

Project 3 I/O Monitoring (Linux)

Task 5.1

Task 5.1.1 Rotational Delay and IOPS of a 5400 RPM Drive

5.1) Rotational Delay :-

- RPM to RPS (Revolution Per Second) :-
$$RPS = \frac{5400}{60} = 90 \text{ RPS}$$
- Time taken per rotation :-
$$1 \text{ rotation} = \frac{1}{90} = 0.011 \text{ seconds}$$
$$1 \text{ rotation} \approx 11 \text{ ms}$$
- Rotational delay is half of time taken for 1 rotation :-
$$\text{Rotational delay} = \frac{11 \text{ ms}}{2}$$
$$\boxed{RD = 5.5 \text{ ms}}$$
- Add an average of 3 ms for seek time :-
$$\text{time} = 5.5 \text{ ms} + 3 \text{ ms}$$
$$= 8.5 \text{ ms}$$
- Add 2 ms for latency (internal transfer) :-
$$\text{time} = 8.5 + 2 \text{ ms} = 10.5 \text{ ms}$$
- IOPS
$$IOPS = \frac{1}{\text{avg. latency} + \text{avg. seek time}}$$
$$\boxed{\text{Avg. IOPS} = \frac{1}{10.5 \times 10^{-3}} \approx 95 \text{ IOPS}}$$

Task 5.2

Task 5.2.1 Monitoring Behavior with IOSTAT

Output 1

Device	qm	w_wait	wareq-sz	r/s	rKB/s	rrqm/s	%rrqm	r_wait	rareq-sz	w/s	wKB/s	wrqm/s	%wr
%util	d/s	dkB/s	drqm/s	%drqm	d_wait	dareq-sz	f/s	f_wait	aq-sz				
loop0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.21	0.00	0.00	0.00	0.00
00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00													
loop1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.20	12.78	0.00	0.00	0.00
00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01													
loop2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.38	18.73	0.00	0.00	0.00
00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00													
loop3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09	12.93	0.00	0.00	0.00
00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01													
loop4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.16	13.78	0.00	0.00	0.00
00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01													
loop5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	4.73	0.00	0.00	0.00
00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01													
loop6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	8.97	0.00	0.00	0.00
00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00													
loop7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.21	32.89	0.00	0.00	0.00
00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.03													
loop8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.27	0.00	0.00	0.00	0.00
00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00													
vda	77	0.56	35.18	6.49	234.96	2.31	26.27	0.23	36.18	14.92	524.83	13.11	46.01
0.61													

Efficiency for Read IOPS

$$\text{Efficiency for Read IOPS} = \frac{rKB/s}{r/s}$$
$$\text{Efficiency for Read IOPS} = \frac{234.96}{6.49} = 36.20$$

Efficiency for Write IOPS

$$\text{Efficiency for Write IOPS} = \frac{wKB/s}{w/s}$$
$$\text{Efficiency for Write IOPS} = \frac{524.83}{14.92} = 35.17$$

Output 2

Device	qm	w_wait	wareq-sz	r/s	rKB/s	rrqm/s	%rrqm	r_wait	rareq-sz	w/s	wKB/s	wrqm/s	%wr
%util				d/s	dkB/s	drqm/s	%drqm	d_wait	dareq-sz	f/s	f_wait	aq-sz	
loop0	00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.21	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
loop1	00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	12.78	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
loop2	00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	18.73	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
loop3	00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	12.93	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
loop4	00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	13.78	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
loop5	00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.73	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
loop6	00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	8.97	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
loop7	00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	32.89	0.00	0.00	0.00	0.00
0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
loop8	00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.27	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
vda	66	0.56	34.98	5.69	205.93	2.03	26.27	0.23	36.18	13.16	460.51	11.52	46.19
0.54													

Efficiency for Read IOPS

$$\text{Efficiency for Read IOPS} = \frac{rKB/s}{r/s}$$

$$\text{Efficiency for Read IOPS} = \frac{205.93}{5.69} = 36.19$$

Efficiency for Write IOPS

$$\text{Efficiency for Write IOPS} = \frac{wKB/s}{w/s}$$

$$\text{Efficiency for Write IOPS} = \frac{460.51}{13.16} = 34.99$$

Data Written

As the efficiency for the Write IOPS decreases ever so slightly, we can say that the amount of data written has decreased. Although the change is very small, on a general level if the efficiency decreases, it means that the amount of data being written is decreasing.

