TOSHIBA

How to reduce the write io to SD Card on Debian

2019/03/08

Toshiba Memory Corporation
Institute of Memory Technology Research & Development
System Technology Research & Development Center
Masahiro Yamada

BiCS FLASH™

Today's agenda

BiCS FLASH™

01 Background

02 Problem

03 Approach

04 Evaluation

05 Conclusion and future work

Many embedded boards uses SD card as Boot Device

Conventional embedded board

- e.g. : On Raspberry Pi, raspbian installed SD card is bootable
- There is a limitation about the number of write io on SD Card
- If there is only a SD card as Boot device and the limitation is over, the board would not boot.

Linux Distribution like Debian

- There are many user unaware(unintended) write io
- If user don't take care of it, the lifespan of SD Card is not so long

BiCS FLASH™

The example of unintended write io

- While nginx.service is running, by the access to the web server or by the error caused by web server, below 2 files are written

```
$ tree /var/log/nginx
/var/log/nginx
├──── access.log
└──── error.log

0 directories, 2 files
```

- (To be extreme) The lifespan of SD card is consumed by the external access

Reduce the unintended write io AMAP, extend the lifespan

Complete workaround (shut out all wirte ios to SD card) :

- Mount rootfs as read only, and not write anything to SD card
- By NFS mount, the all write ios go external storage

Flexible workaround (shut out a part of write ios to SD card) :

- Make write io volatile or disable write io, whose frequency is high and the priority is low.
- After system reboot, it is acceptable precondition to disappear the previous write

Assumption of this presentation

- CPU is x86(64bit). Environment for evaluation is VM
- OS is Debian9.5, the kernel is v4.9
- RootFS is installed in SD card
- When system boot, sd card is mounted and is used as rootfs
- Use ext4 format as rootfs
- There are no external storages for writing

Categorize the cause of write, which can be likely shut out

- Problem 1 systemd service write log file periodically
- Problem 2 cache file
- Problem 3 the record of access time to file
- Problem 4 swap file
- Problem 5 find out unknown write process except problem 1-4

Problem 1 systemd service write log file periodically

- After system boot, many systemd services run at same time, these continue to run. Among these services, by default setting, many services, which user is not aware of, are active.
- You can check acive system service by below command

```
$ systemctl list-unit-files | grep enabled
```

- Those includes some services, which write the log file under /var/log.

Problem 2 cache file

- To speed up some kind of process, there are some programs which reserve cache file on disk (e.g. to reduce network access, use previous cache file)
- File like cache tend to increase the size, so it would be the stress of write to SD card
- Apt, package management system, is a conventional program uses cache file. Deb files, which were installed in system before, are stored under /var/cache/apt.

Problem 3 the record of access time to file

Ext4 has 3 meta data to each directories / files.

- Access time, Modify time, Change time

Access time is the last time to read/write the file.

The default mount option of ext4 is relatime

- Even if the access is read the file, if there is no access from 1 day ago to now, the write io would happen to update the access time.

Problem 4 swap file

- Swap is to increase free memory space by storing long-time unused memory to disk, before system cannot run by the lack of memory.
- Depending on memory size or application running on system, many swap will be caused frequently, so it would be the stress of SD card
- /etc/fstab includes swap setting, systemd-remount-fs.service mount swap space according to the setting on boot time.

Problem 5 find out unknown write process except problem 1-4

Except Problem 1～4, there are some process which write periodically or suddenly

- e.g. : timer process by cron or systemd
- e.g. : journal process for ext4 by kernel thread

Before you decide to disable these process, you need method that you find “who” write “where”

3 types of approach

Approach 1 make directory and file volatile by tmpfs (problem1,2)

Approach 2 disable some functions (problem3,4)

Approach 3 detect ”who” writes “where” (problem 5)

Approach 1 make directory and file volatile by tmpfs (1)

Below 2 systemd services can be used

- To create tmpfs by systemd-remount-fs.service(fstab)
- To replace dir/file on disk as dir/file on tmpfs by systemd-tmpfiles-setup.service

