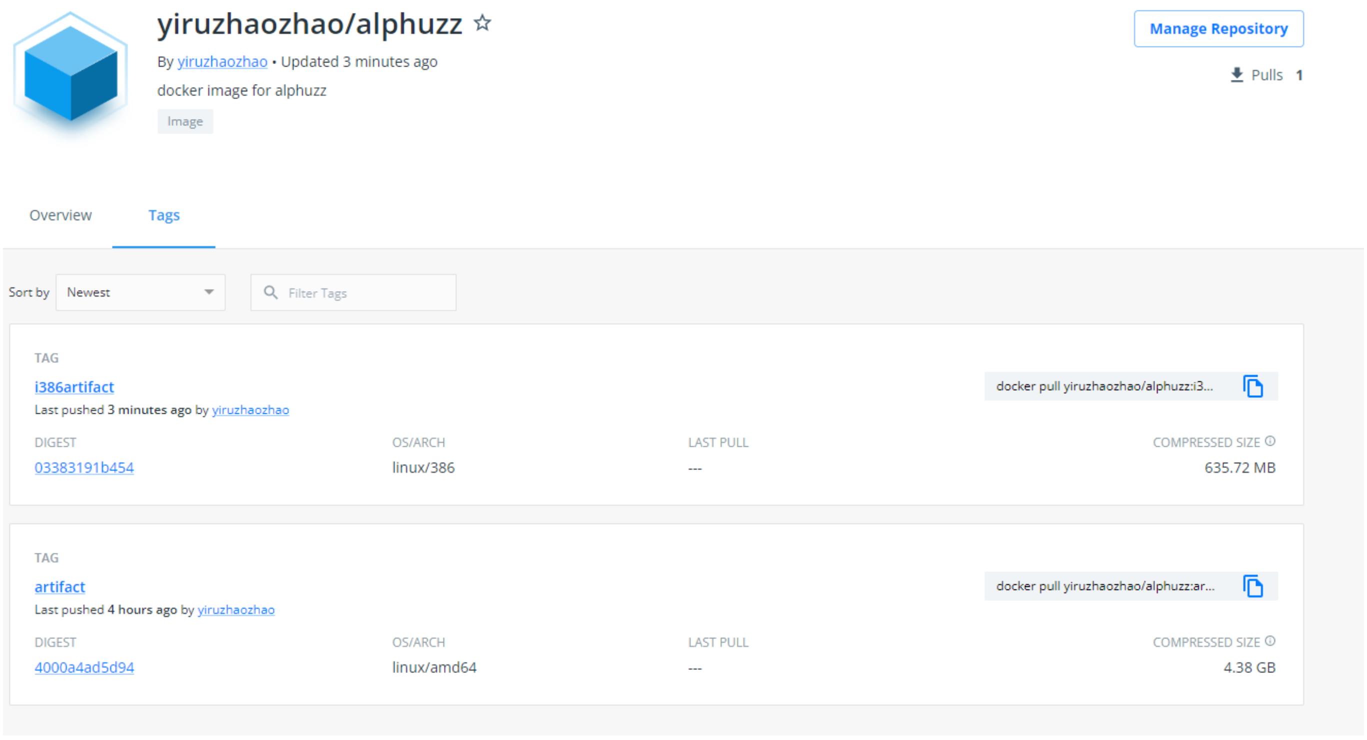
Docker image

We publish two docker images on docker hub. The size of the images are about 14.7GB. For Unifuzz and 12-real-world binaries, we build a docker image on top of ubuntu16.04 X86. For CGC dataset, we build a docker image on top of ubuntu16.04 i386.

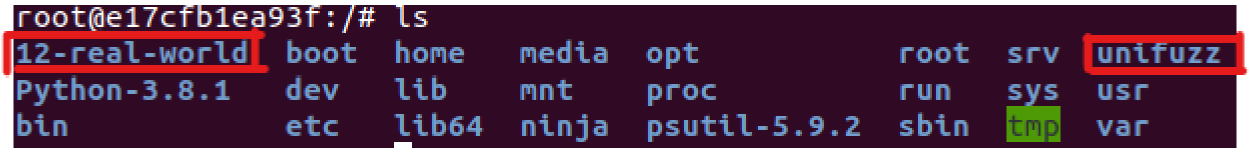

yiruzhaozhao/alphuzz ☆
By yiruzhaozhao • Updated 3 minutes ago
docker image for alphuzz
Image
Manage Repository
Pulls 1
Overview Tags
Sort by Newest Filter Tags

TAG	DIGEST	OS/ARCH	LAST PULL	COMPRESSED SIZE
i386artifact (Last pushed 3 minutes ago by yiruzhaozhao) — docker pull yiruzhaozhao/alphuzz:i3...	03383191b454	linux/386	---	635.72 MB
artifact (Last pushed 4 hours ago by yiruzhaozhao) — docker pull yiruzhaozhao/alphuzz:ar...	4000a4ad5d94	linux/amd64	---	4.38 GB

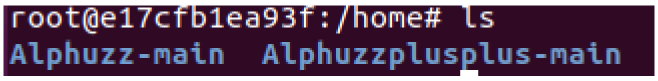
** docker image X86**

```
$ sudo docker pull yiruzhaozhao/alphuzz:artifact

$ sudo docker run --privileged -it yiruzhaozhao/alphuzz:artifact /bin/bash
```

We put the datasets binaries and initial seeds under the root directory.


```
root@e17cfb1ea93f:/# ls
12-real-world  boot  home   media  opt            root  srv  unifuzz
Python-3.8.1   dev   lib    mnt    proc           run   sys  usr
bin            etc   lib64  ninja  psutil-5.9.2   sbin  tmp  var
```

We put the Alphuzz and Alpuzzplusplus under the *home* directory.