Loop Devices

```
sunaam@sunaam-QEMU-Virtual-Machine:~$ df -kh /dev/loop*
Filesystem      Size  Used Avail Use% Mounted on
/dev/loop0      128K  128K    0 100% /snap/bare/5
/dev/loop1       69M   69M    0 100% /snap/core22/867
/dev/loop2      227M  227M    0 100% /snap/firefox/3259
/dev/loop3      476M  476M    0 100% /snap/gnome-42-2204/143
/dev/loop4       12M   12M    0 100% /snap/snap-store/963
/dev/loop5       92M   92M    0 100% /snap/gtk-common-themes/1535
/dev/loop6      512K  512K    0 100% /snap/snapd-desktop-integration/85
/dev/loop7       36M   36M    0 100% /snap/snapd/20298
udev            1.9G    0  1.9G   0% /dev
udev            1.9G    0  1.9G   0% /dev
```

Task 5.2.2 Monitoring Behavior with IOTOP

IOTOP with Script Running

```
sunaam@sunaam-QEMU-Virtual-Machine: ~
sunaam@sunaam-QEMU-Virtual-Machine: ~/Desktop

Total DISK READ:      0.00 B/s | Total DISK WRITE:      7.21 M/s
Current DISK READ:    0.00 B/s | Current DISK WRITE:     31.53 M/s
  PID  PRIO  USER      DISK READ  DISK WRITE  SWAPIN     IO>   COMMAND
 17057 ?sys sunaam      0.00 B/s   7.21 M/s   ?unavailable? dd if=/dev/zero ~00000 oflag=dsync

keys: any: refresh q: quit i: ionice o: all p: threads a: accum
sort: r: asc left: SWAPIN right: COMMAND home: PID end: COMMAND
```

IOTOP when Script is killed

```
sunam@sunaam-QEMU-Virtual-Machin... x sunam@sunaam-QEMU-Virtual-Machin... x sunam@sunaam-QEMU-Virtual-Machin... x
total DISK READ:      0.00 B/s | Total DISK WRITE:      0.00 B/s
current DISK READ:    0.00 B/s | current DISK WRITE:    0.00 B/s
  PID  PRIO  USER      DISK READ  DISK WRITE  SWAPIN     IO>   COMMAND
I

keys: any: refresh q: quit i: ionice o: all p: threads a: accum
sort: r: asc left: SWAPIN right: COMMAND home: PID end: COMMAND
```

Task 5.3

Task 5.3.1

A ext4 RAM Disk is created at /mnt/ramdisk.

