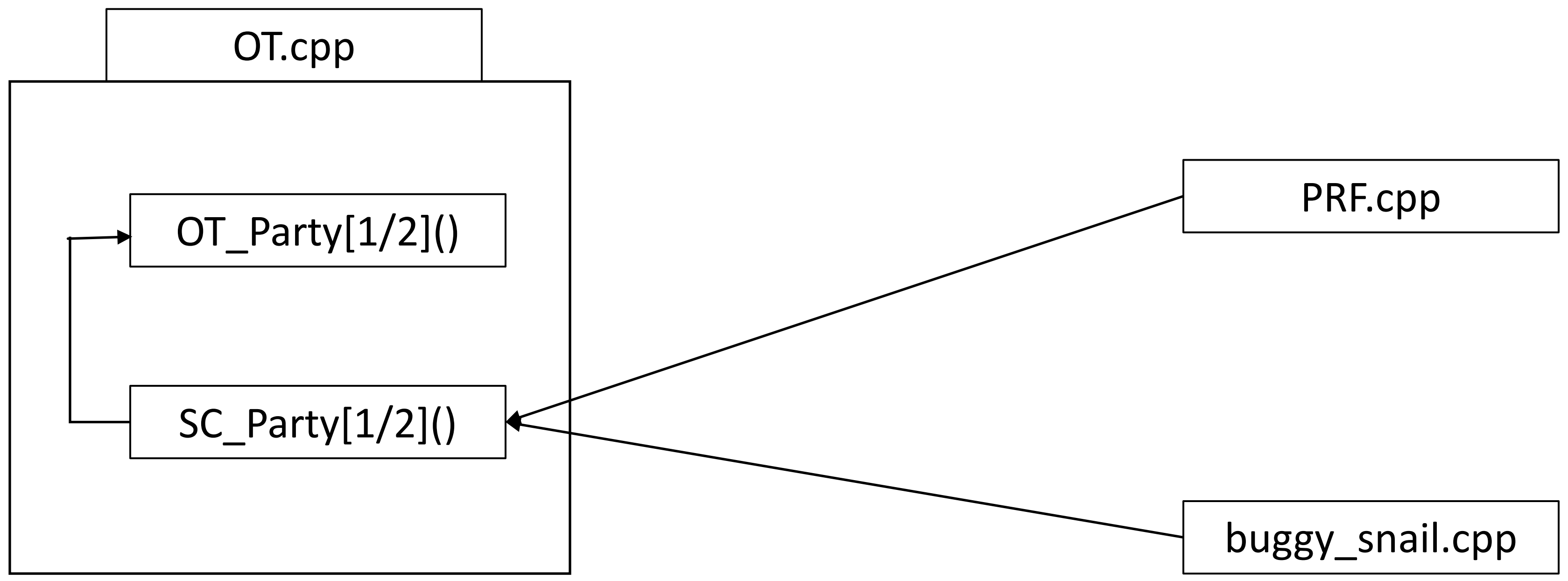
- ✓Replicate PRF.cpp
- ✓Make code modular
- ✓Disable full-length running of each phase and time the protocol.

Aim: To find if running of any phase/round affects the timing for phase3(lookup table)

Removal of phase*	Change in timing of phase 3
Removed phase 1	13.5 → 10.5
Removed phase 1 and phase 2	13.5 → **3.89**
Removed phase 2	13.5 → **3.67**