The target for volatile directories are various

- e.g. : /var/lock、/var/run、/var/tmp、/tmp、/run/systemd、…

But, to simplify this presentation, make /var/log and /var/cache/apt volatile in this time

Approach 1 make directory and file volatile by tmpfs (2)

Mount tmpfs to /var/volatile by fstab

```
$ cat /etc/fstab
#<file system>  <mount point>  <type>  <options>  <dump>  <pass>
...
tmpfs          /var/volatile   tmpfs   defaults           0       0
```

Make 2 directories volatile by systemd-tmpfiles-setup.service

```
$ cat /etc/tmpfiles.d/00_core.conf
d /var/volatile/log 0755 root root -
L+ /var/log 0755 root root - /var/volatile/log
d /var/volatile/cache/apt 0755 root root -
L+ /var/cache/apt 0755 root root - /var/volatile/cache/apt
```

Approach 1 make directory and file volatile by tmpfs (3)

Issue caused by this approach

- Some packages cannot run by making /var/log volatile
- E.g. : nginx has a precondition that /var/log/nginx exists on boot time, if it doesn't exist, the boot would fail

```
Sep 15 00:58:17 debian09 systemd[1]: Starting A high performance web server and a reverse proxy server...
Sep 15 00:58:17 debian09 nginx[4161]: nginx: [alert] could not open error log file: open() "/var/log/nginx/error.log" failed
Sep 15 00:58:17 debian09 nginx[4161]: 2018/09/15 00:58:17 [emerg] 4161#4161: open() "/var/log/nginx/access.log" failed (2:
Sep 15 00:58:17 debian09 nginx[4161]: nginx: configuration file /etc/nginx/nginx.conf test failed
Sep 15 00:58:17 debian09 systemd[1]: nginx.service: Control process exited, code=exited status=1
Sep 15 00:58:17 debian09 systemd[1]: Failed to start A high performance web server and a reverse proxy server.
Sep 15 00:58:17 debian09 systemd[1]: nginx.service: Unit entered failed state.
Sep 15 00:58:17 debian09 systemd[1]: nginx.service: Failed with result 'exit-code'.
```

Approach 1 make directory and file volatile by tmpfs (4)

workaround

- Create /var/log/nginx on system boot time

```
$ cat /etc/tmpfiles.d/50_nginx.conf
d /var/log/nginx 0755 root adm - none
f /var/log/nginx/access.log 0640 www-data adm - none
f /var/log/nginx/error.log 0640 www-data adm - none
```

There are a lot of packages like nginx, which cannot run by approach1.

- See Appendix. A for reference to packages like that

Approach 1 make directory and file volatile by tmpfs (5)

Important point by this approach

- Memory usage. The amount of writing log would be the amount of memory used. Tmpfs can use memory until the limitation.

> size: The limit of allocated bytes for this tmpfs instance. The default is half of your physical RAM without swap. If you oversize your tmpfs instances the machine will deadlock since the OOM handler will not be able to free that memory.

- Packages which write log need to be checked whether there is a log rotate setting, or not. And check the frequency is enough.

Quotation：https://www.kernel.org/doc/Documentation/filesystems/tmpfs.txt

Approach 2 disable some functions (1)

Use noatime mount option by fstab

```
$ cat /etc/fstab
#<file system>  <mount point>  <type>  <options>           <dump>  <pass>
...
/dev/root       /              ext4    rw,noatime,data=ordered 0       0
```

Important point by this apporach

- Some kind of packages suppose that atime is updated correctly
- E.g. : mutt (mailer software)
- Between power on and remount, it is likely atime is updated.

Approach 2 disable some functions (2)

Disable swap by fstab

```
$ cat /etc/fstab
…
# swap was on /dev/sda5 during installation
#UUID=7f126736-bc08-4569-a3d0-049e1f09ebd7 none            swap    sw              0       0
```

Important point by this apporach

- Memory usage. With approach1, we have to consider the maximum memory usage system uses in advance.