sunaam@sunaam-QEMU-Virtual-Machine: /mnt/ramdisk

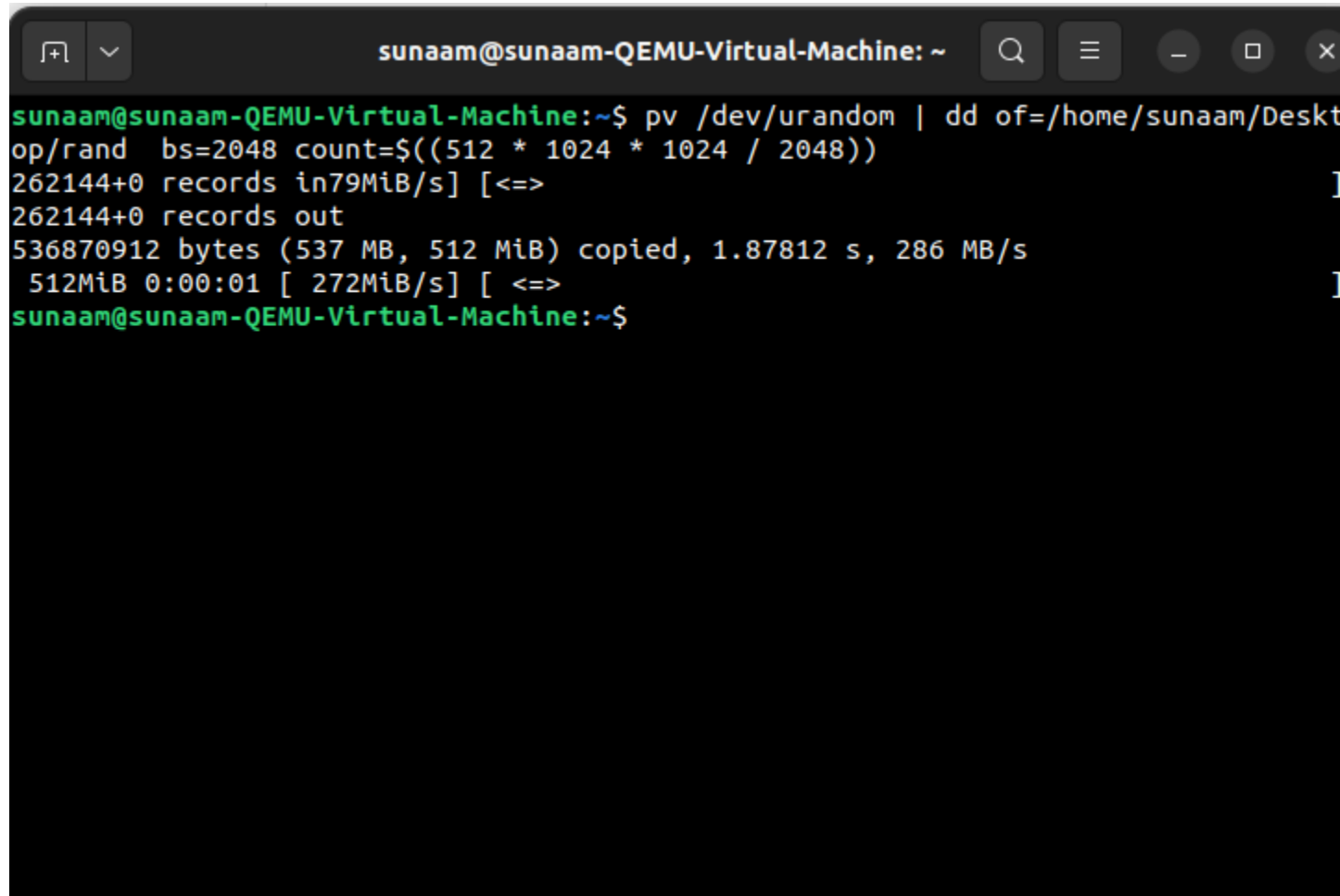
```
sunaam@sunaam-QEMU-Virtual-Machine:/mnt/ramdisk$ df -Th
Filesystem      Type  Size  Used Avail Use% Mounted on
tmpfs           tmpfs 391M  1.8M 389M   1% /run
/dev/vda2       ext4   63G   14G   46G  23% /
tmpfs           tmpfs 2.0G    0 2.0G   0% /dev/shm
tmpfs           tmpfs 5.0M  4.0K 5.0M   1% /run/lock
/dev/vda1       vfat   512M   5.3M 507M   2% /boot/efi
tmpfs           tmpfs 391M 116K 391M   1% /run/user/1000
ext4            tmpfs 1.0G    0 1.0G   0% /mnt/ramdisk
sunaam@sunaam-QEMU-Virtual-Machine:/mnt/ramdisk$
```


Task 5.3.2

Transfer in RAM DISK

```
sunam@sunam-QEMU-Virtual-Machine: ~  
sunam@sunam-QEMU-Virtual-Machine:~$ pv /dev/urandom | dd of=/mnt/ramdisk/rand  
bs=2048 count=$((512 * 1024 * 1024 / 2048))  
262144+0 records in79MiB/s] [<=>  
262144+0 records out  
536870912 bytes (537 MB, 512 MiB) copied, 1.30812 s, 410 MB/s  
512MiB 0:00:01 [ 382MiB/s] [ <=>  
sunam@sunam-QEMU-Virtual-Machine:~$
```

