An Embedded Storage Framework Abstracting Each Raw Flash Device as An MTD

Wei Wang, Deng Zhou, and Tao Xie,
San Diego State University, San Diego, CA 92182

The 8th ACM International Systems and Storage Conference (SYSTOR 2015, Full Paper), Haifa, Israel, May 26-28, 2015

Introduction

mobile devices such as smart- phones and 3-D digital cameras normally use raw flash memory device directly managed by an embedded flash file system, a dedicated file system designed for storing files on raw flash memory devices. Thus, in addition to provide normal file system functions and interface to up layer system, flash file systems are also designed to directly control raw flash memory devices, they can address the inherent constraints of flash (e.g., wear out issue and the erase-before-write problem). Although existing embedded flash storage systems are effective for traditional mobile applications, they are becoming increasingly inadequate for emerging data- intensive mobile applications due to the following limitations:

(1) *Insufficient storage capacity*: One of the most remarkable trends of mobile computing is that emerging mobile applications are creating huge volume of data.

(2) *Incompetent performance*: current flash file system only supports serial IO access. This architecture significantly hinders I/O performance not only because of serial IO but also due to features of flash memory: asymmetric read & write latency.

(3) *Inadequate level of reliability*: This is mainly because data sampled from mobile and dynamic environments is most likely irreproducible, and thus, data loss is completely unacceptable.

To break the barrier of current single-MTD device architecture, in this paper we propose an MTD-array based embedded flash storage system framework called MA (MTD array)

Proposed Storage Structure

Current Single MTD Structure

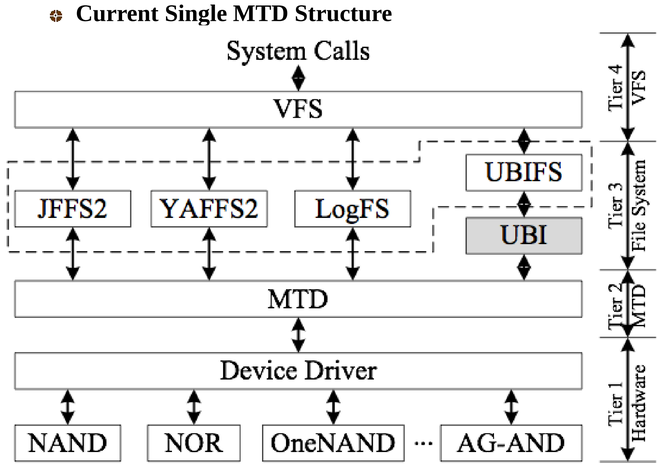

Figure 1. Current MTD subsystem in Linux kernel.

Our Design1: MTD Proxy Middleware

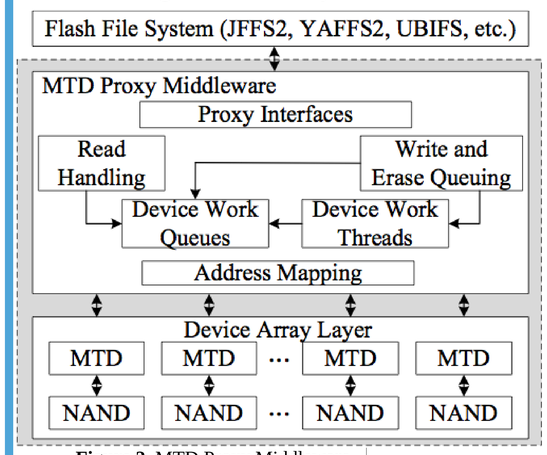

Figure 2. MTD Proxy Middleware.

The MTD proxy middleware has the following components: address mapping, access queuing management, multiple work thread layer, proxy interface module, With the support of the four components, the MTD proxy middleware enables an existing flash file system to communicate with an array of MTD devices without no modifications of existing flash file systems, which is an attractive feature for system design and optimization.

Our design 2: MTD-array

To fully utilize the underlying MTD array, we redesign software stacks as shown in Figure 3.

Compared with the first scheme, it can fully employ the parallelism among multiple MTD devices not only by splitting requests but also by issuing multiple requests concurrently.

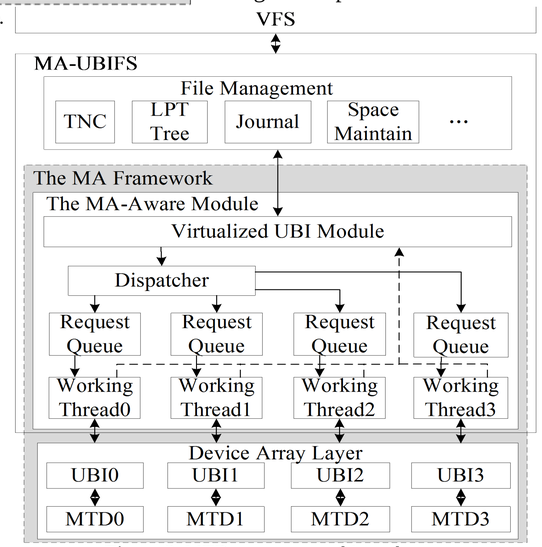

Figure 3. MTD-array storage hierarchy.

