Selective In-Place Appends for Real: Reducing Erases on Wear-prone DBMS Storage

Sergey Hardock*, Ilia Petrov†, Robert Gottstein*, Alejandro Buchmann*
*Databases and Distributed Systems Group, TU-Darmstadt, Germany
Email: {hardock; gottstein; buchmann}@dvs.tu-darmstadt.de
†Data Management Lab, Reutlingen University, Germany
Email: ilia.petrov@reutlingen-university.de

***Abstract*—In the present paper we demonstrate the novel technique to apply the recently proposed approach of In-Place Appends – overwrites on Flash without a prior erase operation. IPA can be applied selectively: only to DB-objects that have frequent and relatively small updates. To do so we couple IPA to the concept of NoFTL regions, allowing the DBA to place update-intensive DB-objects into special IPA-enabled regions. The decision about region configuration can be (semi-)automated by an advisor analyzing DB-log files in the background.**

We showcase a Shore-MT based prototype of the above approach, operating on real Flash hardware. During the demonstration we allow the users to interact with the system and gain hands-on experience under different demonstration scenarios.

Video: https://youtu.be/GMKCoZmSZ4Y

I. Introduction

In-Place Appends (IPA) [1] is a recently proposed approach handling two common types of write-amplification (WA). *DBMS WA* reflects flushing of the whole 4-32KB DB-page even if only few bytes on it are changed, while the *SSD on-device WA* is caused by page migrations due to the internal garbage collection. The basic idea of the approach is to transform small in-place updates performed by DBMS transactions into delta-records upon page eviction. Furthermore, those delta-records are appended to a reserved area on the *very same physical Flash pages* along with the original content. We utilize the commonly ignored fact that under certain conditions physical Flash pages can be updated in-place without a prior erase operation. Thus, by relaxing the *erase-before-overwrite principle* of Flash memory we can significantly reduce the number of page invalidations and out-of-place updates. Moreover, this results in reduction of GC overhead (page migrations and erase operations) and consequently in decrease of I/O response times. In addition, the *DBMS WA* is reduced by a newly defined command *write_delta*, which allows the DBMS to write out only the delta-records instead of whole pages.

The approach presented here is implemented and evaluated as part of NoFTL [2], [3]. The main concept behind NoFTL is to integrate Flash management into the DBMS (Figure 1). The access to rich DBMS run-time information and statistics allows for significant optimization of the Flash management functionality. Moreover, NoFTL allows native DBMS subsystems (e.g. buffer and storage management) to benefit from controlled data placement and knowledge of the internal Flash organization. In [3] we introduce the notion of NoFTL *regions*, which we utilze currently to apply IPA selectively to specific DB objects (or sets thereof).

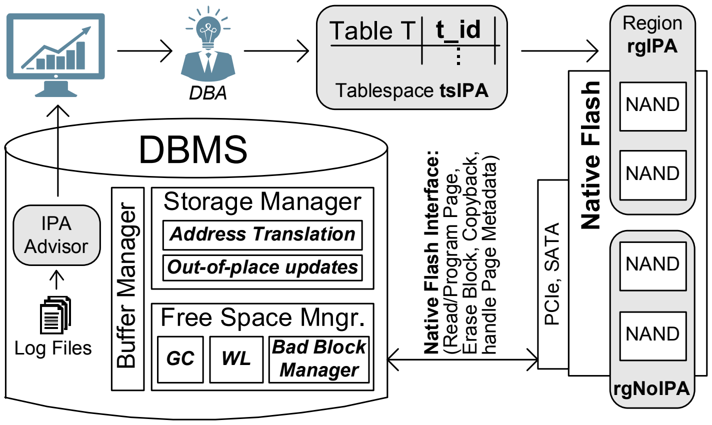


Fig. 1. NoFTL Architecture with Regions supporting IPA.

