Performance Analysis of Undefined Behavior Optimizations

***Lucian I. Popescu, **Nuno P. Lopes**

*Faculty of Automatic Control and Computer Science,
Politehnica University of Bucharest
** Instituto Superior Técnico,
Universidade de Lisboa

*lucian.popescu187@gmail.com, **nuno.lopes@tecnico.ulisboa.pt

1. Introduction

Clang and LLVM use undefined behavior (UB) to issue code optimizations. Currently, there is no study that evaluates the performance impact of this class of optimizations. We fill this gap by presenting some early results in this area. Phoronix Test Suite was used to evaluate the performance of a diverse set of applications, including webservers, compression algorithms, graphical environments, etc. By compiling each application with flags that trigger specific UBs, we gathered various metrics (requests per second, MB/s, FPS, etc) for further analysis. Early results show that in nearly 90% of the cases the performance impact is insignificant (between -2% and 2%).

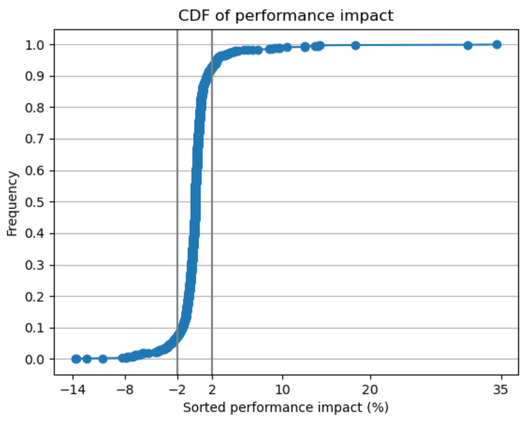

Figure 1: *CDF of performance impact for all UB flags*

2. Experiment Setup

The performance tests were run on a machine with the following specs:

- Processor: 2 x Intel Xeon E5-2680 v2 @ 3.60GHz (20 Cores / 40 Threads)
- Motherboard: HP 158A (J61 v03.69 BIOS)
- Chipset: Intel Xeon E7 v2/Xeon
- Memory: 8 x 8 GB DDR3-1600MT/s HMT41GR7AFR4A-PB
- Disk: 2000GB TOSHIBA DT01ACA2
- Graphics: NVIDIA NVE4 4GB
- OS: Debian 11
- Kernel: 5.10.0-21-amd64 (x86_64)
- Display Server: X Server 1.20.11
- Display Driver: nouveau
- Compiler: Clang 15.0.7 + LLVM 15.0.7
- File-System: ext4
- Screen Resolution: 1920x1080

The experiments were conducted using the following steps:

- Compile the benchmark with no UB flags enabled (baseline)
- Compile the benchmark using one UB flag at a time
- Run baseline using Phoronix and fetch the results
- Run the benchmarks with UB flags using Phoronix and fetch the results
- Compare the UB flags benchmark results with the baseline

3. Undefied Behavior flags

3.1 Existing flags

-fwrapv Treat signed integer overflow as two's complement. Stops the compiler from optimizing loops and TODO

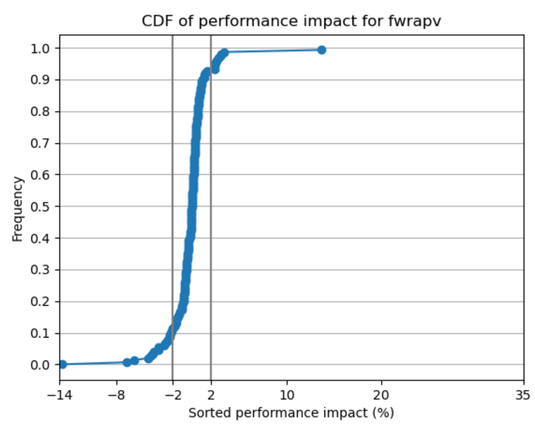

Figure 2: *CDF of performance impact for fwrapv*

-fno-strict-aliasing Drop the strictest aliasing rules applicable to the language being compiled.