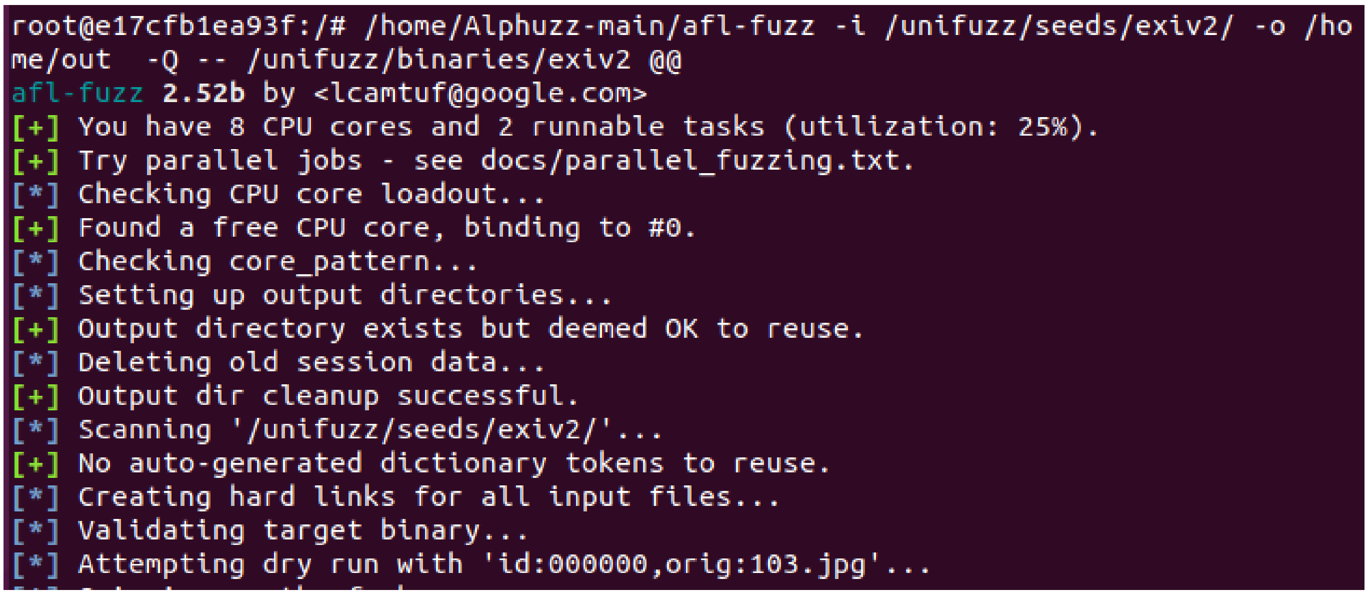

```
root@e17cfb1ea93f:/home# ls
Alphuzz-main  Alphuzzplusplus-main
```

Example

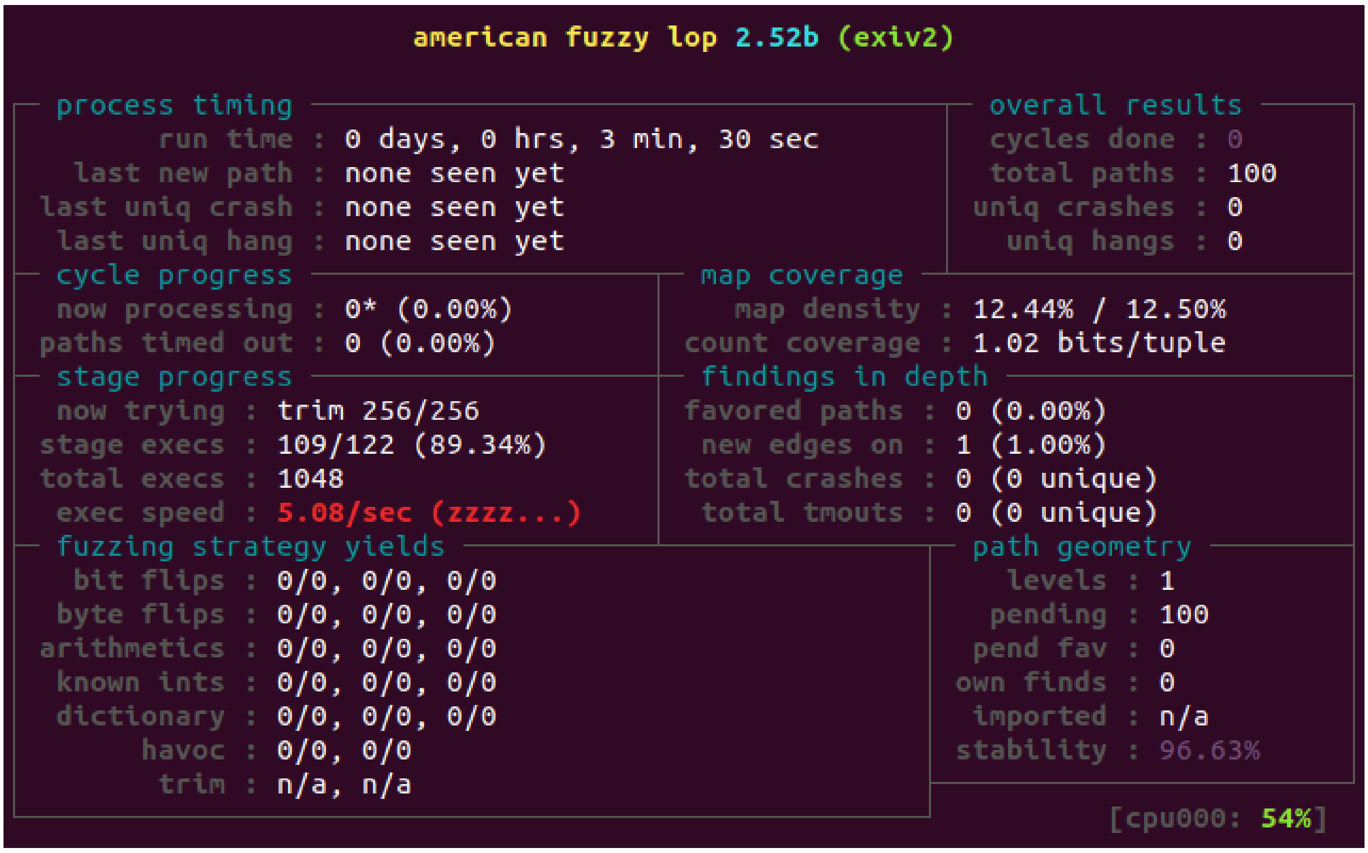
Take Unifuzz dataset for example.