Transfer in Other DISK

A terminal window titled 'sunaam@sunaam-QEMU-Virtual-Machine: ~' with standard window controls. The terminal shows a command to copy data from /dev/urandom to a file in the desktop directory using pv and dd. The output shows the transfer of 537 MB at a speed of 286 MB/s.

```
sunaam@sunaam-QEMU-Virtual-Machine:~$ pv /dev/urandom | dd of=/home/sunaam/Desktop/
op/rand bs=2048 count=$((512 * 1024 * 1024 / 2048))
262144+0 records in79MiB/s] [ <=>
262144+0 records out
536870912 bytes (537 MB, 512 MiB) copied, 1.87812 s, 286 MB/s
 512MiB 0:00:01 [ 272MiB/s] [ <=>
sunaam@sunaam-QEMU-Virtual-Machine:~$
```

Conclusion

In the first screenshot, the data is being transferred to a tmpfs filesystem mounted at /mnt/ramdisk. As tmpfs is stored in RAM, the data transfer speed is faster compared to traditional disk-based filesystems.

In the second screenshot, the data is being transferred to a regular filesystem located at /home/student. The performance here may be influenced by the characteristics of the underlying storage device (HDD or SSD) where /home is located. It is slower than the RAM DISK.

The tmpfs-based transfer (/mnt/ramdisk/rand) shows significantly faster transfer speeds and shorter elapsed times, it suggests that utilizing tmpfs (RAM) for certain types of operations can offer better performance, especially for temporary or frequently accessed data.

Task 5.4

Installed AFL

american fuzzy lop 2.57b (fuzzgoat)		
process timing		overall results
run time : 0 days, 0 hrs, 3 min, 53 sec		cycles done : 1000000
last new path : 0 days, 0 hrs, 0 min, 12 sec		total paths : 1000000
last uniq crash : 0 days, 0 hrs, 0 min, 30 sec		uniq crashes : 1000000
last uniq hang : none seen yet		uniq hangs : 1000000
cycle progress	map coverage	
now processing : 15 (11.54%)	map density : 0.05% / 0.51%	
paths timed out : 0 (0.00%)	count coverage : 2.04 bits/tuple	
stage progress	findings in depth	
now trying : arith 8/8	avored paths : 51 (39.23%)	
stage execs : 52/266 (19.55%)	new edges on : 66 (50.77%)	
total execs : 42.4k	total crashes : 36 (7 unique)	
exec speed : 246.8/sec	total tmouts : 13 (1 unique)	
fuzzing strategy yields	path geometry	
bit flips : 13/192, 1/187, 2/177	levels : 3	
byte flips : 0/24, 0/19, 1/9	pending : 126	
arithmetics : 14/1110, 0/41, 0/0	pend fav : 48	
known ints : 1/98, 0/448, 1/352	own finds : 129	
dictionary : 0/0, 0/0, 0/0	imported : n/a	
havoc : 103/38.7k, 0/0	stability : 100	
trim : 40.00%/5, 0.00%		
[cpu00]		

Perf Report

```
Command Prompt
sunam@DESKTOP-0S2HDS7: X
sunam@DESKTOP-0S2HDS7: X
+ v

sunam@DESKTOP-0S2HDS7:~$ perf report -i perf.data
# To display the perf.data header info, please use --header/--header-only options.
#
#
# Total Lost Samples: 0
#
# Samples: 635K of event 'cycles'
# Event count (approx.): 635821000
#
# Children      Self  Command      Shared Object      Symbol
# .....      .....
```