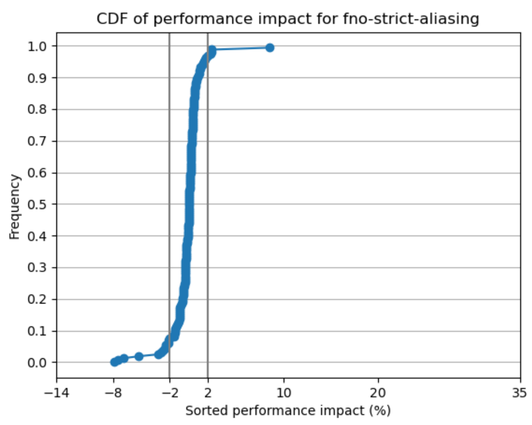

Figure 3: *CDF of performance impact for fno-strict-aliasing*

-fstrict-enums Enable optimizations based on the strict definition of an enum's value range.

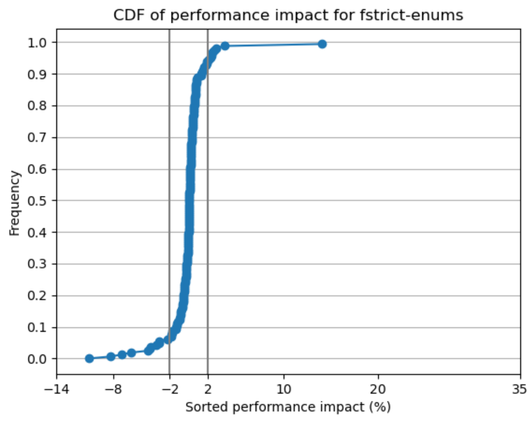

Figure 4: *CDF of performance impact for fstrict-enums*

-fno-delete-null-pointer-checks Assume that programs can safely dereference null pointers, and that code or data elements may resides at address zero. This disables the deletion of "redundant" NULL pointer checks.

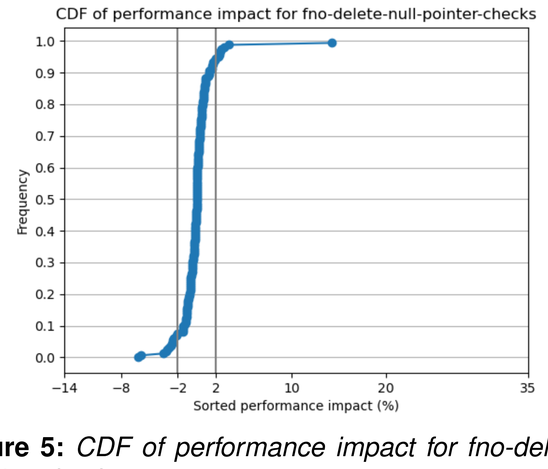

Figure 5: *CDF of performance impact for fno-delete-null-pointer-checks*

3.2 Flags added by us

-fconstrain-shift-value Shift amount is required to be at most the word size of LHS. An additional 'and' instruction is inserted.

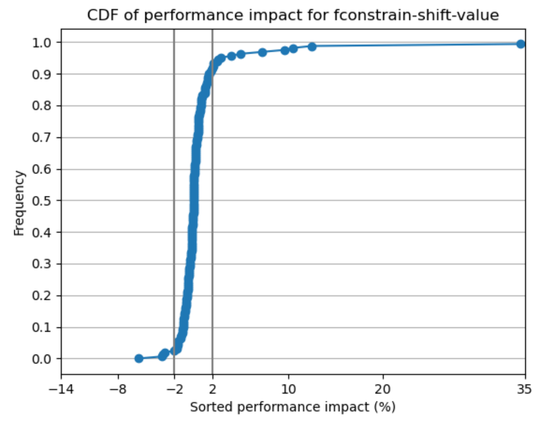

Figure 6: *CDF of performance impact for fconstrain-shift-value*

-fno-constrain-bool-value Do not constrain bool values in 0,1. Instead of using LLVM's range metadata, an additional 'and' instruction is inserted.

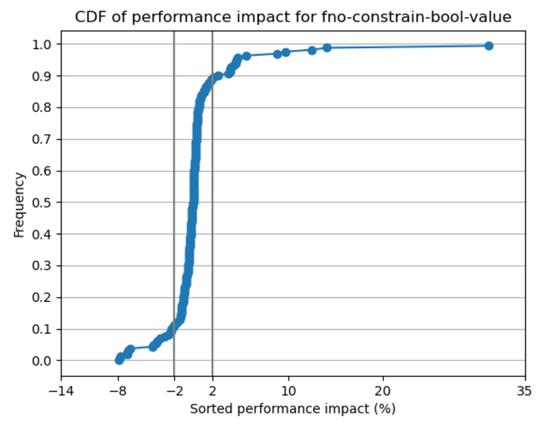

Figure 7: *CDF of performance impact for fno-constrain-bool-value*

-fno-use-default-alignment Use alignment of 1 for all memory operations including load, store, memcpy et al. Global variables and alloca's remain unaffected.

