Landslide: Systematic Testing in CMPSC 473
User Guide

Ben Blum

1 Introduction

You should have received a document titled *User Study Information Sheet* that describes your rights as a study participant and our confidentiality procedures. If you are missing this, please contact me immediately. My email address is `bblum@cs.cmu.edu`.

Likewise, please make sure you have read and understand the lecture slides on systematic testing, which are available on the CMPSC 473 course website. If you have any questions about the research background I would be happy to answer them.

2 Instructions

We encourage you to use Landslide on the Vagrant VM or the W204 cluster machines. Landslide has not been tested with other configurations and may fail mysteriously on other development environments, and we cannot promise to be of any help.

1. Clone the repository - `git clone https://github.com/bblum/landslide.git` - this will be your workspace. On the Vagrant VM, do this in the home directory, `~/`, *not* in the shared host directory `/vagrant`.
2. Run `./psu-setup.sh /vagrant`, or if using on the W204 cluster, replace `/vagrant` with the absolute path to your project directory.
   - This command will import your project into the `./pebsim/p2-basecode/` subdirectory of the landslide repository and then attempt to build it. Whenever you update your thread library code, please do so in its original `/vagrant` location and then run `psu-setup.sh` again.
3. Run `./landslide OPTIONS` to run tests with Landslide, replacing `OPTIONS` with any of the following:
   (a) Use `./landslide -h` to view command line options. The most important ones are `-t`, max testing time, and `-p`, which test program to run.
   (b) Suggested test configurations to run are as follows. You can, of course, change the time limits or test programs as you see fit.
      i. Atomic tests
         - `./landslide -t 30m -p atomic_fetch_add`
         - `./landslide -t 30m -p atomic_fetch_sub`
         - `./landslide -t 30m -p atomic_exchange`
         - `./landslide -t 30m -p atomic_compare_swap`
      ii. Threading tests
         - `./landslide -t 30m -p thr_exit_join`
         - `./landslide -t 30m -p mutex_test`

- `./landslide -t 30m -p paradise_lost`
- `./landslide -t 30m -p broadcast_test`
- `./landslide -t 30m -p broadcast_two_waiters`
- `./landslide -t 30m -p paraguay`
- `./landslide -t 30m -p rwlock_downgrade_read_test`

- `./landslide -t 8h -p thr_exit_join`
- `./landslide -t 8h -p mutex_test`
- `./landslide -t 8h -p paradise_lost`
- `./landslide -t 8h -p broadcast_test`
- `./landslide -t 8h -p broadcast_two_waiters`
- `./landslide -t 8h -p paraguay`
- `./landslide -t 8h -p rwlock_downgrade_read_test`

During the course of your debugging you may wish to read the tests' source code to see what they're doing. You can find these tests in `pebsim/p2-basecode/410user/progs/`.

(c) Landslide's bug reports will show up in the current directory as HTML files which you can view in a web browser.

- On the Vagrant VM, move these files to `/vagrant`, the shared directory, and open them with a web browser on the host.
- On the W204 cluster, you should be able to open the files directly from the landslide repository.

Landslide may also print some *data races*, even if it doesn't emit an HTML bug report, for example as follows:

```
Found a racy access at 0x0100291e in atomic_compare_swap (atomic.S:42)
```

These are not necessarily bugs; please refer to slide 29 of the Landslide lecture for details about what this means.

4. After fixing any found bugs, re-run Landslide with the same test options to confirm that your fix works. (Remember to re-run `psu-setup.sh` as in step 2 to make sure Landslide sees your updates.)

Note on simultaneous Landslides: Landslide does not support running multiple tests at once; it needs to create some temporary files to annotate your code, and multiple instances of Landslide can step on each other's toes by clobbering those files.

Asking questions: If you have any techical questions about Landslide, such as how to interpret its html interleaving traces, how to write a custom test case, etc., feel free to email me (`bblum@cs.cmu.edu`). I will *not* provide advice on the thread library project itself, nor will I personally look at your code to help you fix your bugs. The point of the study is that Landslide's automation plus your own brain should be enough! On the other hand, if you have a design question while deciding between potential ways to fix a bug, send it to CMPSC 473 course staff (not me) as usual.

If something goes wrong: If you find a Landslide bug, such as an assertion/crash (it will say "Landslide crashed. This is not your fault."), or such as a bug report that you think is actually a bug in Landslide rather than a bug in your project, or such as it getting stuck and not exiting in the given time limit, please create a tarball of your workspace using the command `tar cjvf landslide-crash.tar.bz2 ./`, email it to me (`bblum@cs.cmu.edu`), and I'll try to fix your issue promptly.