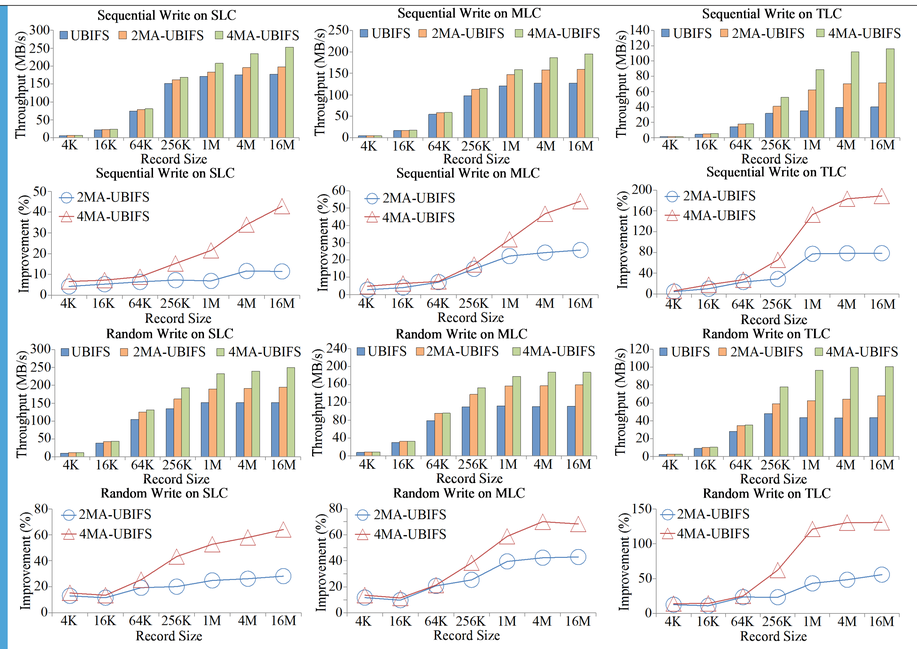

Figure 4. Sequential write and random write throughput on three flash memory types.

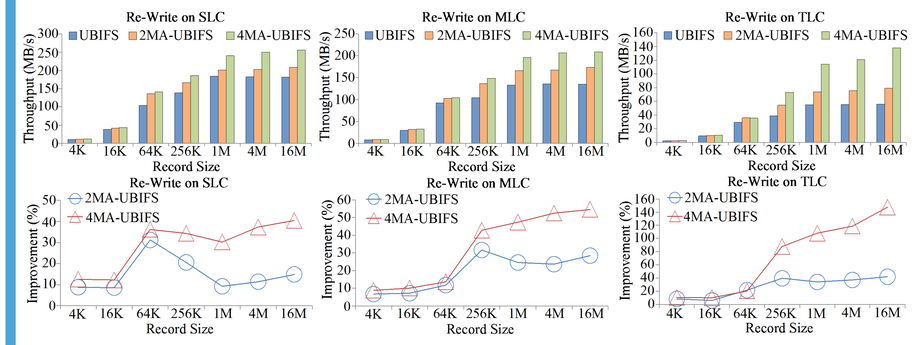

Figure 5. Re-write throughput on three flash memory types.

Experiment Setup

Evaluation environment is set on a 3.1GHz Intel Core i5 machine with 8 GB of RAM. Operating system is Ubuntu 13.04 with 3.8.0 Linux kernel. We also modified NANDSim to support multiple MTD devices emulation.

We change the number of raw flash memory devices used in MA-UBIFS to measure its scalability. Table 2 illustrates the number of flash memory devices and the capacity of each device in every experiment

Table 1: Flash memory major characteristics

Type	SLC	MLC	TLC
Page size (KB)	2	2	4
Read time (μs)	25	25	75
Program time (μs)	200	300	1300
Erase time (ms)	1.5	2	4

Table 2: Flash memory capacity and UBI volume size

	Conf_1	Conf_2	Conf_3
Device num.	1	2	4
MTD capacity (MB)	1024	512	256
UBI Volume (MB)	986	493	246

Conclusions

Emerging data-intensive and mission-critical mobile applications are increasingly demanding a huge storage capacity, superior I/O performance, while existing embedded flash storage systems only support single MTD devices and serial IO access. To cope with this problem, we designs, implements, and evaluates a high-performance embedded flash storage system. Supporting an array of MTD devices is a breakthrough for contemporary embedded flash storage systems. In addition, the positive experimental results demonstrate the feasibility of the proposed MA framework. Lastly, to further enhance data reliability, MA will be extended to incorporate RAID- like data redundancy.

Acknowledgement

This work was supported by the U.S. NSF under grant CNS-1320738.

References

[1] ENGEL,J.,ANDETAL. Logfs-finally a scalable flash file system. In *12th International Linux System Technology Conference*.

[2] HUNTER,A.A brief introduction to the design of ubifs,2008.

[3] LINUX. Nand simulator in linux kernel, http://www.linux- mtd.infradead.org/faq/nand.html.