For instance, write-intensive tables or indexes dominated by small updates can be placed in a region that uses IPA (e.g. region *rgIPA* in the example below). Read-only objects or objects dominated by large updates can be placed in yet another region, which does not utilize IPA and relies only on out-of-place updates. The presented solution incurs negligible or no extra DBA complexity, due to the integration of IPA to regions and coupling to existing logical database storage structures (e.g. tablespaces). Using the *IPA advisor* proposed here the decision regarding the proper IPA-region configuration is taken in a highly automated manner.

```
CREATE REGION rgIPA(MAX_SIZE=32G,MAX_CHIPS=4,
     MAX_CHANNELS=2, IPA_MODE = pSLC);
CREATE TABLESPACE tsIPA (
     REGION=rgIPA, EXTENT SIZE 128K );

CREATE TABLE T (t_id INT) TABLESPACE tsIPA;
CREATE INDEX I ON T(t_id) TABLESPACE tsIPA;
```

II. Demonstration

During the demonstration we introduce the audience to basics of the proposed approach and let them explore it interactively on real hardware. The demonstration system consists of the OpenSSD Flash research platform [4] (Figure 2) connected to a host PC running the Shore-MT storage engine[1]. Using an intuitive GUI (Figure 3) the audience can configure a sequence of tests and experience live the performance advantages of the

[1]https://sites.google.com/site/shoremt/

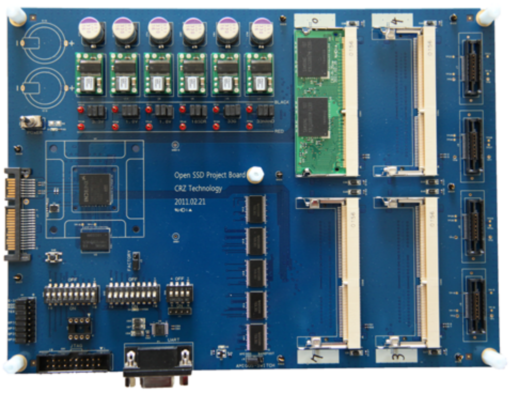
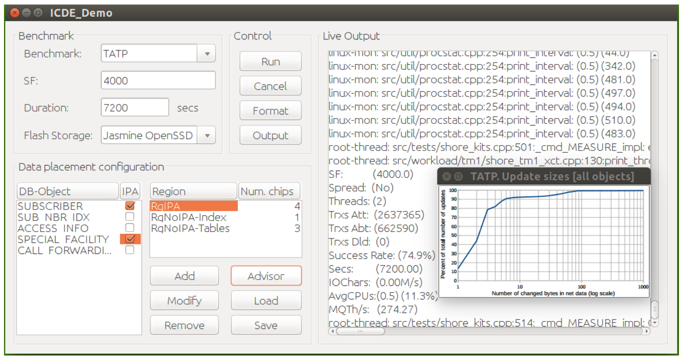


Fig. 2. OpenSSD Flash research board.



ICDE_Demo
Benchmark
Control
Run
Cancel
Format
Output
Live Output
Data placement configuration
Add
Advisor
Modify
Load
Remove
Save
TATP. Update sizes [all objects]
Percent of total number of updates
Number of changed bytes in net data (log scale)

Benchmark: TATP
SF: 4000
Duration: 7200 secs
Flash Storage: Jasmine OpenSSD

DB-Object	IPA
SUBSCRIBER	✓
SUB NBR IDX	☐
ACCESS INFO	☐
SPECIAL FACILITY	✓
CALL FORWARDI...	☐

Region	Num. chips
RgIPA	4
RgNoIPA-Index	1
RgNoIPA-Tables	3

```
linux-mon: src/util/procstat.cpp:254:print_interval: (0.5) (44.0)
linux-mon: src/util/procstat.cpp:254:print_interval: (0.5) (342.0)
linux-mon: src/util/procstat.cpp:254:print_interval: (0.5) (481.0)
linux-mon: src/util/procstat.cpp:254:print_interval: (0.5) (497.0)
linux-mon: src/util/procstat.cpp:254:print_interval: (0.5) (494.0)
linux-mon: src/util/procstat.cpp:254:print_interval: (0.5) (510.0)
linux-mon: src/util/procstat.cpp:254:print_interval: (0.5) (483.0)
root-thread: src/tests/shore_kits.cpp:501:_cmd_MEASURE_impl:
root-thread: src/workload/tm1/shore_tm1_xct.cpp:130:print_thr
SF:          (4000.0)
Spread:  (No)
Threads: (2)
Trxs Att:  (2637365)
Trxs Abt: (662590)
Trxs Dld:  (0)
Success Rate: (74.9%)
Secs:        (7200.00)
IOChars:  (0.00M/s)
AvgCPUs:(0.5) (11.3%)
MQTh/s:  (274.27)
root-thread: src/tests/shore_kits.cpp:514:_cmd_MEASURE_impl:
```