Some solutions tried

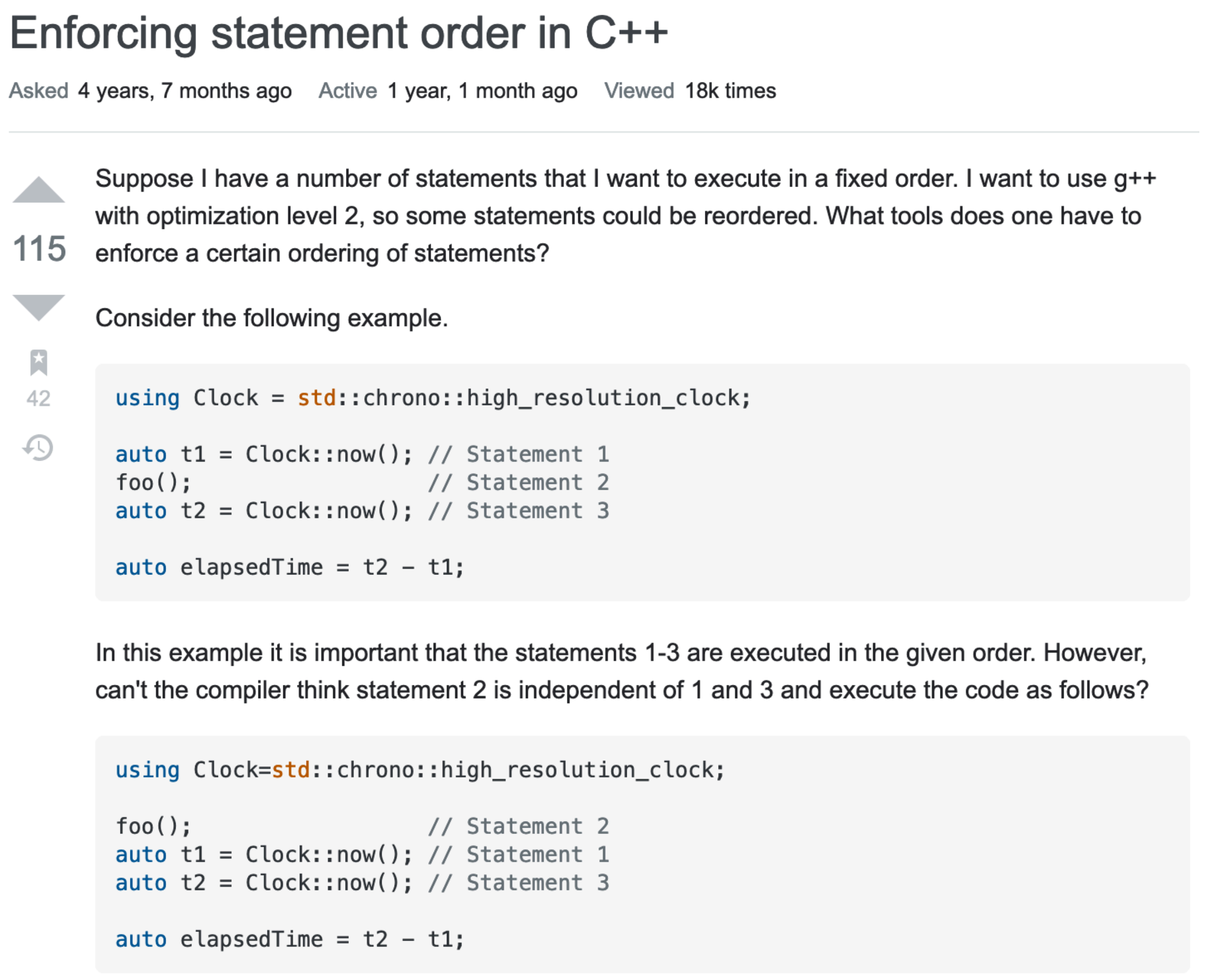
- Time phase 2 on outer level, earlier it was inside OT.cpp – FAILED!



The Culprit

- -O3 flag was optimizing the code to a point where reordering was done.
- Wouldn't show up in debugging.
- This was the order(suspected)

Start phase3 time
|
Complete phase 2
|
Complete phase 3
|
Stop phase 3 timer



Enforcing statement order in C++

Asked 4 years, 7 months ago Active 1 year, 1 month ago Viewed 18k times

115

42

Suppose I have a number of statements that I want to execute in a fixed order. I want to use g++ with optimization level 2, so some statements could be reordered. What tools does one have to enforce a certain ordering of statements?

Consider the following example.

```
using Clock = std::chrono::high_resolution_clock;

auto t1 = Clock::now(); // Statement 1
foo();                  // Statement 2
auto t2 = Clock::now(); // Statement 3

auto elapsedTime = t2 - t1;
```

In this example it is important that the statements 1-3 are executed in the given order. However, can't the compiler think statement 2 is independent of 1 and 3 and execute the code as follows?