Alphuzz

```
$ /home/Alphuzz-main/afl-fuzz -i /unifuzz/seeds/exiv2 -o /home/out -Q --
/unifuzz/binaries/exiv2 @@
```



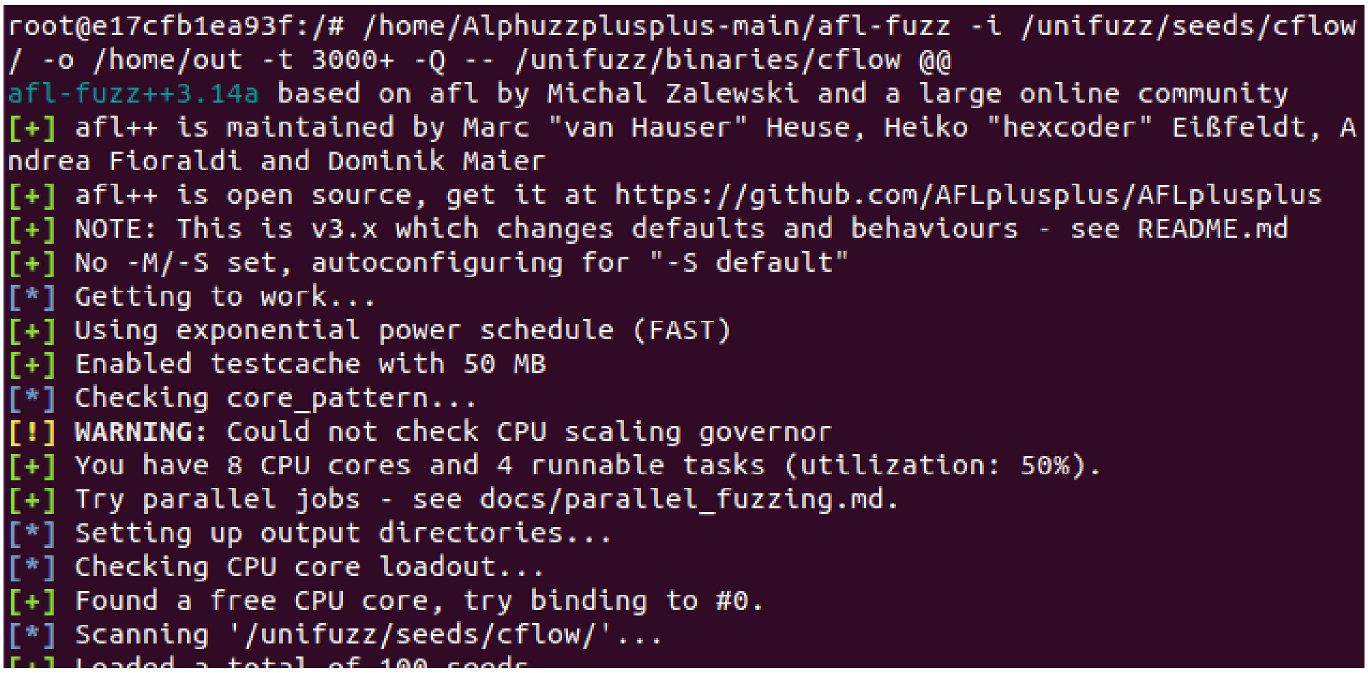
```
root@e17cfb1ea93f:/# /home/Alphuzz-main/afl-fuzz -i /unifuzz/seeds/exiv2/ -o /ho
me/out  -Q -- /unifuzz/binaries/exiv2 @@
afl-fuzz 2.52b by <lcamtuf@google.com>
[+] You have 8 CPU cores and 2 runnable tasks (utilization: 25%).
[+] Try parallel jobs - see docs/parallel_fuzzing.txt.
[*] Checking CPU core loadout...
[+] Found a free CPU core, binding to #0.
[*] Checking core_pattern...
[*] Setting up output directories...
[+] Output directory exists but deemed OK to reuse.
[*] Deleting old session data...
[+] Output dir cleanup successful.
[*] Scanning '/unifuzz/seeds/exiv2/'...
[+] No auto-generated dictionary tokens to reuse.
[*] Creating hard links for all input files...
[*] Validating target binary...
[*] Attempting dry run with 'id:000000,orig:103.jpg'...
```