Approach 3 detect "who" writes "where"

Use fanotify in this time

- Fanotify can monitor the event of filesystem
- By including <sys/fanotify.h>, each fanotify API can be used
- By using fanotify API, you can monitor all file operations under some mount point, and you can know the process id.
- Example program : https://manpages.debian.org/stretch/manpages/fanotify.7.en.html
- Monitoring write op and output the pid by patch(Appendix. B)

(Reference) Bcc-tools also can be used as fanotify (Appendix. C)

How to evaluate the write access to disk

Read stat file in block device of sysfs subsystem, and check write io

- If /dev/sda is boot device

```
$ cat /sys/block/sda/stat
   11976     2145   680990    12200      212      383     5320      340        0     4704    12532
```

- The fifth parameter shows the number of write io

```
Name            units         description
----            -----         -----------
read I/Os       requests      number of read I/Os processed
read merges     requests      number of read I/Os merged with in-queue I/O
read sectors    sectors       number of sectors read
read ticks      milliseconds  total wait time for read requests
write I/Os      requests      number of write I/Os processed
```

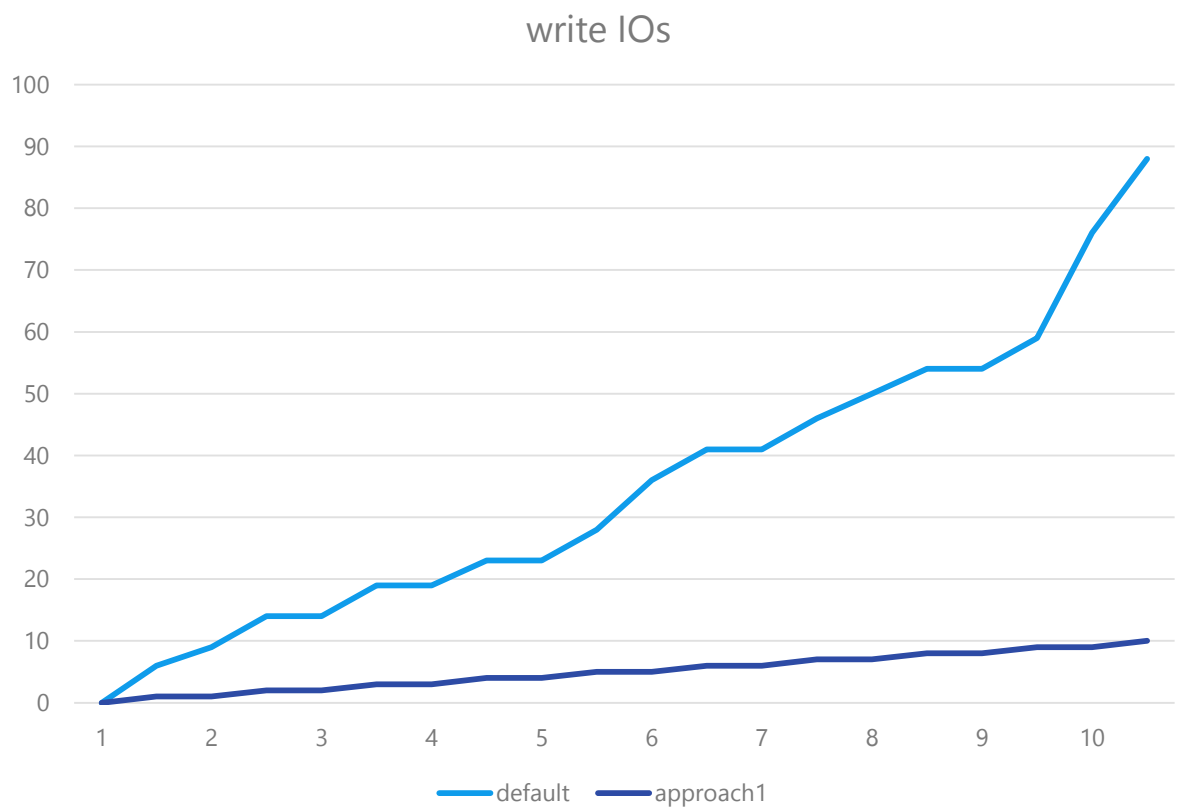
Quotation: https://www.kernel.org/doc/Documentation/block/stat.txt