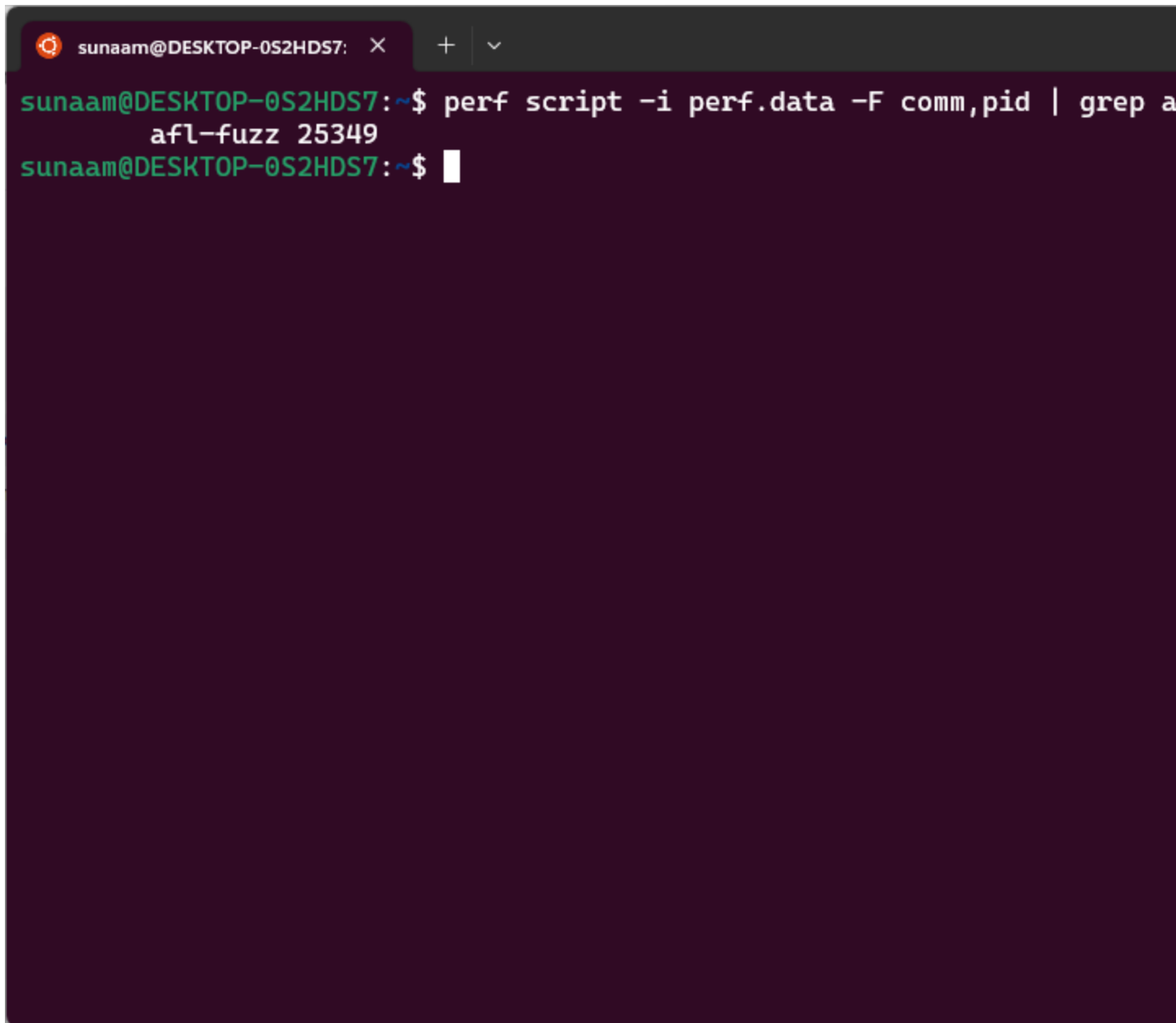
88.87%	88.77%	afl-fuzz	afl-fuzz	[.] run_target
	79.29%	0		
		run_target		
	9.51%	run_target		
79.38%	0.00%	afl-fuzz	[unknown]	[.] 0000000000000000
	0			
	79.36%	run_target		
2.46%	1.83%	fuzzgoat	libc.so.6	[.] read
	2.46%	read		