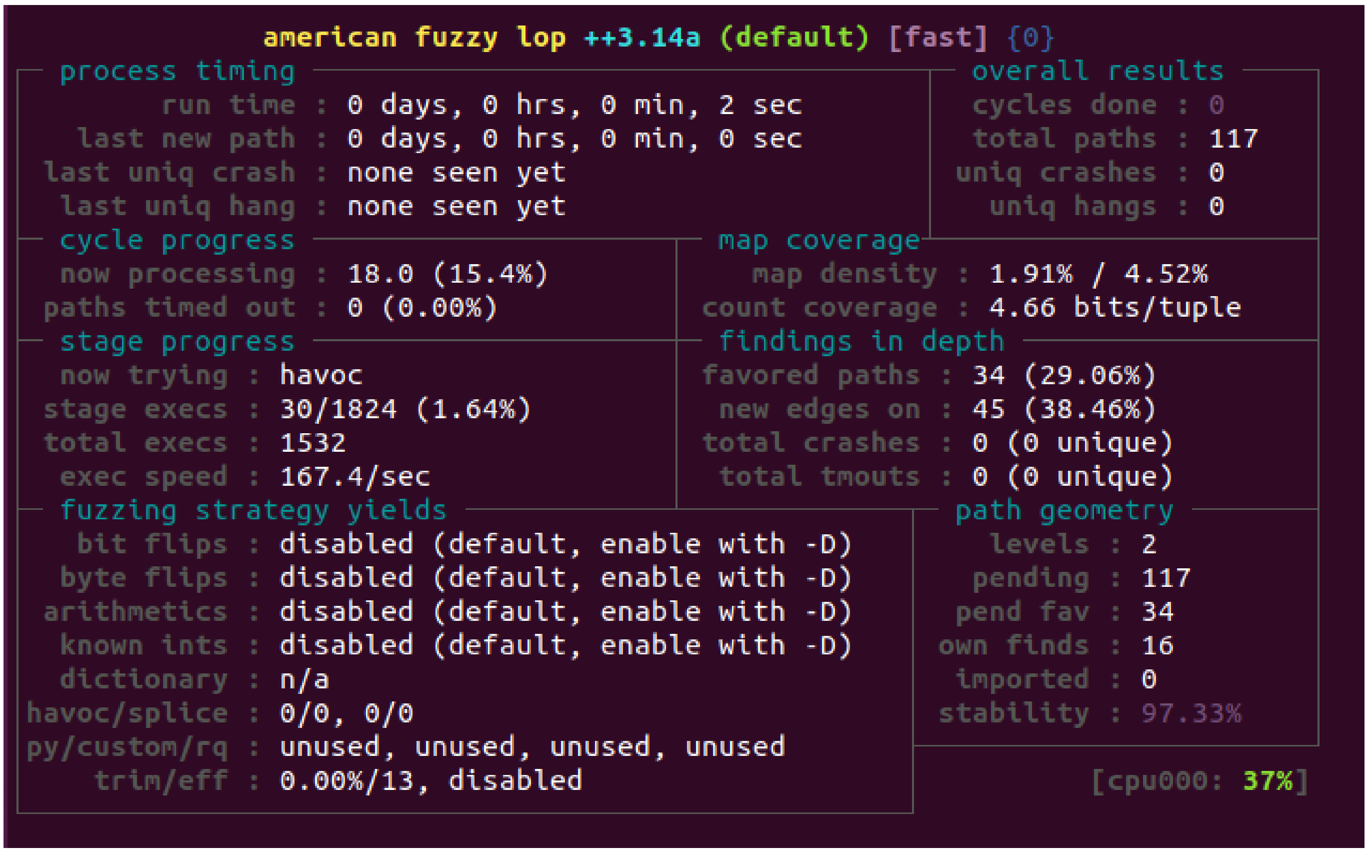
```
                     american fuzzy lop 2.52b (exiv2)

┌─ process timing ─────────────────────────────────────┬─ overall results ────┐
│        run time : 0 days, 0 hrs, 3 min, 30 sec       │  cycles done : 0     │
│   last new path : none seen yet                      │  total paths : 100   │
│ last uniq crash : none seen yet                      │ uniq crashes : 0     │
│  last uniq hang : none seen yet                      │   uniq hangs : 0     │
├─ cycle progress ────────────────────┬─ map coverage ─┴──────────────────────┤
│  now processing : 0* (0.00%)        │    map density : 12.44% / 12.50%      │
│ paths timed out : 0 (0.00%)         │ count coverage : 1.02 bits/tuple      │
├─ stage progress ────────────────────┼─ findings in depth ───────────────────┤
│  now trying : trim 256/256          │ favored paths : 0 (0.00%)             │
│ stage execs : 109/122 (89.34%)      │  new edges on : 1 (1.00%)             │
│ total execs : 1048                  │ total crashes : 0 (0 unique)          │
│  exec speed : 5.08/sec (zzzz...)    │  total tmouts : 0 (0 unique)          │
├─ fuzzing strategy yields ───────────┴───────────────┬─ path geometry ────────┤
│   bit flips : 0/0, 0/0, 0/0                         │    levels : 1          │
│  byte flips : 0/0, 0/0, 0/0                         │   pending : 100        │
│ arithmetics : 0/0, 0/0, 0/0                         │  pend fav : 0          │
│  known ints : 0/0, 0/0, 0/0                         │ own finds : 0          │
│  dictionary : 0/0, 0/0, 0/0                         │  imported : n/a        │
│       havoc : 0/0, 0/0                              │ stability : 96.63%     │
│        trim : n/a, n/a                              ├────────────────────────┘
└─────────────────────────────────────────────────────┘          [cpu000: 54%]
```

Alphuzzplusplus

```
$ /home/Alphuzzplusplus-main/afl-fuzz -i /unifuzz/seeds/cflow -o /home/out -Q -t
3000+ -- /unifuzz/binaries/cflow @@
```



```
root@e17cfb1ea93f:/# /home/Alphuzzplusplus-main/afl-fuzz -i /unifuzz/seeds/cflow
/ -o /home/out -t 3000+ -Q -- /unifuzz/binaries/cflow @@
afl-fuzz++3.14a based on afl by Michal Zalewski and a large online community
[+] afl++ is maintained by Marc "van Hauser" Heuse, Heiko "hexcoder" Eißfeldt, A
ndrea Fioraldi and Dominik Maier
[+] afl++ is open source, get it at https://github.com/AFLplusplus/AFLplusplus
[+] NOTE: This is v3.x which changes defaults and behaviours - see README.md
[+] No -M/-S set, autoconfiguring for "-S default"
[*] Getting to work...
[+] Using exponential power schedule (FAST)
[+] Enabled testcache with 50 MB
[*] Checking core_pattern...
[!] WARNING: Could not check CPU scaling governor
[+] You have 8 CPU cores and 4 runnable tasks (utilization: 50%).
[+] Try parallel jobs - see docs/parallel_fuzzing.md.
[*] Setting up output directories...
[*] Checking CPU core loadout...
[+] Found a free CPU core, try binding to #0.
[*] Scanning '/unifuzz/seeds/cflow/'...
[+] Loaded a total of 100 seeds.
```

