View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by CiteSeerX

Poster Paper

Revisiting DBMS Space Management for Native Flash

Sergey Hardock
Databases and Distributed Systems Group
TU-Darmstadt, Germany
hardock@dvs.tu-darmstadt.de

Ilia Petrov
Data Management Lab
Reutlingen University, Germany
ilia.petrov@reutlingen-university.de

Robert Gottstein
Databases and Distributed Systems Group
TU-Darmstadt, Germany
gottstein@dvs.tu-darmstadt.de

Alejandro Buchmann
Databases and Distributed Systems Group
TU-Darmstadt, Germany
buchmann@dvs.tu-darmstadt.de

ABSTRACT

In this paper we present our work in progress on revisiting traditional DBMS mechanisms to manage space on native Flash and how it is administered by the DBA. Our observations and initial results show that: the standard logical database structures can be used for physical organization of data on native Flash; at the same time higher DBMS performance is achieved without incurring extra DBA overhead. Initial experimental evaluation indicates a 20% increase in transactional throughput under TPC-C, by performing intelligent data placement on Flash, less erase operations and thus better Flash longevity.

1. INTRODUCTION

We argue that the design of the storage architecture is not well suited for new kinds of memory in terms of both software and hardware. Flash memory has significant performance potential, which is underutilized due to the present architecture of Flash SSDs and the way they are used by the DBMS. To provide backwards compatibility with magnetic drives, modern Flash SSDs implement legacy block-device interfaces, supporting *reading* and *writing* at *Flash page granularity* from *immutable device addresses*. As a result a black-box abstraction over the Flash memory is created inside the device by the so called Flash Translation Layer (FTL). Although, this has facilitated the widespread use of SSDs, it results in: (i) significant overhead primarily due to limited on-device resources available to the FTL; (ii) unpredictable performance caused by the background FTL processes (wear-levelling (WL) and garbage collection (GC)) [1]; (iii) inability to optimize the DBMS I/O behavior for new kinds of storage due to an additional level of indirection.

To overcome these disadvantages we recently proposed the NoFTL approach [2], 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 address space of Flash storage directly (see Figure 1). Under NoFTL the DBMS is not confronted with the intricate low-level NAND control. The Flash device is still assumed to have a thin hardware management layer (a low-level controller).

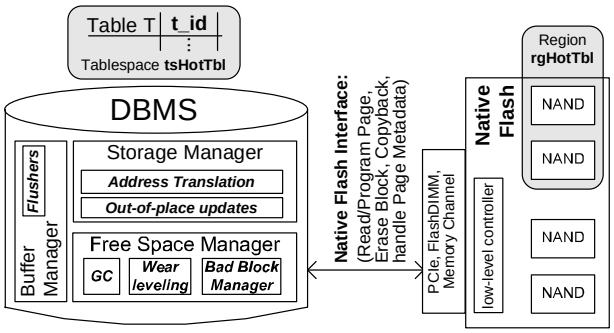

Figure 1: General NoFTL Architecture including Regions.

The major advantages of NoFTL over the traditional FTL-based Flash SSD are the following: (i) usage of the more powerful computational and memory resources of the host system for complex Flash maintenance tasks; (ii) utilization of the DBMS run-time information and knowledge about the stored data and I/O for optimization of GC, WL and the address mapping scheme; (iii) better utilization of available Flash parallelism through intelligent data placement; (iv) direct control over the out-of-place updates, which allows implementing short atomic writes without additional overhead; (v) elimination of redundant functionality along the I/O path.

In the present paper we revisit traditional methods for physical space management on Flash under NoFTL. *The central questions are: how can native Flash comprising a loose set of Flash chips be organized and utilized by the DBMS; do we need new logical storage structures; will they overcomplicate the job of the DBA?*

2. LOGICAL STORAGE STRUCTURES AND DATA PLACEMENT ON FLASH

We introduce the concept of **NoFTL regions** as a new physical storage structure, to simplify the organization and management of native Flash storage. A *region* comprises multiple Flash chips or dies, over which the data is evenly distributed. The number of

©2016, Copyright is with the authors. Published in Proc. 19th International Conference on Extending Database Technology (EDBT), March 15-18, 2016 - Bordeaux, France: ISBN 978-3-89318-070-7, on OpenProceedings.org. Distribution of this paper is permitted under the terms of the Creative Commons license CC-by-nc-nd 4.0

dies in each region, as well as the structure of their set is dynamic and can change over time depending on different factors: size of objects, required level of I/O parallelism and global wear-levelling.

```
CREATE REGION rgHotTbl (
     MAX_CHIPS=8, MAX_CHANNELS=4, MAX_SIZE=1280M);
CREATE TABLESPACE tsHotTbl (
     REGION=rgHotTbl, EXTENT SIZE 128K );
CREATE TABLE T(t_id NUMBER(3))TABLESPACE tsHotTbl;
```

One or more database 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. This gives us two distinct advantages discussed in detail below: (a) coupling of regions and existing logical structures to simplify database administration, and (b) ability to perform intelligent data placement to increase performance and improve Flash longevity.

Logical Storage Structures and NoFTL Regions.