Perf Script

```
Command Prompt x sunaam@DESKTOP-0S2HDS7: x sunaam@DESKTOP-0S2HDS7: x
afl-fuzz 3409 104.734067: 1000 cycles:
          ffffffff82000b40 [unknown] ([unknown])
          7f6df1c2a290 [unknown] (/usr/lib/x86_64-linux-gnu/ld-linux-
afl-fuzz 3409 104.734085: 1000 cycles:
          7f6df1c2b030 [unknown] (/usr/lib/x86_64-linux-gnu/ld-linux-
afl-fuzz 3409 104.734094: 1000 cycles:
          7f6df1c2b055 [unknown] (/usr/lib/x86_64-linux-gnu/ld-linux-
          7f6df1c2a298 [unknown] (/usr/lib/x86_64-linux-gnu/ld-linux-
afl-fuzz 3409 104.734100: 1000 cycles:
          7f6df1c2b05c [unknown] (/usr/lib/x86_64-linux-gnu/ld-linux-
          7f6df1c2a298 [unknown] (/usr/lib/x86_64-linux-gnu/ld-linux-
afl-fuzz 3409 104.734108: 1000 cycles:
          7f6df1c2b078 [unknown] (/usr/lib/x86_64-linux-gnu/ld-linux-
          7f6df1c2a298 [unknown] (/usr/lib/x86_64-linux-gnu/ld-linux-
afl-fuzz 3409 104.734115: 1000 cycles:
          7f6df1c2b078 [unknown] (/usr/lib/x86_64-linux-gnu/ld-linux-
          7f6df1c2a298 [unknown] (/usr/lib/x86_64-linux-gnu/ld-linux-
afl-fuzz 3409 104.734121: 1000 cycles:
          7f6df1c2b07f [unknown] (/usr/lib/x86_64-linux-gnu/ld-linux-
          7f6df1c2a298 [unknown] (/usr/lib/x86_64-linux-gnu/ld-linux-
:
```

Identify PID of AFL-Fuzz from Perf Script

Command: `$ perf script -i perf.data -F comm,pid | grep afl-fuzz | head -1`

A terminal window with a dark background and light green text. The window title bar shows 'sunaam@DESKTOP-0S2HDS7:'. The terminal content shows a user running 'perf script -i perf.data -F comm,pid | grep afl-fuzz' and then 'afl-fuzz 25349'. The prompt returns to the shell.

```
sunaam@DESKTOP-0S2HDS7:~$ perf script -i perf.data -F comm,pid | grep a
afl-fuzz 25349
sunaam@DESKTOP-0S2HDS7:~$
```

Filter Unrelated Samples

Command: `$ perf script -i perf.data | grep afl-fuzz > perf_afl_fuzz.out`

```
sunaam@DESKTOP-0S2HDS7: X + v
# To display the perf.data header info, please use --header/--header-on
#
#
# Total Lost Samples: 0
#
# Samples: 290K of event 'cycles'
# Event count (approx.): 290588000
#
# Children      Self  Command      Shared Object      Symbol
# .....      .....      .....      .....      .....
#
# 86.07%      85.97%  afl-fuzz  afl-fuzz      [...] run_target
#      |
#      |--77.51%--0
#      |          run_target
#      |
#      |--8.48%--run_target
#
# 77.62%      0.00%  afl-fuzz  [unknown]      [...] 0000000000000000
#      |
#      ---0
#      |
#      |--77.58%--run_target
#
# 2.84%      2.22%  fuzzgoat  libc.so.6      [...] read
#      |
#      |--2.84%--read
#
#
```

Heavily Used Functions

The top functions are the most heavily used functions.



