B. Mitschang et al. (Hrsg.): Datenbanksysteme für Business, Technologie und Web (BTW 2017),
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 605

Effective DBMS space management on native Flash

Sergej Hardock[1] Ilia Petrov[2] Robert Gottstein[1] Alejandro Buchmann[1]

Abstract: In this paper we build on our research in data management on native Flash storage. In particular we demonstrate the advantages of intelligent data placement strategies. To effectively manage physical Flash space and organize the data on it, we utilize novel storage structures such as regions and groups. These are coupled to common DBMS logical structures, thus require no extra overhead for the DBA. The experimental results indicate an improvement of transactional throughput for OLTP benchmarks of up to 60% and decrease in write-amplification of up to 2x, which doubles the longevity of Flash SSD. During the demonstration the audience can experience the advantages of the proposed approach on real Flash hardware.

Keywords: Native Flash interface, Flash management, FTL, storage manager, data placement, region.

1 Introduction

To provide backwards compatibility with spinning drives modern Flash SSDs create a black-box abstraction over Flash memory. This is realized inside the device by the so called Flash Translation Layer (FTL). Thus, FTL masks the native properties of Flash memories (e.g. erase-before-overwrite principle, wear-out of Flash blocks, etc.) and emulates the behavior of traditional HDDs, i.e. supporting reads and writes from immutable device addresses. While this architecture makes the replacement of HDDs seamless, it is responsible for underutilizing the performance potential of Flash SSDs. The major disadvantages of the FTL-based black-box SSDs are: (i) DBMS information about data and run-time statistics cannot be utilized to optimize FTL algorithms; (ii) the DBMS neither has control over the physical data placement, nor (iii) over the internal FTL processes, e.g. wear-leveling (WL), garbage collection (GC) or bad-block management (BBM) [CKZ09]; (iv) a high level of functional redundancy along the I/O path down to storage. All these result in high write-amplification (i.e. ratio between the amount of data written by the DBMS and the actually written on physical storage), intensive wear of Flash memory and often unpredictable and state-dependent performance [CKZ09].

To overcome these disadvantages we recently proposed the NoFTL approach [Ha15], which assumes native Flash as secondary DBMS storage. NoFTL removes all intermediate abstraction layers along the critical I/O path (block device interface, file system and FTL), and enables the DBMS to control the physical Flash storage directly. This is achieved by integrating Flash management functionality (address mapping, GC, WL, BBM) into the subsystems of the DBMS (Figure 1). This integration creates the win-win situation for both the DBMS and the Flash management algorithms.

[1] TU Darmstadt, DVS Group, Hochschulstraße 10, 64289 Darmstadt, \protect\protect\T1\textbraceleftlastname\protect\protect\T1\textbraceright@dvs.tu-darmstadt.de
[2] Reutlingen University, DB Lab, Alteburgstraße 150, 72762 Reutlingen, ilia.petrov@reutlingen-university.de

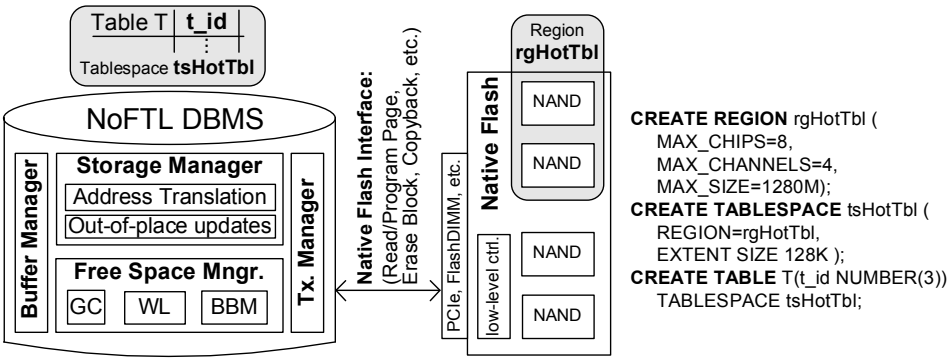

Fig. 1: General NoFTL Architecture including Regions.