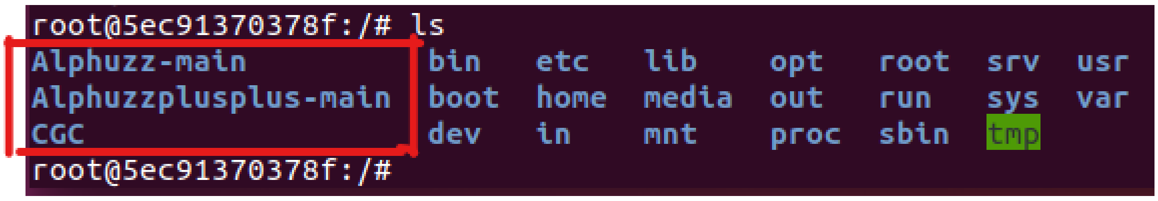


```
            american fuzzy lop ++3.14a (default) [fast] {0}
┌─ process timing ────────────────────────────────────┬─ overall results ────┐
│        run time : 0 days, 0 hrs, 0 min, 2 sec       │  cycles done : 0     │
│   last new path : 0 days, 0 hrs, 0 min, 0 sec       │  total paths : 117   │
│ last uniq crash : none seen yet                     │ uniq crashes : 0     │
│  last uniq hang : none seen yet                     │   uniq hangs : 0     │
├─ cycle progress ───────────────────┬─ map coverage──┴──────────────────────┤
│  now processing : 18.0 (15.4%)     │    map density : 1.91% / 4.52%        │
│ paths timed out : 0 (0.00%)        │ count coverage : 4.66 bits/tuple      │
├─ stage progress ───────────────────┼─ findings in depth ───────────────────┤
│  now trying : havoc                │ favored paths : 34 (29.06%)           │
│ stage execs : 30/1824 (1.64%)      │  new edges on : 45 (38.46%)           │
│ total execs : 1532                 │ total crashes : 0 (0 unique)          │
│  exec speed : 167.4/sec            │  total tmouts : 0 (0 unique)          │
├─ fuzzing strategy yields ──────────┴─────────────┬─ path geometry ────────┤
│   bit flips : disabled (default, enable with -D) │    levels : 2          │
│  byte flips : disabled (default, enable with -D) │   pending : 117        │
│ arithmetics : disabled (default, enable with -D) │  pend fav : 34         │
│  known ints : disabled (default, enable with -D) │ own finds : 16         │
│  dictionary : n/a                                │  imported : 0          │
│havoc/splice : 0/0, 0/0                           │ stability : 97.33%     │
│py/custom/rq : unused, unused, unused, unused     ├────────────────────────┘
│    trim/eff : 0.00%/13, disabled                 │          [cpu000: 37%]
└──────────────────────────────────────────────────┘
```

** docker image i386**

```
$ sudo docker pull yiruzhaozhao/alphuzz:i386artifact

$ sudo docker run --privileged -it yiruzhaozhao/alphuzz:i386artifact /bin/bash
```

We put the datasets binaries, Alphuzz and Alpuzzplusplus under the / directory.



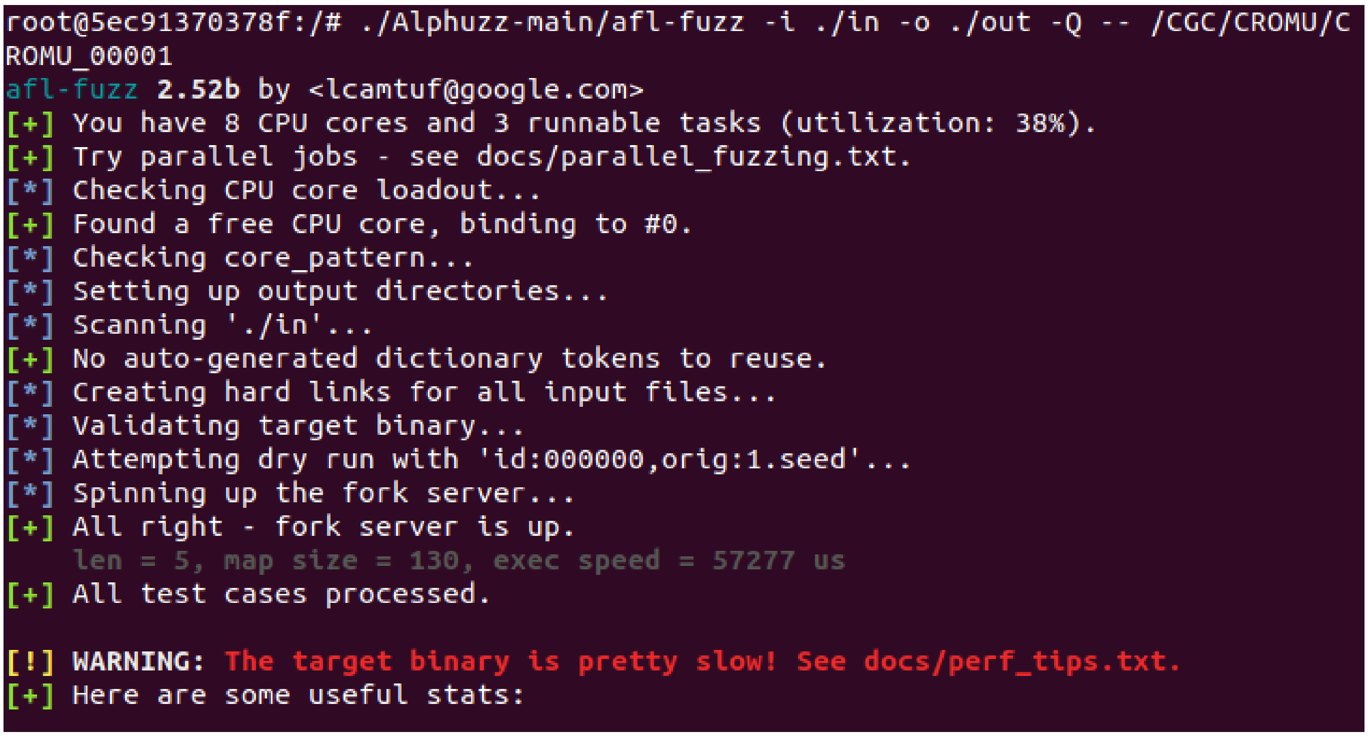
```
root@5ec91370378f:/# ls
Alphuzz-main          bin   etc   lib    opt   root  srv  usr
Alphuzzplusplus-main  boot  home  media  out   run   sys  var
CGC                   dev   in    mnt    proc  sbin  tmp
root@5ec91370378f:/#
```