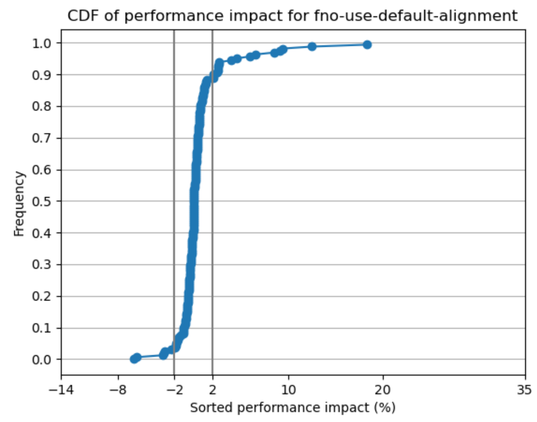

Figure 8: *CDF of performance impact for fno-use-default-alignment*

4. Outliers

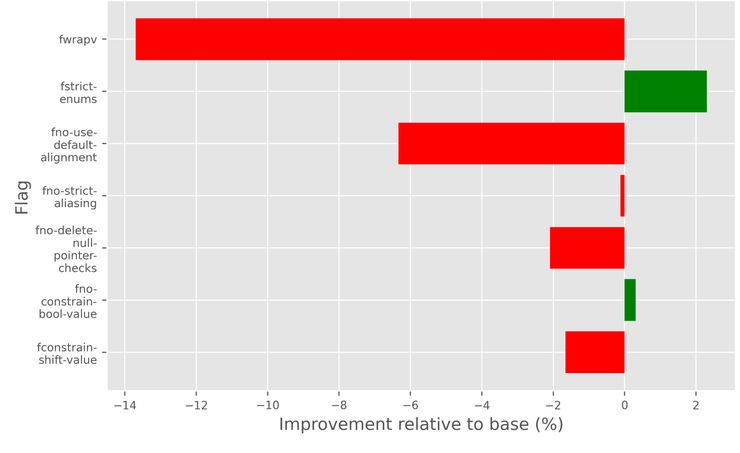

Figure 9: *eSpeak-NG Speech Engine - Text-To-Speech Synthesis, Baseline: 41.59 Sec*

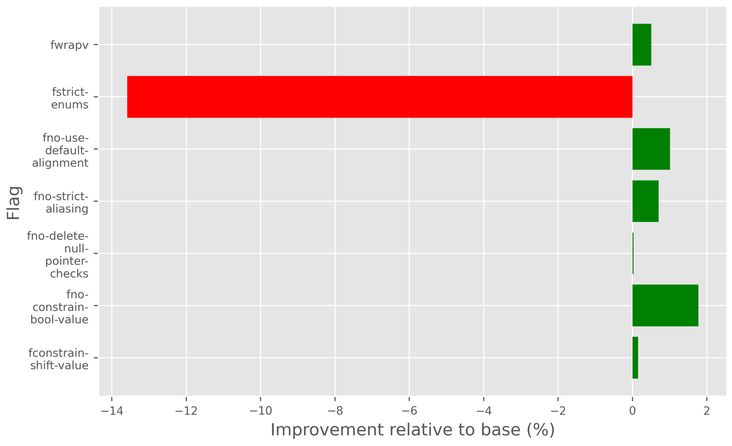

Figure 10: *Apache HTTP Server - Concurrent Requests: 500, Baseline: 102070.20 Reqs/Sec*

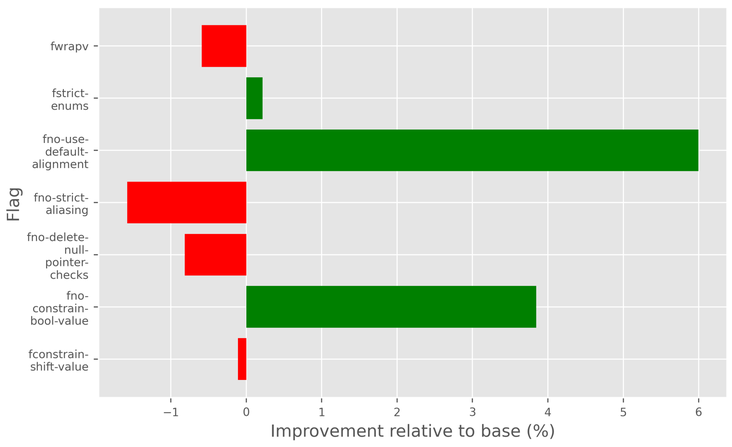

Figure 11: *Redis - Test: SADD - Parallel Connections: 500, Baseline: 1990361.58 Reqs/Sec*