BiCS FLASH™

The result of approach1 (volatile) evaluation for problem1 (/var/log)

Repeat below method 10 times

- Check the write io of stat file
- Access nginx by curl
- Do sync
- Check the write io of stat file
- Sleep 60 seconds

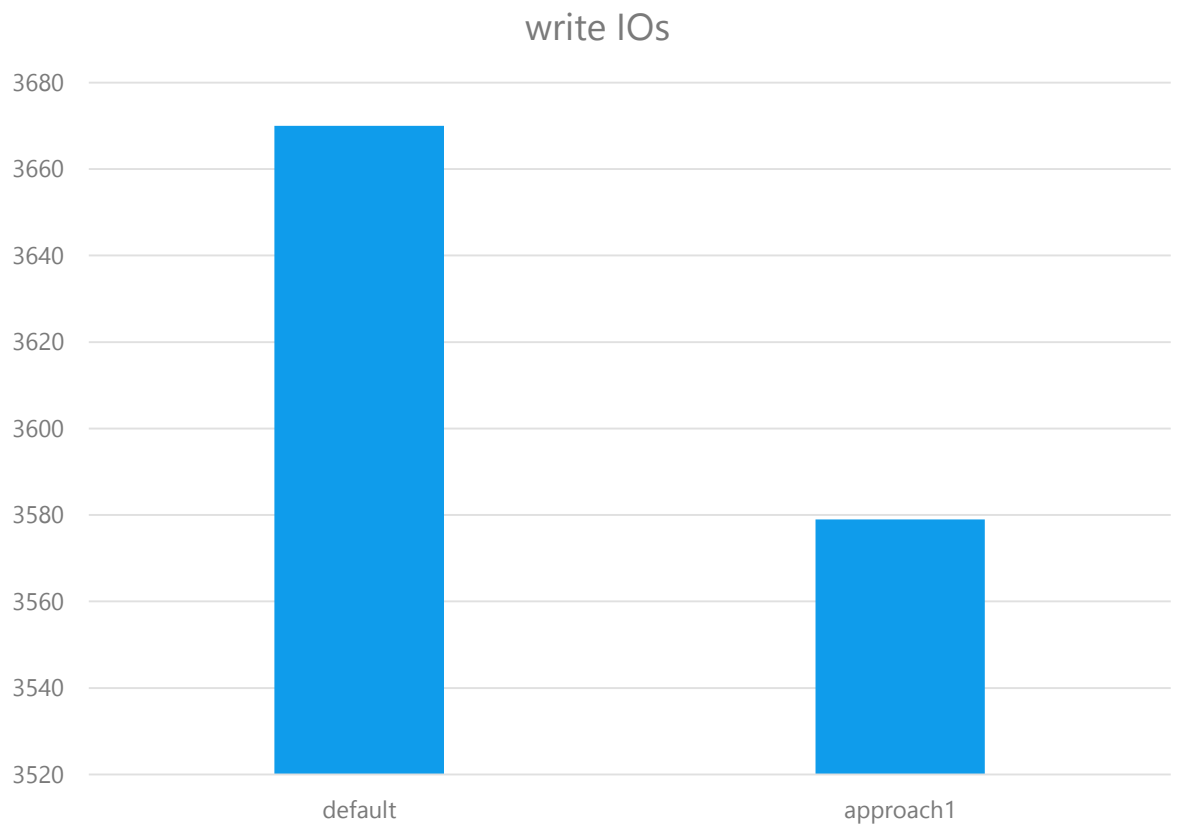


BiCS FLASH™

The result of approach1 (volatile) evaluation for problem2 (cache)