sunaam@DESKTOP-052HDS7: X



|
---0x56417419b2b0

0.36%	0.35%	fuzzgoat	fuzzgoat	[.]	main
0.34%	0.34%	afl-fuzz	[unknown]	[k]	0xfffffffff820
0.31%	0.00%	fuzzgoat	[unknown]	[.]	00000000000000
0.30%	0.30%	fuzzgoat	[unknown]	[k]	0xfffffffff820
0.28%	0.00%	fuzzgoat	[unknown]	[.]	0x00000000000000
0.23%	0.18%	fuzzgoat	fuzzgoat	[.]	__afl_store
0.23%	0.18%	fuzzgoat	libc.so.6	[.]	_Fork
0.19%	0.00%	afl-fuzz	[unknown]	[.]	0x677a7a75662
0.18%	0.15%	fuzzgoat	libc.so.6	[.]	__close
0.17%	0.00%	afl-fuzz	[unknown]	[.]	0x000055ead61
0.16%	0.15%	fuzzgoat	fuzzgoat	[.]	__afl_return
0.14%	0.14%	fuzzgoat	fuzzgoat	[.]	0x00000000000000
0.14%	0.00%	fuzzgoat	fuzzgoat	[.]	0x00005641741
0.12%	0.12%	afl-fuzz	afl-fuzz	[.]	save_if_inter
0.12%	0.12%	fuzzgoat	fuzzgoat	[.]	0x00000000000000
0.12%	0.00%	fuzzgoat	fuzzgoat	[.]	0x00005641741
0.11%	0.11%	afl-fuzz	libc.so.6	[.]	read
0.10%	0.10%	afl-fuzz	afl-fuzz	[.]	0x00000000000000
0.10%	0.00%	afl-fuzz	afl-fuzz	[.]	0x000055ead5a
0.09%	0.00%	afl-fuzz	afl-fuzz	[.]	0x000055ead5a
0.09%	0.09%	afl-fuzz	afl-fuzz	[.]	0x00000000000000
0.08%	0.08%	fuzzgoat	ld-linux-x86-64.so.2	[.]	__tunable_get
0.08%	0.00%	fuzzgoat	[unknown]	[.]	0x00007ffc83d
0.07%	0.00%	fuzzgoat	libc.so.6	[.]	0x00007ff9ff0
0.07%	0.06%	fuzzgoat	libc.so.6	[.]	malloc
0.07%	0.00%	fuzzgoat	libc.so.6	[.]	0x00007ff9ff0
0.07%	0.00%	fuzzgoat	libc.so.6	[.]	0x00007ff9ff0
0.06%	0.06%	fuzzgoat	libc.so.6	[.]	0x00000000000000
0.06%	0.06%	afl-fuzz	afl-fuzz	[.]	0x00000000000000
0.06%	0.00%	afl-fuzz	afl-fuzz	[.]	0x000055ead5a
0.06%	0.06%	fuzzgoat	libc.so.6	[.]	0x00000000000000
0.05%	0.05%	fuzzgoat	libc.so.6	[.]	wait4
0.05%	0.00%	afl-fuzz	afl-fuzz	[.]	0x000055ead5a
0.05%	0.00%	afl-fuzz	libc.so.6	[.]	0x00007f7bb59
0.05%	0.05%	afl-fuzz	libc.so.6	[.]	0x00000000000001
0.05%	0.05%	afl-fuzz	afl-fuzz	[.]	0x00000000000000
0.05%	0.00%	fuzzgoat	libc.so.6	[.]	0x00007ff9fef
0.05%	0.05%	afl-fuzz	[unknown]	[k]	0xfffffffff820
0.04%	0.00%	fuzzgoat	libc.so.6	[.]	0x00007ff9ff0
0.04%	0.04%	fuzzgoat	libc.so.6	[.]	0x00000000000000
0.04%	0.04%	fuzzgoat	libc.so.6	[.]	0x00000000000000
0.04%	0.00%	fuzzgoat	libc.so.6	[.]	0x00007ff9ff0
0.04%	0.04%	fuzzgoat	libc.so.6	[.]	0x00000000000000

Task 5.5

Automotive

```
6.248278722140 radians = 358 degrees
6.265732014660 radians = 359 degrees
6.283185307180 radians = 360 degrees
```

```
Performance counter stats for './basicm
```

50.52 msec	task-clock
511	context-switches
1	cpu-migrations
21	page-faults
<not supported>	cycles
<not supported>	instructions
<not supported>	branches
<not supported>	branch-misses

```
0.098760267 seconds time elapsed
```

```
0.021368000 seconds user
```

```
0.029915000 seconds sys
```

```
sunaam@sunaam-QEMU-Virtual-Machine:~/Dow
```