Example

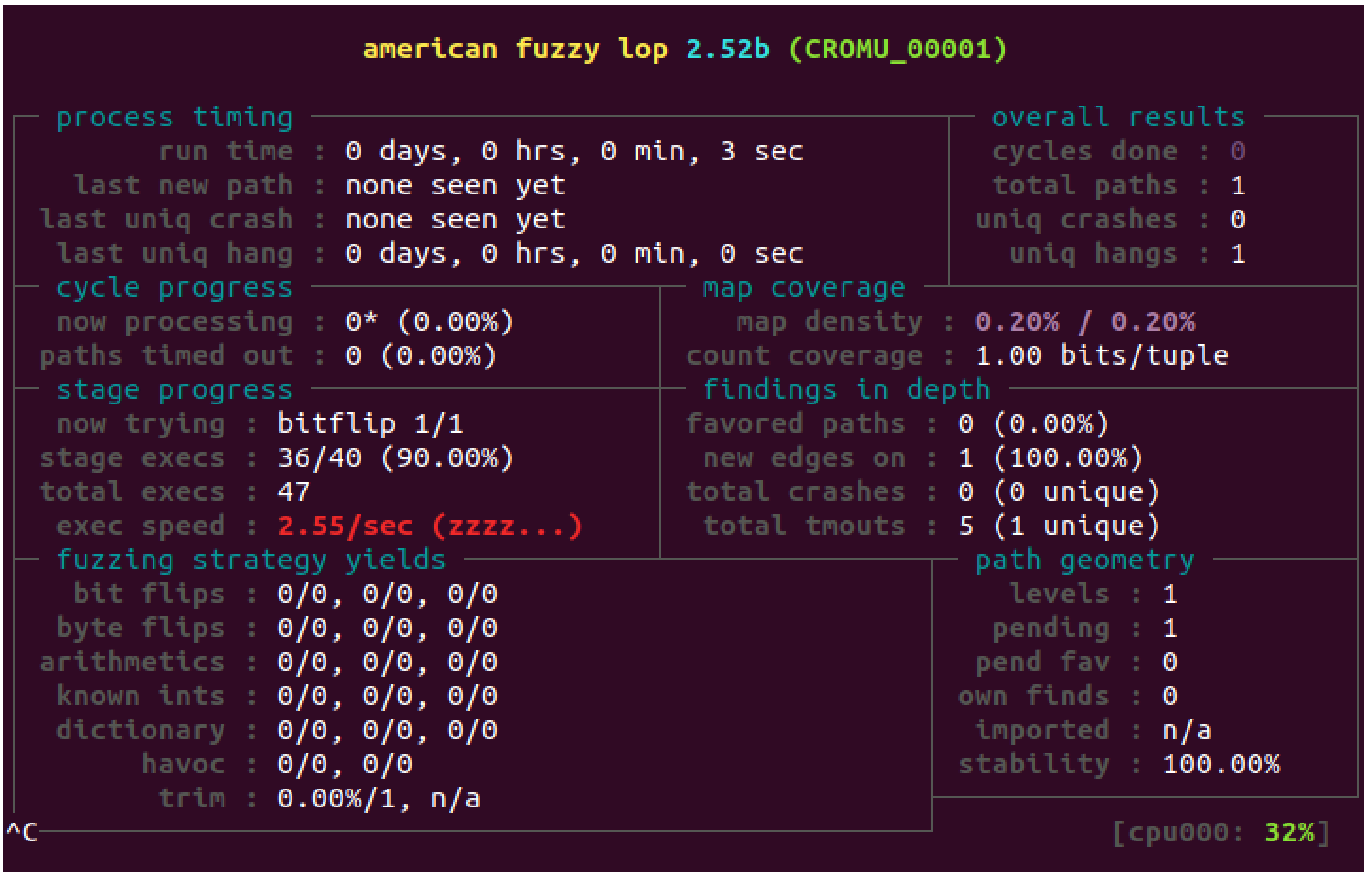
Take Unifuzz dataset for example.

Alphuzz

```
$ /Alphuzz-main/afl-fuzz -i ./in -o ./out -Q -- /CGC/CROMU/CROMU_00001
```