Run below method

- Check the write io of stat file
- Install vim by apt
- Do sync
- Check the write io of stat file



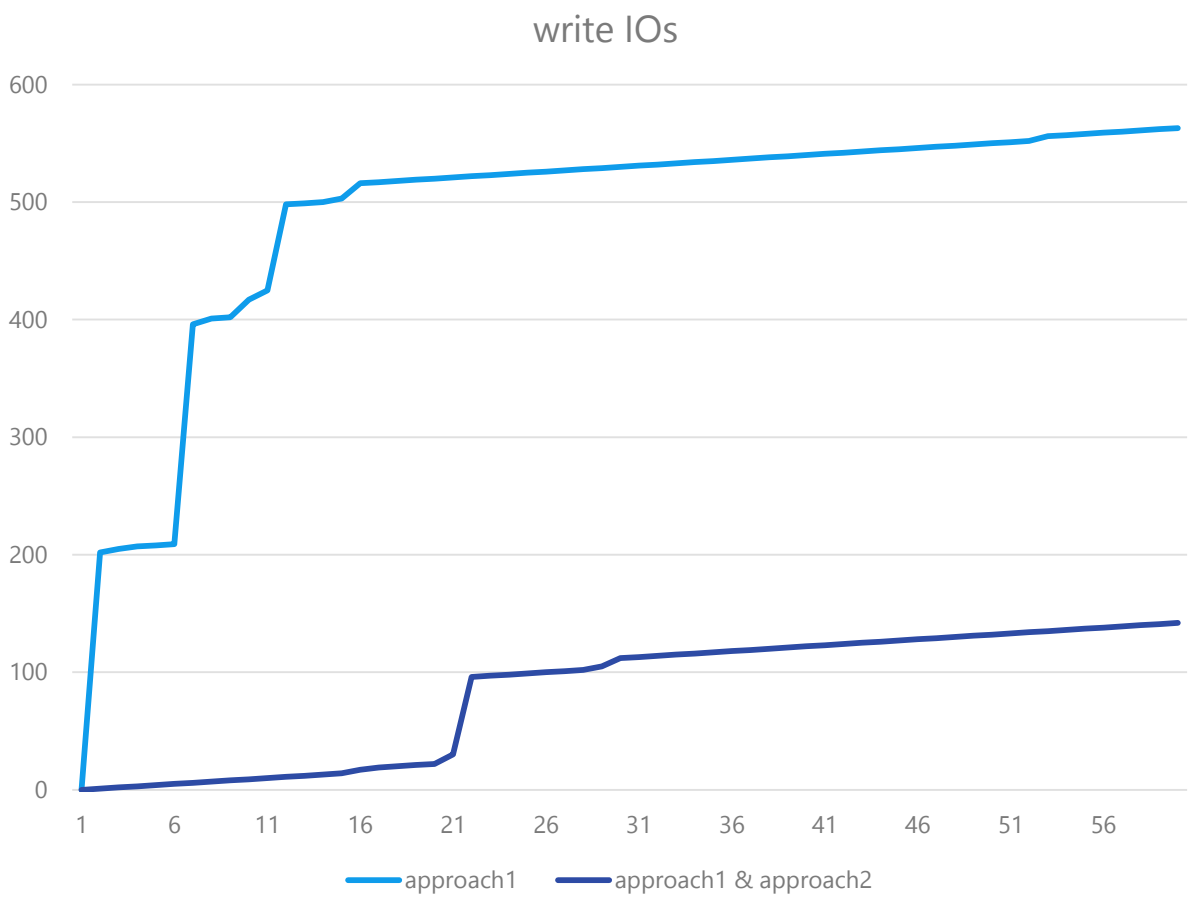
BiCS FLASH™

The result of approach2 (disable) evaluation for problem3 (atime)

To remove the influence of system service writing to /var/log, apply approach1(volatile).

