EXT2-Aware Shingled Translation Layer for Autonomous SWD

1. System Architecture

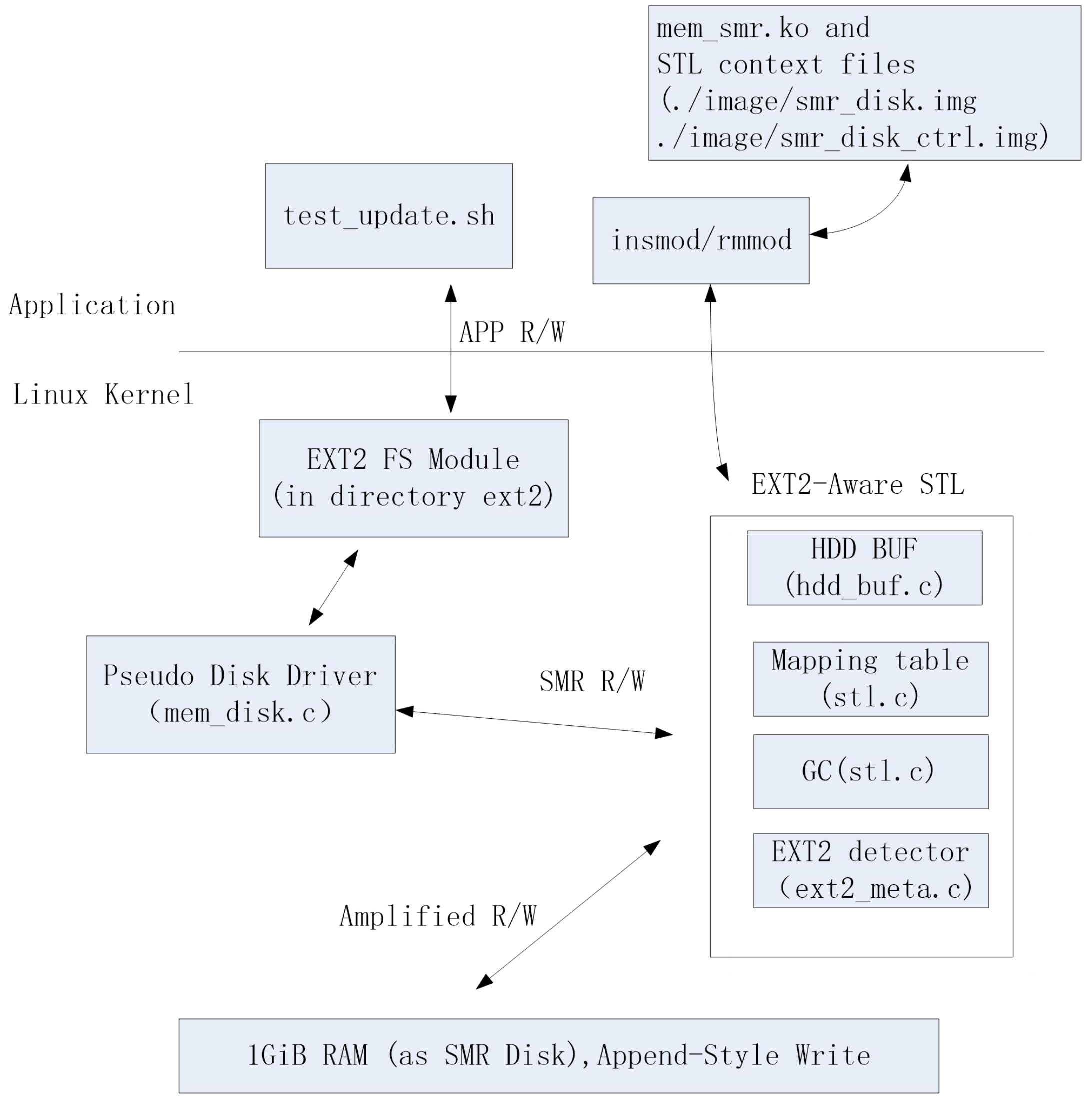

Figure 1: System Architecture

Basic Idea:

(1) 64MiB HDD + 1GiB SWD

HDD is used as a write BUFFER, that is, a random write area while SWD as append-style write area.

