IPA-IDX: In-Place Appends for B-Tree Indices

Sergey Hardock, Andreas Koch
Technische Universität Darmstadt, Germany

ACM Reference Format:

1 INTRODUCTION

Indexes are some of the hottest and most write-intensive database objects. Modification and maintenance operations in a B-Tree may spread several nodes yielding high write-amplification and degrading I/O performance on modern storage technologies. Techniques such as In-Place Appends (IPA) [5] are designed to handle such small random writes graciously. Upon an in-place update to a DB-page, IPA [5] tracks the changed bytes between the original and updated record, computes their offsets producing value/offset pairs and appends these as delta-records to that page. However, IPA’s offset-value delta-record format is unsuitable for B-Tree indices. Assume a unique B-Tree index Idx1 on a numeric field Attr1 of database table Tbl1 (Figure 1). Transaction Tx1 modifies a single record of Tbl1 on page 123, changing the value of Attr1 from 10 to 100, modifying 4 bytes. (1) The index entry with key 10 is deleted from leaf page 456. Hence, either the deleted vector is updated, or the slots are right-shifted by one. (2) A new index entry with key 100 is inserted into leaf page 789. The insertion requires modification of both, the slot directory and the page body. Thus, a one-field modification of a single record in Tbl1 causes three database pages to be updated (123, 456 and 789). IPA can only handle the small 4 byte update to page 123. Whereas, the updates to pages 456 and 789 result in numerous multi-byte changes, yielding many offset-value pairs in excess of 500 bytes, making it impractical for IPA.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org. Conference’17, July 2017, Washington, DC, USA © 2019 Association for Computing Machinery. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00 https://doi.org/10.1145/nnnnnn.mnnnnn

2 IPA-IDX

The basic idea of IPA-IDX is straightforward. Whenever a transaction triggers an index modification, IPA-IDX tracks the changes to the corresponding index pages in the buffer. IPA-IDX appends a copy of the respective physiological log records to those pages if: (i) the page can be overwritten in-place, i.e., number of already performed in-place overwrites is less than N); (ii) corresponding log record for this change fits into the remaining space in the delta-record area. This repeats for the changes pending on those page. Subsequently, the buffer manager sets an IPA flag on those pages. If they get evicted the storage manager uses a special write_delta command [5], physically appends the delta-records in-place on flash. Conversely, if one of the above conditions is not satisfied the page is flagged to be written out-of-place to a clean Flash address, and no further IPA-IDX tracking is performed until the page is flushed. In both cases modifications to index page body are performed as usual. Note that IPA-IDX has no implications on recovery as regular WAL and recovery protocols are in place. IPA-IDX neither triggers additional I/Os, nor does it influence the eviction strategy.

Whenever an index page is fetched, the storage manager checks if the page contains delta records. If so, it performs logical operations of the log records stored in those delta records. Applying delta records in IPA-IDX is similar to executing the REDO phase of recovery for a particular page.
3 RELATED WORK
IPA-IDX builds on the concept of unsorted B*-Tree (leaf) nodes, which has already been explored in numerous works [2, 3, 9–11]. [10] proposes controlled sort imbalance based on the read-to-write ratio. The concept of transforming in-place index updates into physiological log records has been pioneered by IPL for B*-Treen[8] and refined by d-IPL B*-Tree[7]. [6–8] append the log records to an extra log-page within the respective Flash block. These approaches are designed for SLC Flash, while IPA-IDX can handle MLC and 3D NAND. IPL [6], d-IPL B*-Tree[7] also show significant read-amplification compared to IPA-IDX as the log-page(s) must be read for each index node/page. IPA-IDX places the delta records on the very same node, using ISPP-based write techniques, keeping the write- and read-amplification low.

4 EVALUATION
IPA-IDX is implemented in NoFTL under Shore-MT and evaluated on real hardware (OpenSSD JASMINE [1]). IPA-IDX can be applied selectively, i.e. only to specific regions [4], and within those regions only to selected DB-objects.

Figure 3 presents the performance results of IPA-IDX under TPC-C, where IPA-IDX is enabled for five B-Tree indexes: NO_IDX(no_w_id, no_d_id, no_o_id), O_IDX(o_w_id, o_d_id, o_id), O_CUST_IDX(o_w_id, o_d_id, o_c_id, o_id), OL_IDX(ol_w_id, ol_d_id, ol_o_id, ol_number). As these are only appended to, we add an additional S_QUANTITY_IDX (s_w_id, s_i_id, s_quantity) index to have a B-Tree with insertions and deletions. All these indexes together occupy about 30% of database space.

We apply a [2x270] scheme (Figure 2), that allows performing 40% of all database writes using IPA-IDX, while for five indexes the fraction of delta_writes is about 65%. IPA-IDX improves the transactional throughput by 9%. Noticeably larger is the benefit for the GC overhead. The avg. amount of page migrations per one 4KB host write was reduced by 40%, while the number of erases by 44%.

In a second experiment we investigate the combined effects of basic IPA and IPA-IDX (Figure 4). Firstly, we apply basic IPA with [NxM]=[2x16] (Figure 2) scheme to the Account table in TPC-B. The transactional throughput improves by 12%, and garbage collection overhead decreases by 37%. In this experiment there is an additional index H_IDX(h_a_id, h_date) on the History table. Since every transaction inserts a new entry in the History table, this index becomes write-intensive, and accounts for approx. 45% of the database writes (another 54% of write I/Os falls on the Account table).

Secondly, on top of basic IPA, we enable IPA-IDX for the H_IDX index using [NxM]=[2x270] (Figure 2). The total fraction of IPA write I/Os (delta_writes) increases to 64%. The GC overhead decreases by a further 30% (66%-68% in total compared to baseline without IPA). The tx. throughput improves by 12% with basic IPA, and by 28% with IPA and IPA-IDX. The space overhead of IPA and IPA-IDX is 2%.

5 CONCLUSION
We introduce IPA-IDX – an approach to handle index modifications modern storage technologies (NVM, Flash) as physical in-place appendings, using simplified physiological log records. IPA-IDX provides similar performance and longevity advantages for indexes as basic IPA [5] does for tables. The selective application of IPA-IDX and basic IPA to certain regions [4] and objects, lowers the GC overhead by over 60%, while keeping the total space overhead to 2%. The combined effect of IPA and IPA-IDX increases performance by 28%.
REFERENCES