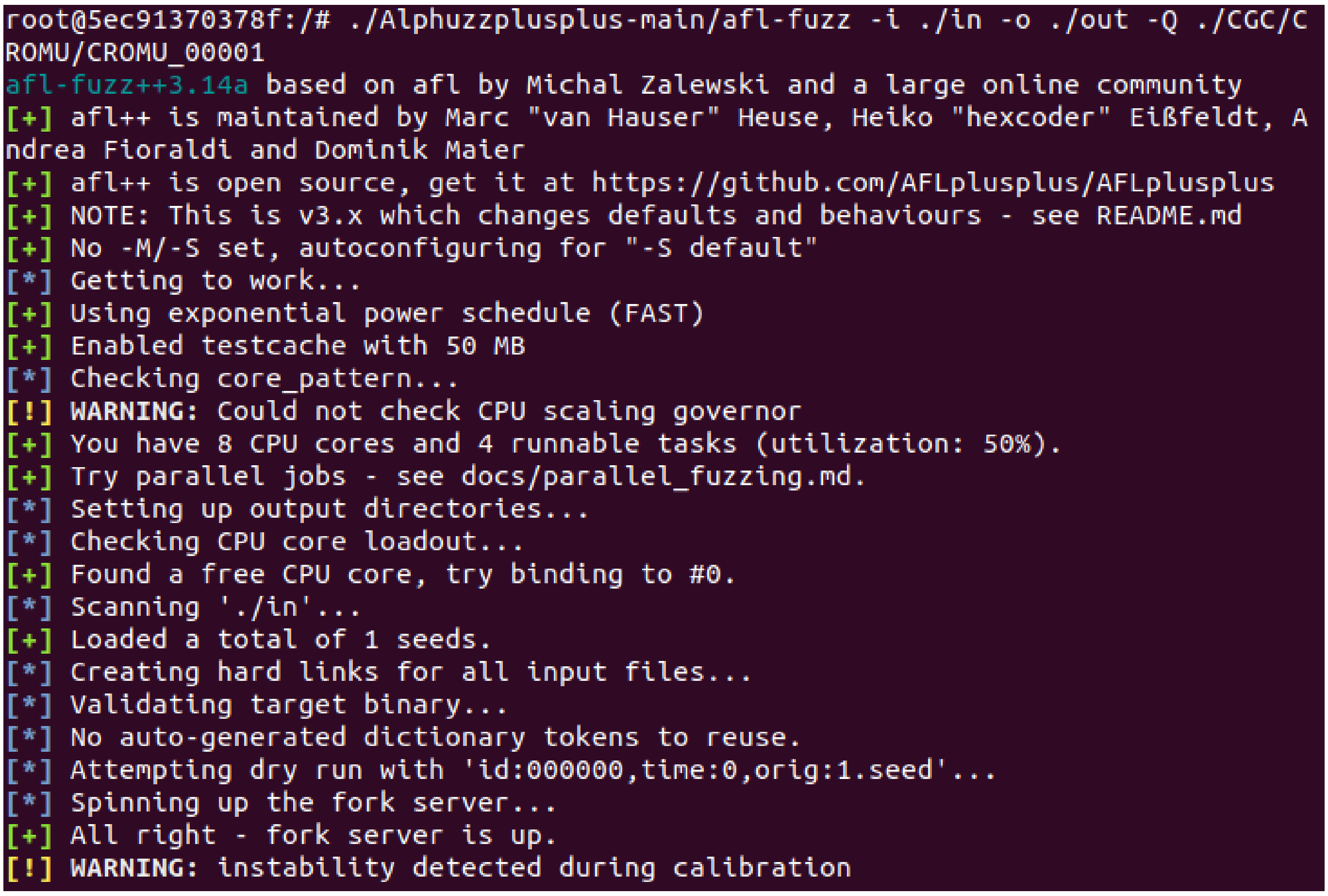
```
root@5ec91370378f:/# ./Alphuzz-main/afl-fuzz -i ./in -o ./out -Q -- /CGC/CROMU/C
ROMU_00001
afl-fuzz 2.52b by <lcamtuf@google.com>
[+] You have 8 CPU cores and 3 runnable tasks (utilization: 38%).
[+] Try parallel jobs - see docs/parallel_fuzzing.txt.
[*] Checking CPU core loadout...
[+] Found a free CPU core, binding to #0.
[*] Checking core_pattern...
[*] Setting up output directories...
[*] Scanning './in'...
[+] No auto-generated dictionary tokens to reuse.
[*] Creating hard links for all input files...
[*] Validating target binary...
[*] Attempting dry run with 'id:000000,orig:1.seed'...
[*] Spinning up the fork server...
[+] All right - fork server is up.
    len = 5, map size = 130, exec speed = 57277 us
[+] All test cases processed.

[!] WARNING: The target binary is pretty slow! See docs/perf_tips.txt.
[+] Here are some useful stats:
```



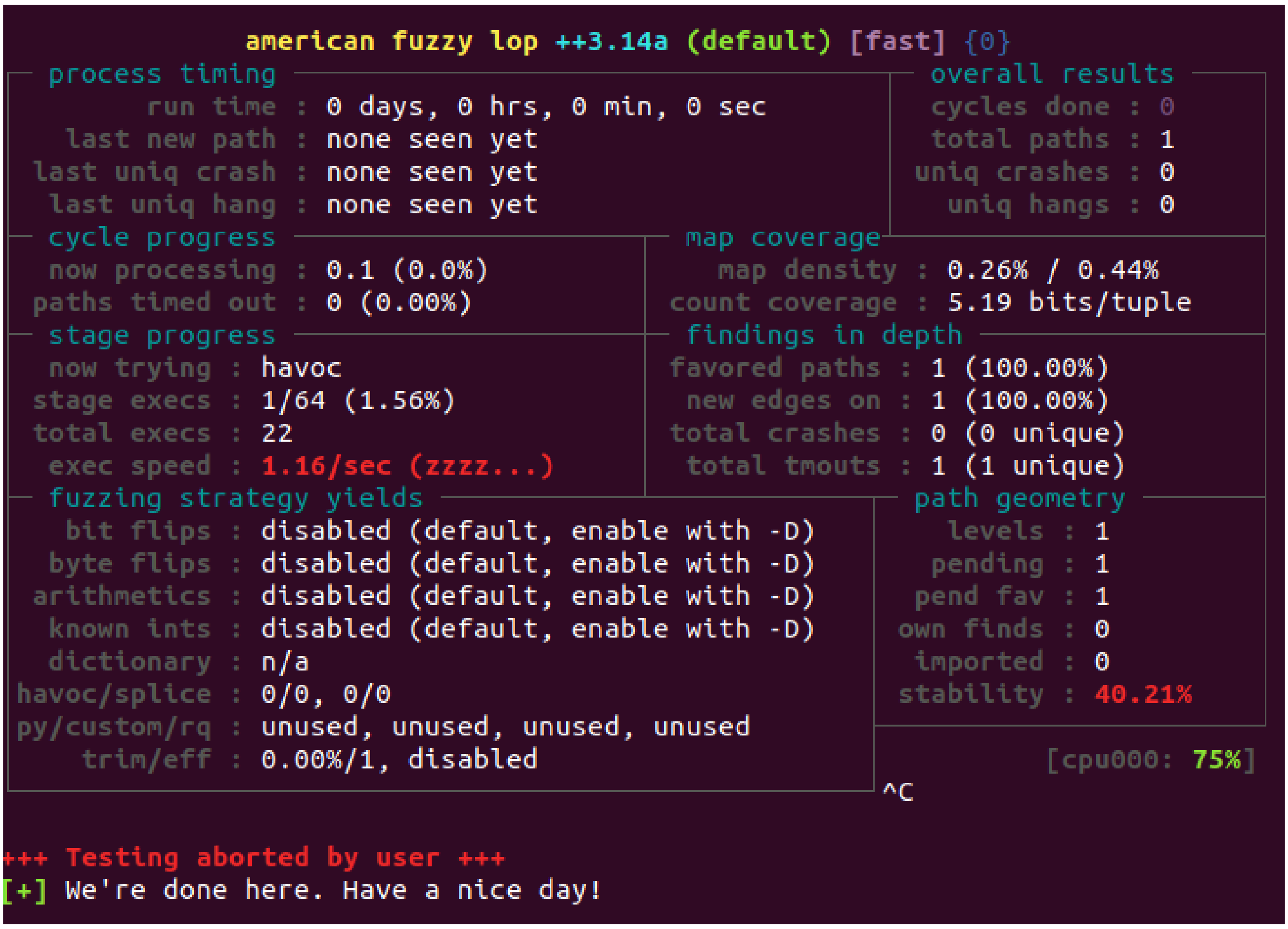
```
                    american fuzzy lop 2.52b (CROMU_00001)

┌─ process timing ─────────────────────────────────────┬─ overall results ─────┐
│        run time : 0 days, 0 hrs, 0 min, 3 sec        │  cycles done : 0      │
│   last new path : none seen yet                      │  total paths : 1      │
│ last uniq crash : none seen yet                      │ uniq crashes : 0      │
│  last uniq hang : 0 days, 0 hrs, 0 min, 0 sec        │   uniq hangs : 1      │
├─ cycle progress ────────────────────┬─ map coverage ─┴───────────────────────┤
│  now processing : 0* (0.00%)        │    map density : 0.20% / 0.20%         │
│ paths timed out : 0 (0.00%)         │ count coverage : 1.00 bits/tuple       │
├─ stage progress ────────────────────┼─ findings in depth ────────────────────┤
│  now trying : bitflip 1/1           │ favored paths : 0 (0.00%)              │
│ stage execs : 36/40 (90.00%)        │  new edges on : 1 (100.00%)            │
│ total execs : 47                    │ total crashes : 0 (0 unique)           │
│  exec speed : 2.55/sec (zzzz...)    │  total tmouts : 5 (1 unique)           │
├─ fuzzing strategy yields ───────────┴───────────────┬─ path geometry ────────┤
│   bit flips : 0/0, 0/0, 0/0                         │    levels : 1          │
│  byte flips : 0/0, 0/0, 0/0                         │   pending : 1          │
│ arithmetics : 0/0, 0/0, 0/0                         │  pend fav : 0          │
│  known ints : 0/0, 0/0, 0/0                         │ own finds : 0          │
│  dictionary : 0/0, 0/0, 0/0                         │  imported : n/a        │
│       havoc : 0/0, 0/0                              │ stability : 100.00%    │
│        trim : 0.00%/1, n/a                          ├────────────────────────┘
^C────────────────────────────────────────────────────┘          [cpu000: 32%]
```