Any NEW WRITE (no mapping yet) is appended to SMR Disk directly, bypassing the HDD BUF. For those write-once-read-many files, they don't need to be written to buffer and then swapped out to SMR Disk. Large Videos are good examples for such files.

(This emulator could also be used to emulate 64MibPCM + 1GiB MLC NAND FLASH)

(2) an ext2-aware detector

As an autonomous SWD, if all the Logical space has been written, it will assume that all the mapping entries are valid even though the corresponding files have been deleted by file system. In this case, there is not any NEW WRITE later and the SWD is running as if it were occupied by full of data. This would cause high write amplification. In this view, a mechanism to detect the blocks released by FS is necessary. In order to avoid any change in the existing host I/O software, we try to implement a FS-aware detector in the device-side. We use the EXT2 as an example.

A Definite Finite Automata is designed to dynamically detect the EXT2 file system and mark the Meta Data(super block, block group descriptors, inode table, block bitmap) for further statistics .

After that, we could detect the free blocks released by EXT2 at run time. To be conservative, we only actually free the blocks when SWD is powered on in this version.

2. Evaluation

We format the 1GiB SWD into EXT2, which can hold at most 14508 files (64KiB each). The block size we choose is 4KiB and each file consists of 16 blocks. The total file data is 64KiB*14508 while the other space is used for FS meta data.

(1) Absolutely Random Write

Take 15% as an example.

When the usage is 15%, there are 2176 files. Each iteration, we first generate 2176 different random values then update a random block of each file. In other words, we update data blocks in ABSOLUTELY RANDOM order. This procedure is repeated 320 times.

The total data updated is about 3*64KiB*14508 for different usage, which is triple of the file data the SWD can accommodate.

Even in such an random access sequence, the write amplification is still acceptable.

Table-1 Write Amplification at different usage of 1GiB disk

usage	number of files	repeat	APP Write (blocks)	SMR Write (blocks)	Amplifiled_R&W (blocks)	Write Amplicfication
15%	2176	320	696320	583099	882875	1.514
20%	2901	240	696240	630313	1025279	1.627
25%	3627	192	696384	646870	1096828	1.696
30%	4352	160	696320	654623	1149615	1.756
35%	5077	137	695549	659414	1200726	1.821
40%	5803	120	696360	668048	1252362	1.875
45%	6528	106	691968	671438	1299530	1.935
50%	7254	96	696384	683156	1371176	2.007
55%	7979	87	694173	694158	1453234	2.094
60%	8704	80	696320	692159	1535161	2.218
65%	9430	73	688390	687707	1636183	2.379
70%	10155	68	690540	696548	1795674	2.578
75%	10881	64	696384	703785	1993979	2.833
80%	11606	60	696360	707571	2254939	3.187
85%	12331	56	690536	700644	2593672	3.702
90%	13057	53	692021	715998	3188056	4.453
95%	13782	50	689100	703338	3989238	5.672
100%	14508	48	696384	732909	5764983	7.866

(2) **With / without EXT2-FS aware**

In this test, we create 14508 files (64KiB each) to occupy the whole SWD, then delete all the files.
And then recreate 14508 files.
These are almost sequential writes, not random writes. However, without FS-Aware, the write amplification is about 2.0. Table 2 shows that with FS-Aware, the Write Amplifcation is 1.0 while 1.932 without it.

Table 2: with / without EXT2-aware

	SMR_Write_	Amplifiled_R&W	Write Amplicfication
FS-unaware	247661	478573	1.932
FS-aware	247182	247182	1.000