Run below method

- Set system clock to tomorrow
- Do sync
- Run 60 times loop with below 3 methods
- Do sync
- Check the write io of stat file
- Sleep 1 minute



The result of approach2 (disable) evaluation for problem4 (swap)

- Before

```
# cat /proc/swaps
Filename                        Type            Size    Used    Priority
/dev/sda5                       partition       2095100 0       -1
```

- After

```
# cat /proc/swaps
Filename                        Type            Size    Used    Priority
```

The result of approach3 (monitoring) evaluation for problem5

- Monitoring root directory 1 hour by fanotify
- Apply approach1,2 to environment (not to detect known write)
- A part of result (See Appendix. D for mote details)

```
# fanotify_example /
...
FAN_OPEN_PERM: File /var/lib/upower/history-time-empty-50.dat.VTEKPZ (657)
FAN_MODIFY: File /var/lib/upower/history-time-empty-50.dat.VTEKPZ (657)
FAN_CLOSE_WRITE: File /var/lib/upower/history-time-empty-50.dat (657)
...
```

- PID 657 was the command by upower.service on my system

Consider approach1,2 can be applied to reduce more disk access

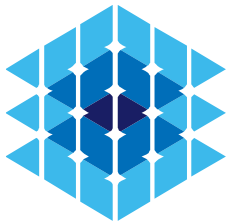
Conclusion and future work

Conclusion

- Reduce the write ios to SD card by making directory and file volatile and by disabling some features.
- Detect the unintended write process by monitoring write op

Future work

- Make it easy to fix packages, which cannot run by volatile.
- About Kdump, /var/log on memory would be deleted after panic, it is difficult to read the contents from the dump file.



BiCS FLASH™

Appendix

Appendix A. need to care packages when /var/log is tmpfs

BiCS FLASH™

Package / command	Directory / file
openssh-server	/var/run/sshd
nginx	/var/log/nginx /var/log/nginx/access.log /var/log/nginx/error.log
Uwsgi	/var/log/uwsgi /var/log/uwsgi/app /var/run/uwsgi
audit	/var/log/audit
dpkg	/var/log/dpkg
libvirt	/var/log/libvirt/qemu /var/log/libvirt/uml /var/log/libvirt/lxc
apache2	/var/log/apache2 /var/run/apache2
sysstat	/var/log/sa
Exim	/var/log/exim4

Appendix B. patch to fanotify example

BiCS FLASH™

```
$ diff -up fanotify_example.c fanotify_example_write.c
--- fanotify_example.c  2018-09-20 07:21:17.000000000 +0900
+++ fanotify_example_write.c    2018-09-20 13:18:34.328000000 +0900
@@ -78,6 +78,9 @@ handle_events(int fd)
                if (metadata->mask & FAN_CLOSE_WRITE)
                    printf("FAN_CLOSE_WRITE: ");

+               if (metadata->mask & FAN_MODIFY)
+                   printf("FAN_MODIFY: ");
+
                /* Retrieve and print pathname of the accessed file */

                snprintf(procfd_path, sizeof(procfd_path),
@@ -90,7 +93,7 @@ handle_events(int fd)
                }

                path[path_len] = '¥0';
-               printf("File %s¥n", path);
+               printf("File %s (%d)¥n", path, metadata->pid);

                /* Close the file descriptor of the event */

@@ -136,7 +139,7 @@ main(int argc, char *argv[])
         file descriptor */

     if (fanotify_mark(fd, FAN_MARK_ADD | FAN_MARK_MOUNT,
-                      FAN_OPEN_PERM | FAN_CLOSE_WRITE, AT_FDCWD,
+                      FAN_OPEN_PERM | FAN_CLOSE_WRITE | FAN_MODIFY, AT_FDCWD,
                       argv[1]) == -1) {
         perror("fanotify_mark");
         exit(EXIT_FAILURE);
```

Appendix C. example by using bcc-tools

BiCS FLASH™

Use biosnoop and filetop, which are examples attached Bcc-tools(https://github.com/iovisor/bcc)