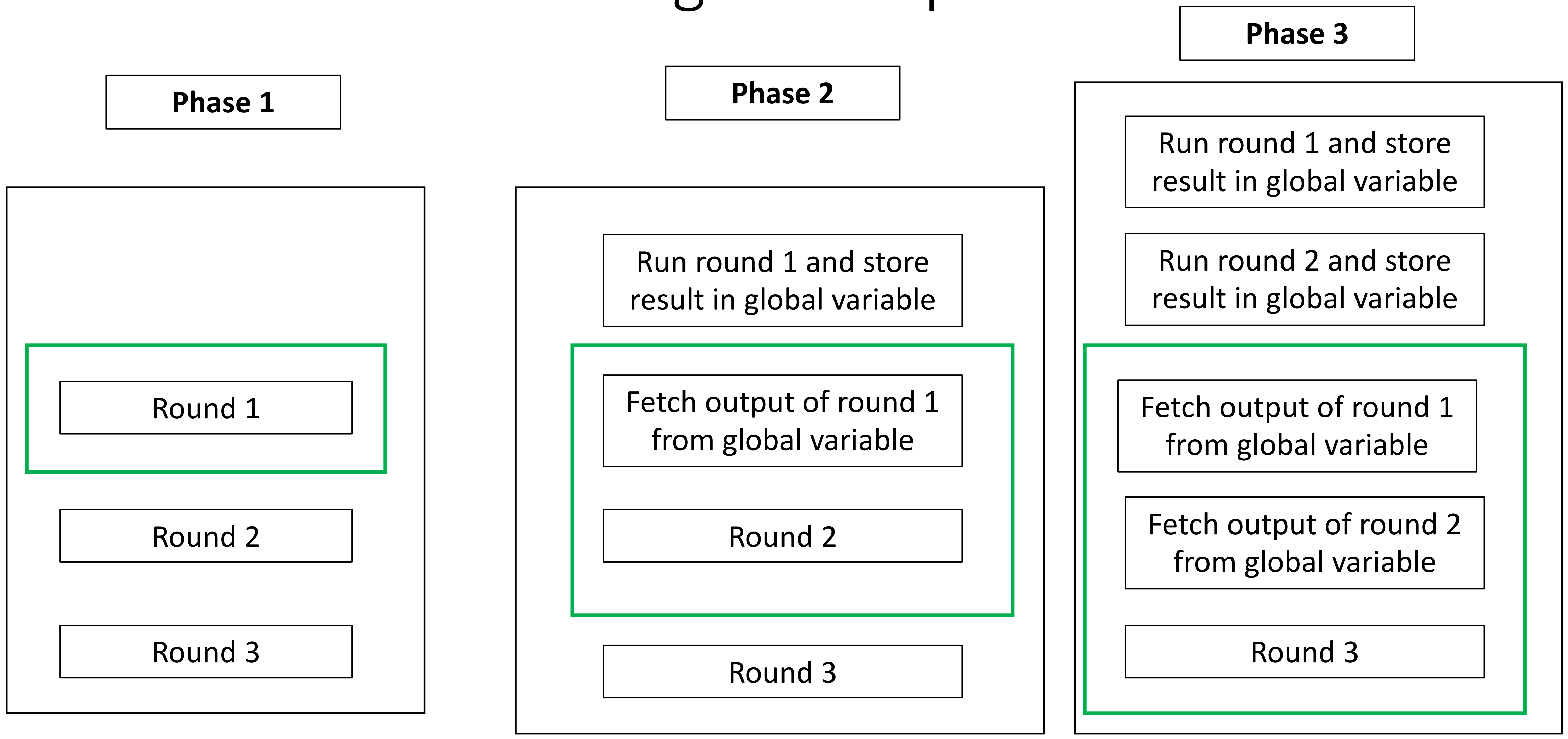
```
using Clock=std::chrono::high_resolution_clock;

foo();                  // Statement 2
auto t1 = Clock::now(); // Statement 1
auto t2 = Clock::now(); // Statement 3

auto elapsedTime = t2 - t1;
```

- https://stackoverflow.com/questions/37786547/enforcing-statement-order-in-c/37789799

Solution(s)

- Being in LLVM committee Chandler Carruth acknowledges this problem and suggest microbenchmarking.
- Microbenchmarking is similar to method suggested by Tzipora on 1/22/21 of benchmarking each section separately.
- Apart from that another benchmarking method is used where optimization is –O2 flag.

Micro benchmarking technique


Phase 1
Round 1
Round 2
Round 3
Phase 2
Run round 1 and store result in global variable
Fetch output of round 1 from global variable
Round 2
Round 3
Phase 3
Run round 1 and store result in global variable
Run round 2 and store result in global variable
Fetch output of round 1 from global variable
Fetch output of round 2 from global variable
Round 3

Green box indicates a loop that runs for 1000 times, rest of the blocks runs just once.

Micro benchmarking result

Phases	Time
Phase 1	21.31
Phase 2	3.17
Phase 3	3.54
Total PRF	**28.02**

-O2 flag optimization result

Phases	Time
Phase 1	23.17
Phase 2	3.31
Phase 3	9.9
Total PRF	**36.48**