Fig. 3. GUI for the evaluation of selective IPA.

selective use of IPA based on the utilization of NoFTL regions. The proposed demonstration scenarios are as follows.

a) 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) and the duration of the test. The DBMS executes the benchmark using the traditional approach as a baseline, i.e. without IPA and without utilization of regions. During the run of the benchmark the user can observe the current transactional throughput. After the benchmark is finished - the detailed statistics of performed I/Os are shown.

b) Demo-Scenario 2 – IPA log-analyzer: In this scenario we the audience interacts with the IPA log-analyzer. The log-analyzer is designed to assist the DBA in taking the right decision regarding the application of IPA. By analyzing the DBMS log files (e.g. those resulting from from the baseline scenario) it produces comprehensive statistics about the distribution of updates and their sizes for the duration of the whole workload (see example on Figure 3), as well as for each particular DB-object. Moreover, the IPA log-analyzer allows to examine the skew characteristics of the workload. Based on those statistics the DBA can easily decide which database objects would benefit from IPA and what IPA-region configuration would be the most appropriate for the current workload.

c) Demo-Scenario 3 – IPA without Regions: In this scenario the audience examines the application of IPA to all database objects (global application of IPA). After the main parameters of IPA have been selected (based on the statistics from the IPA log-analyzer), and the Flash SSD is completely formatted (low-level formatting) the benchmark is run with the same scaling factor and for the same duration as in the baseline test. Although under this scenario the reduction of the garbage collector overhead is the highest, in the average case it doubles the space overhead compared to the selective application of IPA. Thus, DB-sizes increase in those experiments by up to 10% as compared to the baseline. This is due to reserving space for *delta-record area* in each db-page. Although the global application of in-page appends is rather impractical, together with baseline scenario it represents another extreme, with respect to the decision about which regions (i.e. db-objects) IPA should be applied to.

d) Demo-Scenario 4 – Selective IPA using Regions: Using the GUI and the results from the three preceding scenarios the audience creates a multi-region data placement configuration and configures, which regions should support IPA (and in which mode), and which not. This decision is practically a trade-off between the reduction of garbage collection overhead, the longevity and the increase of DB-size (up to 10% max.). The audience can compare the output results of three approaches (throughput, I/O statistics).

TABLE I. TATP: TRADITIONAL APPROACH VS. IPA WITH REGIONS.

	0x0 Absolute	**IPA [2*3] with regions** *Absolute*	**IPA [2*3] with regions** *Relative [%]*
Out-of-Place Writes vs. IPA	100/0	55/45	
Host Reads (16KB)	5 576 036	6 185 809	+11
Host Writes (16KB)	487 257	540 311	+11
GC Page Migrations	445 348	234 356	-47
GC Erases	5 931	2 803	-53
GC Pg. Migrations per Host Write	0.9140	0.4337	-53
GC Erases per Host Write	0.0122	0.0052	-57
Transactional Throughput	277	306	+11

For instance, Table I shows the comparative results from a TATP benchmark run for two hours on the OpenSSD board (during the demonstration the durations of 5 or 10 minutes are sufficient for a comparison). The experiments were performed without In-Place Appends (column “0x0”) and with IPA using regions. The latter case uses the data placement configuration with two regions: one for *Subscriber* and *Special_Facility* tables with enabled IPA, and another region for the all remaining tables without IPA support. The configuration with IPA and regions outperforms the baseline approach by executing 57% less erases and 53% less GC page migrations per host write. This reduction of GC overhead has two major advantages: (i) the increase of the transactional throughput (11%), and (ii) doubling the Flash SSD lifetime.

REFERENCES

[1] S. Hardock, I. Petrov, R. Gottstein, and A. Buchmann, “From in-place updates to in-place appends: Revisiting out-of-place updates on flash,” in *accepted at SIGMOD 2017*.

[2] ——, “Noftl: Database systems on ftl-less flash storage,” in *Proc. VLDB’13*.

[3] ——, “Revisiting dbms space management for native flash,” in *Proc. EDBT*, 2016.

[4] “The openssd project,” http://www.openssd-project.org/wiki/The_OpenSSD_Project, 2014.