Network

```
sunaam@sunaam-QEMU-Virtual-Machine:~/Do
Usage: dijkstra <filename>
Only supports matrix size is #define'd.
./dijkstra_small: Segmentation fault

Performance counter stats for './dijks

              0.91 msec task-clock
                2      context-switches
                1      cpu-migrations
               28      page-faults
<not supported>      cycles
<not supported>      instructions
<not supported>      branches
<not supported>      branch-misses

0.076202250 seconds time elapsed

0.0000000000 seconds user
0.001791000 seconds sys
```

Security

```
sunaam@sunaam-QEMU-Virtual-Machine:~/Dow
Usage: blowfish {e|d} <input> <output>

Performance counter stats for './bf':

          0.72 msec task-clock
              0      context-switches
              0      cpu-migrations
             25      page-faults
<not supported>      cycles
<not supported>      instructions
<not supported>      branches
<not supported>      branch-misses

    0.001076098 seconds time elapsed

    0.000000000 seconds user
    0.001381000 seconds sys

sunaam@sunaam-QEMU-Virtual-Machine:~/Dow
```

```
sunaam@sunaam-QEMU-Virtual-Machine:~/Dow
```

```
Usage: fft <waves> <length> -i
```

```
-i performs an inverse fft
```

```
make <waves> random sinusoids<length> is
```

```
Performance counter stats for './fft':
```

```
1.64 msec task-clock
```

```
0 context-switches
```

```
0 cpu-migrations
```

```
21 page-faults
```

```
<not supported> cycles
```

```
<not supported> instructions
```

```
<not supported> branches
```

```
<not supported> branch-misses
```

```
0.001987738 seconds time elapsed
```

```
0.0000000000 seconds user
```

```
0.002347000 seconds sys
```

Office

```
seen" is not in "be associated with seen or  
"guide" is not in "and it will recommend  
"regard" is in "in this regard. The Univ  
"officers" is not in "Executive Officers  
"implement" is in "whether and how to im  
"principalities" is not in "principles."
```

```
Performance counter stats for './search'
```

```
          0.74 msec task-clock  
             0      context-switches  
             0      cpu-migrations  
            22      page-faults  
<not supported>    cycles  
<not supported>    instructions  
<not supported>    branches  
<not supported>    branch-misses
```

```
0.001132659 seconds time elapsed
```

```
0.001376000 seconds user
```

```
0.000000000 seconds sys
```

Questions

Do the programs have the same proportion of instructions?

The proportion of instructions may vary among programs based on their nature and functionality. Programs with different computational requirements, control flow structures, and memory access patterns are likely to exhibit different proportions of instructions. For example, compute-intensive programs may have a higher proportion of computational instructions, while programs with frequent branching may have a higher proportion of control flow instructions.

Do some programs have higher microarchitectural events than others?

Yes, different programs are likely to have varying microarchitectural events based on their characteristics. For instance, programs with complex control flow may exhibit higher branch miss rates, while programs with intensive memory access patterns may experience higher cache miss rates. The nature of the workload and algorithmic choices significantly influences the occurrence of microarchitectural events.

Can the processor keep its computational speed over the run? Are there periods that the processor becomes relatively slower? If there are, why?

The processor's computational speed may vary over the run of a program. Factors such as changes in workload intensity, memory access patterns, and external interference can influence the processor's speed. For example, if a program enters a phase with a high number of cache misses, the processor may experience slowdowns due to increased memory latency. Similarly, external factors like CPU contention with other processes or system events may lead to fluctuations in computational speed.

Do most programs have different phases? What metrics do you use? How many are they?

Many programs exhibit different phases during execution, characterized by variations in microarchitectural events. Metrics such as cache misses, branch misses, and CPU cycles are commonly used to identify program phases. The number of phases depends on the program's structure and workload. For instance, a sorting algorithm may have distinct phases for data input, sorting, and output, each with unique characteristics. The identification of phases can provide insights into the dynamic behavior of a program and help optimize specific code sections.