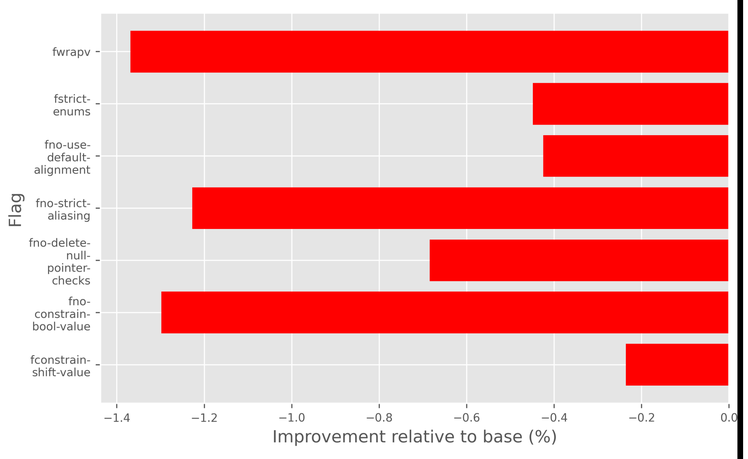

Figure 12: *CHANGEME*

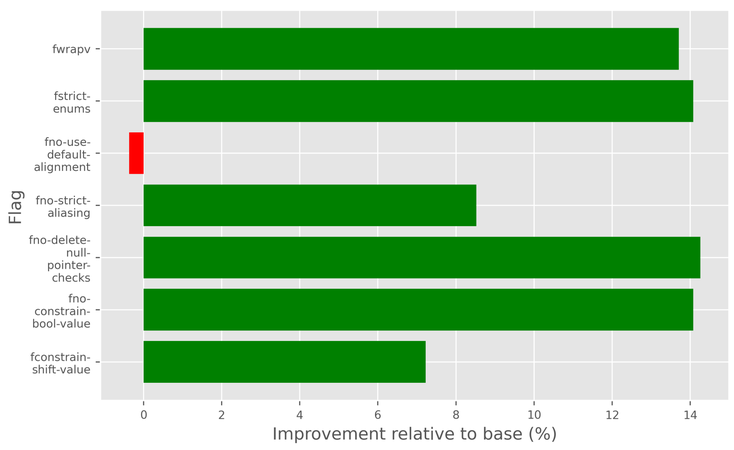

Figure 13: *GraphicsMagick - Operation: Rotate, Baseline: 540 Iterations/Min*

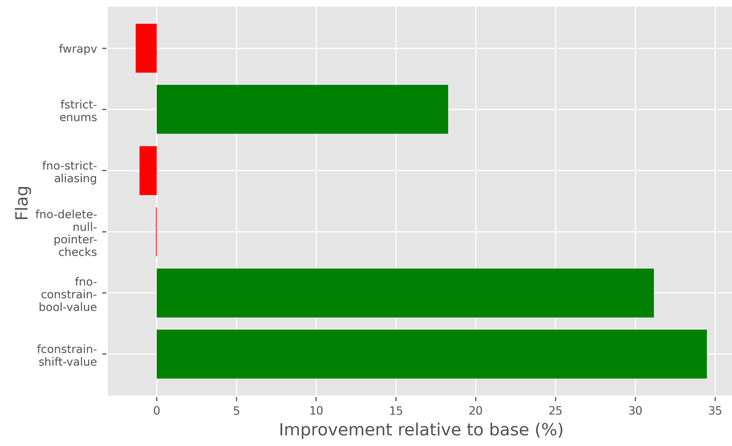

Figure 14: *GtkPerf - GTK Widget: GtkDrawingArea - Pixbufs, Baseline: 170.08 Sec*

5. Future work

- Clang known UBs are covered, proceed to LLVM UBs
- Run the benchmarks on different hardware architectures (AMD, ARM)
- Find a method of discovering new UBs (maybe using Alive)
- Run the benchmarks taking into account LTO and PGO