In the present paper we revisit traditional methods for physical space management. We introduce two novel storage structures *regions* and *groups*, which allow performing intelligent data placement strategies and reduceing the write-amplification; improve on transaction throughput, and longevity of Flash storage. These structures are coupled to standard logical DBMS structures (tablespaces, extents), hence no extra DBA overhead is incurred.

2 Data organization on native Flash storage

Flash SSDs comprize multiple data channels and Flash chips (dies), which primary define the minimum level of I/O parallelism supported by the device. The general approach in modern SSDs is to spread all the data evenly across the whole physical address space regardless of its properties, since such information is unavailable to traditional black-box SSDs. Although this strategy typically provides good load-balancing and allows for simple WL algorithms, we argue that especially for write-intensive workloads like OLTP it results in (i) enormous write-amplification caused by the GC, (ii) faster wear-out of Flash memory, and (iii) insufficient utilization of available on-device I/O parallelism. To solve these issues we apply an intelligent placement strategies under the NoFTL architecture. NoFTL makes the details about the physical Flash organization visible to the DBMS. This information together with the rich DBMS statistics and metadata provides the basis for efficient data placement. The latter is possible due to the use of native Flash interface and the DBMS control over the physical placement on Flash. To organize and manage data on Flash we introduce a novel storage structure - *regions*. A region comprises multiple data channels and a set of Flash chips. The number of chips in each region as well as the structure of their set is dynamic and can change over time depending on various factors. One or more DB objects with similar access properties can be physically placed in a region; this holds for complete objects or partitions of them. Objects with different properties are placed in different physically separate regions to account and optimize for the specific access characteristics.

The utilization of regions for physical data organization allows for applying hot/cold data separation techniques, which significantly reduce the GC overhead. This in turn, results

in a decrease of the write-amplification and improvement of Flash longevity. Moreover, the proper distribution of data channels and Flash chips among regions, depending on the properties of database objects assigned to them, reduces the contention for physical resources and improves the I/O throughput. Last but not least, the notion of regions introduces only negligible administration overhead for the DBA, since the new structure is coupled to an existing DBMS logical structure - *tablespace*. Consider the example in Figure 1: a region of a certain size *rgHotTbl* is defined over 8 chips. A tablespace *tsHotTbl* is defined on top of *rgHotTbl*, where a newly created table *T* is placed.

In addition to regions we define the notion of *groups*, which are coupled to DBMS *extents*. Groups allow for further hot/cold separation improvement by preventing data with different update frequencies from being mixed within a single Flash block (erase unit). The pages of objects belonging to different groups are kept in separate Flash blocks, thus ensuring higher "temperature homogeneity" within individual blocks. This reduces the write-amplification caused by the GC, by lowering the number of page migrations performed by erasing a victim block [SA13]. Interestingly, the utilization of groups does not harm the even wear-out of Flash blocks within a region, since there is no fixed assignemnt of available blocks to groups. Hence, blocks belonging to different groups are evenly distributed within a certain region, causing thereby its even wear-out in general. Regions and groups exemplify how the utilization of DBMS run-time information on one side, and the knowledge about the Flash physical architecture on the other side, can be used to perform intelligent data placement strategies and achieve higher I/O and transaction throughput, while simultaneously extending the lifetime of Flash SSD. Our experimental results indicate up to 60% improvement in transaction throughput for the TPC-C benchmark and 25% for TPC-B, respectively, with a simultaneous decrease of the performed erase operations by up to 2x, which has a direct impact on the Flash longevity.

