Taming Undefined Behavior in LLVM

Artifact Description

1. LLVM Source Code & Examples

You can download base LLVM and Clang by running `clone-base.sh`. For your convenience, we already archived base LLVM and Clang to `llvm-base` and `clang-base`. Our implementations are stored as patch files. Please see `patch` directory. Patched LLVM and Clang are archived in `llvm-freeze` and `clang-freeze`. You can compile LLVM by running `'build-llvm.sh (base/freeze) <destdir>'`. For example, running `'build-llvm.sh base ./base'` will compile base LLVM, and locate binaries (`clang`, `opt`, ..) into `./base/bin`.

In `example` directory, `basic.ll` contains a simple function with `freeze` instruction inside. `loopunswitch.c` is a C program which makes clang do loop unswitch. There are two LLVM bugs (bug27506 and bug31652) which occurred due to the bad interaction between loop unswitch and GVN. To reproduce the buggy behavior, run `'run.sh <llvm-dir>'`. Directory `'<llvm-dir>/bin/'` must contain executable files `clang` and `clang++`. You can also confirm that it is fixed by our solution.

In our implementation, freeze operation does not accept vector value. It is because simply accepting integer value was enough to fix problematic optimizations under our semantics. We'll add a mention to this into our final draft.

2. Benchmarks

Two directories (`llvm-test-suite` and `lnt`) in the github repo are what we had used for `LLVM Nightly Test` performance estimation. You can checkout them from online by running `checkout-lnt.sh` as well.

`singlefileprograms` directory contains large single file programs which were used to estimate compiler's speed, memory usage, and object file size.

We did not archive SPEC benchmark on our repository because it is a commercial software. I stored configuration files I used as well as `REAMDME.md` into `spec/` folder.

3. Running Experiments

3.1 Environmental Settings

Ubuntu 16.04 is the most suitable OS for running our experiment scripts, but other linux distributions are fine as well. Prior to running experiment, you'll have to install `python 2.7`, `git`, `virtualenv`, `bc`, `python-dev`, `zlib1g-dev`, `yacc`, `tcl-dev` packages. If you're using Ubuntu, you can install these packages with `apt-get` command.

If you want to run experiment under `cpuset` as depicted in our paper, you need a few more steps. First of all, you'll have to create a new user. Running `cpuset` requires super-user priviledge, and our scripts are tailored to an imaginary user `pldi1717` with password `pldi201717`. You have to download the artifact & run your experiment on user `pldi1717`. Second, install `cpuset` by running `'install-cpuset.sh'`. It will require sudoer's priviledge. Third, modify scripts according to instructions in the following sections.

3.2 LLVM Nightly Test

Execute `'init-lnt.sh <sandbox dir>'` to instantiate a python sandbox. Sandbox directory is the place for compiling LNT test cases. After instantiation, run `'run-lnt.sh <sandbox-dir> <llvm-dir>'`. It will automatically start running test cases in LNT. Experimental results will be recorded at `'<sandbox-dir>/test-.../report.simple.csv'`. See `CC_Real_Time` column for compilation time, and `Exec_Real_Time` column for running time. The result is given in seconds.

If you want to use `cpuset`, you need to modify `'llvm-test-suite/RunSafely.sh'`. Please replace the word `'/mnt/freezedisk/cpuset'` with absolute path of `'pldi17-ae/cpuset'`. After that, uncomment the lines (198 - 202), and comment line 197. Finally, run `'run-lnt.sh'` as written above.

3.3 Large Single File Programs

Compilation Time Run `'compiletime.sh <llvm-dir> <output-dir>'`. It will print compilation time for each program. Among measures, `real` is the one we used in the paper. If you want to use `cpuset`, run `compiletime-cpuset.sh` instead of `compiletime.sh`. You should replace `'/mnt/freezedisk/cpuset'` in `compiletime-cpuset.sh` with absolute path of `pldi17-ae/cpuset'`.

Memory Usage Execute `'memfoot.sh <llvm-dir> <output-dir>'`. It will record memory usages for every 0.02 secs, and create `*.summary.csv`. Please refer to `Max RSS` and `Max VSZ` columns. If you want to use `cpuset`, run `memfoot-cpuset.sh` instead of `memfoot.sh`. You need to replace `'/mnt/freezedisk/cpuset'` in the script with absolute path of `pldi17-ae/cpuset`.

Object Size Execute `'compileall.sh <llvm-dir> <output-dir>'`. It will create object files as well as bitcode files to `<output-dir>`. To count # of instructions, use executable `instcounter/instcounter`. To build `instcounter`, run `'instcounter/build.sh <llvm-dir>'`.

You can navigate files from `https://github.com/pldi17-17/pldi17-17`. To use `cpuset`, your computer should have at least 4 cores.