Performance Impact of Exploiting Undefined Behavior in C/C++

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

*Facultatea de Automatică și Calculatoare,
Universitatea Politehnică din București
** 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.

2. Impact

Early results show that in nearly 90% of the cases the performance impact is insignificant (between -2% and 2%).

3. 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), MicroArch: IvyBridge
- OS: Debian 11, kernel: 5.10.0-21-amd64 (x86_64)
- Compiler: Clang 15.0.7

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

4. Undefined Behavior Flags

Flag name	Flag Description
Existing Flags	
-fwrapv	Treat signed integer overflow as two's complement. Drops 'nsw' from IR.
-fno-strict-aliasing	Don't use type-based alias analysis. Drops '!tbaa' from IR.
-fstrict-enums	Enable optimizations that take advantage of enum's value ranges. Adds '!range' from IR.
-fno-delete-null-pointer-checks	Assume that programs can safely dereference null pointers.
-fno-finite-loops	Don't assume that all loops are finite. '!mustprogress' is not added to any loop or function.
Flags Added by Us	
-fconstrain-shift-value	Mask shift RHS so it doesn't produce poison when RHS >= bitwidth.
-fno-constrain-bool-value	Do not constrain bool values in {0,1}. Drops '!range' from IR.
-fno-use-default-alignment	Use alignment of 1 for all memory operations including load, store, memcpy, etc. Global variables and alloca's remain unaffected.
TODO Flags	
	Change uninitialized loads from undef to zero .
	Don't treat out-of-bounds memory accesses as UB.
	Don't treat use-after-free accesses as UB.
	Don't use object-based rules in alias analysis (use an aassembly-like memory mode).
	Remove arithmetic-related UBs (division by 0, etc).

-fwrapv

Out of 162 data points, 83% values are in the interval of $[-2,\ 2]$, 6% exhibit positive performance impact and 11% exhibit negative performance impact. This flag has the biggest overall negative performance impact, as presented in Figure 2.

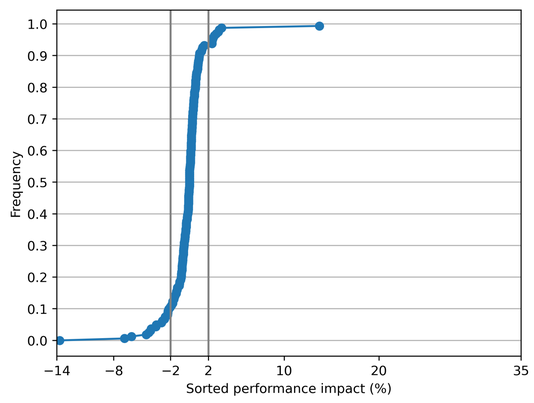

Figure 1: *CDF of performance impact for -fwrapv*

Other benchmarks with negative impact: FFTW - Float + SSE - Size: 1D FFT Size 256, uvg266 (Video Encoder). Other benchmarks with positive impact: OpenSSL - RSA4096.

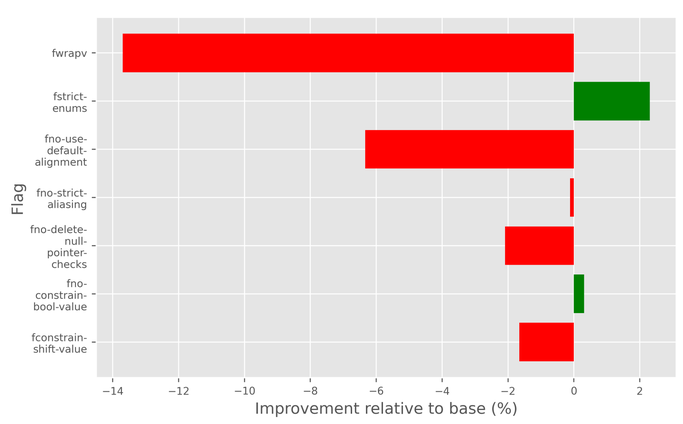

Figure 2: *eSpeak-NG Speech Engine - Text-To-Speech Synthesis, Baseline: 41.59 Sec*

-fno-strict-aliasing

Out of 162 data points, 88.8% values are in the interval of $[-2,\ 2]$, 3% exhibit positive performance impact and 8.2% exhibit negative performance impact. This is the most balanced flag up until this moment.