Existing logical DBMS storage structures can be defined on top of NoFTL regions. Consider the above example: 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. The DBA can continue using established logical storage structures such as *tablespaces* or *extents*, which can be effectively coupled to *regions*. Hence, no new logical structures are needed to manage, organize native Flash storage. *The administration of native Flash does not confront the DBA with additional complexity.*

Data Placement and NoFTL Regions.

It is well known that Flash memory can perform random access almost as fast as sequential (which is not always true for SSDs). Thus, keeping logically adjacent blocks, physically distributed has negligible performance implications. Furthermore, the distribution over available Flash data channels, dies or planes allows for better I/O parallelism than storing those blocks in sequential order physically on Flash. On the other hand, the database knowledge about the data, about its properties and access patterns can be used to perform smarter data placement and optimize important Flash maintenance algorithms, such as WL, GC and address mapping scheme.

For instance, it is proven that the overhead of garbage collection, which is the major factor for unpredictable performance on SSDs, is highly dependent on the ability to separate between hot and cold data [4, 3]. The limited on-device SSD resources rarely allow for maintaining comprehensive statistics about access patterns and access frequencies over the whole logical address space. At the same time, the DBMS maintains such and other statistics and metadata for each particular database object. Since under NoFTL the DBMS has full control over the physical Flash address space and can perform direct data placement, it becomes easy to utilize the DBMS knowledge. Regions, therefore, allow the DBA and DBMS to control physical data placement on the device in order to optimize Flash management. Intelligent data placement using regions is in the general case an optimal trade off between the provided I/O-parallelism and the overhead of GC.

3. PRELIMINARY EVALUATION

We implemented and integrated *regions* in the NoFTL architecture [2] under Shore-MT. Our initial results indicate that the concept of intelligent data placement on Flash has big potential. For instance, under TPC-C we could achieve about 20% increase in transactional throughput (Figure 3) by applying multi-region data placement configuration (Figure 2). In this configuration we have divided database objects of TPC-C based on their I/O properties into 6 regions. Further we have distributed 64 dies of Flash SSD over those regions based on sizes of objects and their I/O rate (required level of I/O parallelism). In addition to performing 20% more transactions than traditional data placement and 20% more READ and WRITE I/Os, the GC performs almost 20% less COPYBACKs and 4.3% less ERASEs. This reduction in write-amplification of multi-region configurations leads to lower I/O latencies and consequently to lower transaction response times. The second effect of decreased write-amplification of multi-region data placement configurations is the better longevity of the Flash devices.

Tablespace/ Region	DB-Objects	Num. of Flash dies
0	DBMS-metadata; HISTORY NEW_ORDER; ORDER	2
1	ORDERLINE	11
2	CUSTOMER	10
3	OL_IDX; STOCK	29
4	C_IDX; I_IDX; S_IDX; W_IDX C_NAME_IDX; ITEM; D_IDX	6
5	WAREHOUSE; DISTRICT NO_IDX; O_IDX; O_CUST_IDX	6

Figure 2: Multi-region data placement configuration for TPC-C

		Traditional data placement	Data placement using Regions
	TPS	595.42	720.43
Response Time	READ 4KB (µs)	531.00	318.63
	WRITE 4KB (µs)	904.00	564.83
	NewOrder TRX (ms)	61.43	58.45
	Payment TRX (ms)	8.88	6.99
	StockLevel TRX (ms)	437.30	293.97
Number of ...	Transactions	359,725	433,192
	Host READ I/Os (4KB)	19,017,255	23,329,310
	Host WRITE I/Os (4KB)	2,740,236	3,259,162
	GC COPYBACKs	4,326,612	3,496,984
	GC ERASEs	110,410	105,564

Figure 3: Performance comparison of traditional and multi-region data placement configuration.

4. CONCLUSIONS

The black box architecture of modern Flash SSDs does not allow to utilize the rich DBMS knowledge about stored data and I/O for optimization of complex Flash maintenance functionality such as garbage collection, wear-levelling and address mapping scheme. In pursuit of a solution, we extend our NoFTL approach with the notion of *regions* for allowing the DBMS to control the physical placement based on the properties of database objects. *NoFTL Regions* can be easily mapped to existing DBMS structures such as tablespaces. Our initial results indicate that intelligent data placement can significantly improve the performance of the DBMS as well as the longevity of Flash devices.

Acknowledgements

This paper was supported by the German BMBF "Software Campus" (01IS12054), the German Research Foundation (DFG) within GRK 1343 "Topology of Technology" and DFG "Flashy-DB" project.

5. REFERENCES

[1] F. Chen, D. A. Koufaty, and X. Zhang. Understanding intrinsic characteristics and system implications of flash memory based solid state drives. In *Proc. SIGMETRICS'09*, 2009.

[2] S. Hardock, I. Petrov, R. Gottstein, and A. Buchmann. Noftl for real: Databases on real native flash storage. In *Proc. EDBT'15*, 2015.

[3] J. Lee and J.-S. Kim. An empirical study of hot/cold data separation policies in solid state drives. In *Proc. SYSTOR '13*.

[4] R. Stoica and A. Ailamaki. Improving flash write performance by using update frequency. In *Proc. VLDB'13*, 2013.