3 Demonstration

During the demonstration we introduce the audience to basics of the proposed approach and let them experience it interactively either on real hardware or on a Flash emulator. The demonstration system consists of Flash storage - OpenSSD Flash research board[3] connected to a host PC running Shore-MT[4] (Figure 2). Using an intuitive GUI (Figure 3) the audience can configure a sequence of tests and experience live the performance advantages of the data placement strategies based on utilization of regions and groups. The GUI allows to create multi-region data placement configurations, manage Shore-MT and visually compare the experimental results. The proposed demonstration scenarios are as follows.

Demo-Scenario 1 – Baseline. The audience picks one of the three available OLTP benchmarks (TPC-B, TPC-C or TATP), selects the desired scaling factor (limited by 64GB of Flash storage), the duration of the test and the kind of Flash storage device (Jasmine OpenSSD or Flash Emulator). Shore-MT executes the benchmark using the traditional data placement, i.e. without utilization of regions and groups. During the benchmark run,

[3] http://www.openssd-project.org
[4] https://sites.google.com/site/shoremt/

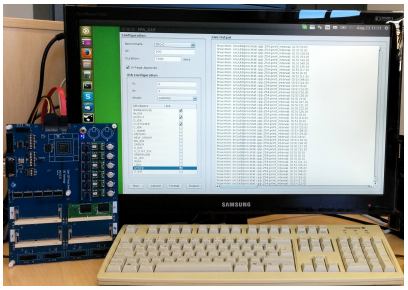

Fig. 2: Demonstration testbed.

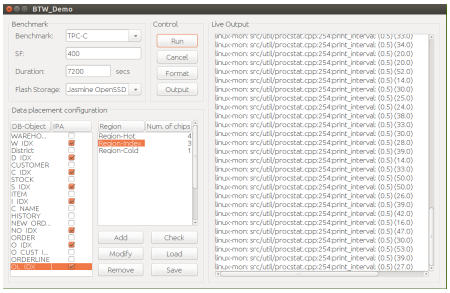

Fig. 3: Demonstration GUI.

the current transactional throughput is visualized. Upon completion, detailed statistics of performed I/Os are shown.

Demo-Scenario 2 – Hot/Cold data separation. In this scenario the audience examines the effects of the data placement strategies based on the hot/cold data separation, which is realized by means of regions and groups. Using the detailed Shore-MT I/O statistics from the *baseline* test the user creates with the help of the demonstration GUI a multi-region data placement configuration. Through the clustering of database objects with similar properties into regions, the overhead of the GC can be significantly reduced. Once the configuration is set and the Flash SSD is completely formatted (low-level) the benchmark is run with the same scaling factor and for the same duration as in the baseline test. The audience can compare the output results of both approaches (throughput, I/O statistics).

Demo-Scenario 3 – Parallelism. This scenario is similar to the previous one, however, the emphasis is placed on controlling the parallelism provided by the Flash storage device. Due to the lack of the SATA NCQ support on the OpenSSD board its level of I/O parallelism is very limited. Thus, the tests in this scenario are performed on the real-time Flash emulator developed in our lab and successfully validated against the real hardware [Ha15]. The Flash chips and data channels of the emulated Flash storage device are divided into regions so that database objects with high demand on I/O concurrency are assigned to regions with more chips and data channels. By doing so the contention for physical resources can be minimized and the available Flash parallelism efficiently utilized.

Acknowledgements. This paper was supported by the German BMBF “Software Campus” (01IS12054) and the German Research Foundation (DFG) project “Flashy-DB”.

References

[CKZ09] Chen, F.; Koufaty, D. A.; Zhang, X.: Understanding intrinsic characteristics and system implications of flash memory based SSDs. In: Proc. SIGMETRICS’09. 2009.

[Ha15] Hardock, S.; Petrov, I.; Gottstein, R.; Buchmann, A.: NoFTL for Real: Databases on Real Native Flash Storage. In: Proc. EDBT’15. 2015.

[SA13] Stoica, R.; Ailamaki, A.: Improving Flash Write Performance by Using Update Frequency. In: Proc. VLDB’13. 2013.