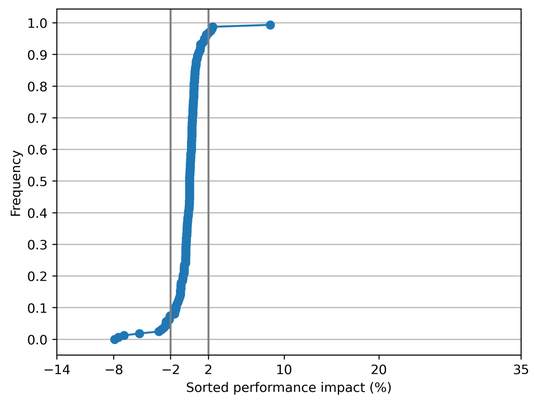

Figure 3: *CDF of performance impact for fno-strict-aliasing*

-fconstrain-shift-value

Out of 161 data points, 88.8% values are in the interval of $[-2,\ 2]$, 8.1% exhibit positive performance impact and 3.1% exhibit negative performance impact. This flag has the biggest overall positive performance impact, as presented in Figure 5.

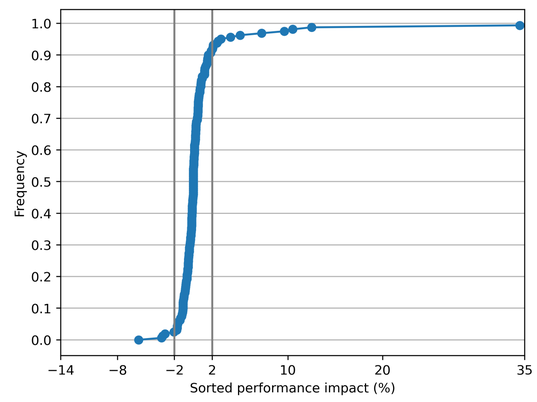

Figure 4: *CDF of performance impact for -fconstrain-shift-value*

Other benchmarks with negative impact: GraphicsMagick - Operation: Swirl, PJSIP - Method: OPTIONS Stateful. Other benchmarks with positive impact: OpenSSL - AES-256-GCM.

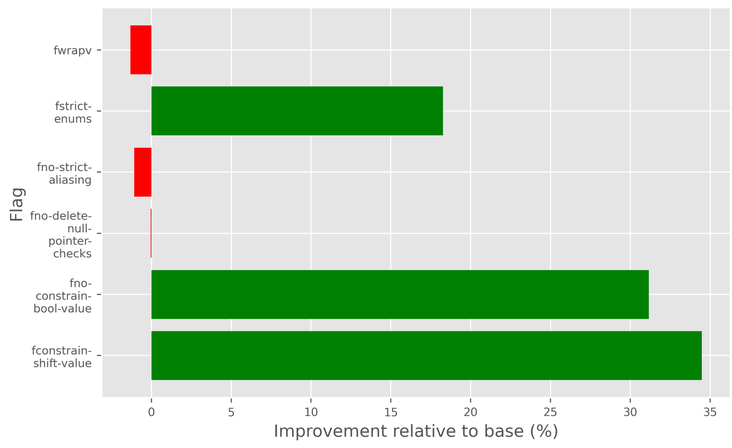

Figure 5: *GtkPerf - GTK Widget: GtkDrawingArea - Pixbufs, Baseline: 170.08 Sec*

-fno-use-default-alignment

Out of 158 data points, 85.5% values are in the interval of $[-2,\ 2]$, 10.5% exhibit positive performance impact and 4% exhibit negative performance impact. For this flag we expected the negative impact to be greater than the positive impact.

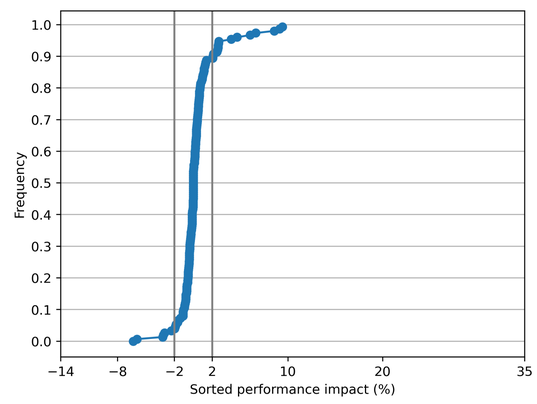

Figure 6: *CDF of performance impact for -fno-use-default-alignment*

5. Future work

- Implement and benchmark other classes of UB.
- Run the benchmarks on different hardware architectures (AMD, ARM).
- Combine the flags when compiling the benchmarks.
- Find a method of discovering new UBs (maybe using Alive2).
- Run the benchmarks taking into account LTO and PGO.