Alphuzzplusplus

```
$ /Alphuzzplusplus-main/afl-fuzz -i ./in -o ./out -Q -- /CGC/CROMU/CROMU_00001
```



```
root@5ec91370378f:/# ./Alphuzzplusplus-main/afl-fuzz -i ./in -o ./out -Q ./CGC/C
ROMU/CROMU_00001
afl-fuzz++3.14a based on afl by Michal Zalewski and a large online community
[+] afl++ is maintained by Marc "van Hauser" Heuse, Heiko "hexcoder" Eißfeldt, A
ndrea Fioraldi and Dominik Maier
[+] afl++ is open source, get it at https://github.com/AFLplusplus/AFLplusplus
[+] NOTE: This is v3.x which changes defaults and behaviours - see README.md
[+] No -M/-S set, autoconfiguring for "-S default"
[*] Getting to work...
[+] Using exponential power schedule (FAST)
[+] Enabled testcache with 50 MB
[*] Checking core_pattern...
[!] WARNING: Could not check CPU scaling governor
[+] You have 8 CPU cores and 4 runnable tasks (utilization: 50%).
[+] Try parallel jobs - see docs/parallel_fuzzing.md.
[*] Setting up output directories...
[*] Checking CPU core loadout...
[+] Found a free CPU core, try binding to #0.
[*] Scanning './in'...
[+] Loaded a total of 1 seeds.
[*] Creating hard links for all input files...
[*] Validating target binary...
[*] No auto-generated dictionary tokens to reuse.
[*] Attempting dry run with 'id:000000,time:0,orig:1.seed'...
[*] Spinning up the fork server...
[+] All right - fork server is up.
[!] WARNING: instability detected during calibration
```



```
            american fuzzy lop ++3.14a (default) [fast] {0}
┌─ process timing ────────────────────────────────────┬─ overall results ────┐
│        run time : 0 days, 0 hrs, 0 min, 0 sec       │  cycles done : 0     │
│   last new path : none seen yet                     │  total paths : 1     │
│ last uniq crash : none seen yet                     │ uniq crashes : 0     │
│  last uniq hang : none seen yet                     │   uniq hangs : 0     │
├─ cycle progress ─────────────────┬─ map coverage─┴──────────────────────┤
│  now processing : 0.1 (0.0%)     │    map density : 0.26% / 0.44%       │
│ paths timed out : 0 (0.00%)      │ count coverage : 5.19 bits/tuple     │
├─ stage progress ─────────────────┼─ findings in depth ──────────────────┤
│  now trying : havoc              │ favored paths : 1 (100.00%)          │
│ stage execs : 1/64 (1.56%)       │  new edges on : 1 (100.00%)          │
│ total execs : 22                 │ total crashes : 0 (0 unique)         │
│  exec speed : 1.16/sec (zzzz...) │  total tmouts : 1 (1 unique)         │
├─ fuzzing strategy yields ────────┴───────────────┬─ path geometry ───────┤
│   bit flips : disabled (default, enable with -D) │    levels : 1         │
│  byte flips : disabled (default, enable with -D) │   pending : 1         │
│ arithmetics : disabled (default, enable with -D) │  pend fav : 1         │
│  known ints : disabled (default, enable with -D) │ own finds : 0         │
│  dictionary : n/a                                │  imported : 0         │
│havoc/splice : 0/0, 0/0                           │ stability : 40.21%    │
│py/custom/rq : unused, unused, unused, unused     ├───────────────────────┘
│    trim/eff : 0.00%/1, disabled                  │          [cpu000: 75%]
└──────────────────────────────────────────────────┘^C


+++ Testing aborted by user +++
[+] We're done here. Have a nice day!
```