- bcc-toolsのbiosnoop

```
TIME(s)       COMM         PID    DISK    T  SECTOR    BYTES   LAT(ms)
0.000000000   ?            0      R  -1       8        0.49
2.015770000   ?            0      R  -1       8        0.25
2.271162000   jbd2/sda1-8  152    sda     W  80805472  16384     0.42
2.275804000   jbd2/sda1-8  152    sda     W  80805504  4096      0.23
4.030423000   ?            0      R  -1       8        0.17
6.048356000   ?            0      R  -1       8        0.45
8.064035000   ?            0      R  -1       8        0.23
8.158926000   kworker/u2:0 17059  sda     W  2048      4096      0.22
8.159107000   kworker/u2:0 17059  sda     W  2080      4096      0.35
```

- bcc-toolsのfiletop

```
03:46:23 loadavg: 0.17 0.10 0.16 2/321 19463

TID    COMM             READS  WRITES R_Kb  W_Kb  T FILE
19463  clear            2      0      8     0     R xterm-256color
19452  python           2      0      2     0     R loadavg
19463  clear            1      0      0     0     R libtinfo.so.5.9
19463  clear            1      0      0     0     R libc-2.24.so
19463  python           3      0      0     0     R clear
19463  python           2      0      0     0     R ld-2.24.so
```

Appendix D. Monitoring "/" by fanotify_example

BiCS FLASH™

```
# fanotify_example /
...
FAN_MODIFY: File /var/lib/upower/history-rate-50.dat.W2VKPZ (657)
FAN_CLOSE_WRITE: File /var/lib/upower/history-rate-50.dat (657)
FAN_OPEN_PERM: File /var/lib/upower/history-charge-50.dat.VUCKPZ (657)
FAN_MODIFY: File /var/lib/upower/history-charge-50.dat.VUCKPZ (657)
FAN_CLOSE_WRITE: File /var/lib/upower/history-charge-50.dat (657)
FAN_OPEN_PERM: File /var/lib/upower/history-time-full-50.dat.DDHKPZ (657)
FAN_MODIFY: File /var/lib/upower/history-time-full-50.dat.DDHKPZ (657)
FAN_CLOSE_WRITE: File /var/lib/upower/history-time-full-50.dat (657)
FAN_OPEN_PERM: File /var/lib/upower/history-time-empty-50.dat.VTEKPZ (657)
FAN_MODIFY: File /var/lib/upower/history-time-empty-50.dat.VTEKPZ (657)
FAN_CLOSE_WRITE: File /var/lib/upower/history-time-empty-50.dat (657)
...
FAN_OPEN_PERM: File /var/lib/systemd/timers/stamp-anacron.timer (1)
FAN_CLOSE_WRITE: File /var/lib/systemd/timers/stamp-anacron.timer (1)
...
FAN_OPEN_PERM: File /etc/anacrontab (1343)
FAN_OPEN_PERM: File /var/spool/anacron/cron.daily (1343)
FAN_CLOSE_WRITE: File /var/spool/anacron/cron.daily (1343)
FAN_OPEN_PERM: File /var/spool/anacron/cron.weekly (1343)
FAN_CLOSE_WRITE: File /var/spool/anacron/cron.weekly (1343)
FAN_OPEN_PERM: File /var/spool/anacron/cron.monthly (1343)
FAN_CLOSE_WRITE: File /var/spool/anacron/cron.monthly (1343)
...
FAN_CLOSE_WRITE: File /tmp/tmpfdrUyvr (deleted) (1356)
FAN_CLOSE_WRITE: File /tmp/tmpfdrUyvr (deleted) (1356)
...
```

Appendix E. Reference

- https://wiki.debian.org/SSDOptimization
- http://blog.nunosenica.com/reduce-write-operations-to-sd-card-with-raspbian/
- https://www.makeuseof.com/tag/extend-life-raspberry-pis-sd-card/
- https://www.kernel.org/doc/Documentation/filesystems/tmpfs.txt
- https://www.kernel.org/doc/Documentation/block/stat.txt
- http://hallard.me/raspberry-pi-read-only/