Beyond Block I/O: Rethinking Traditional Storage Primitives

HPCA'11

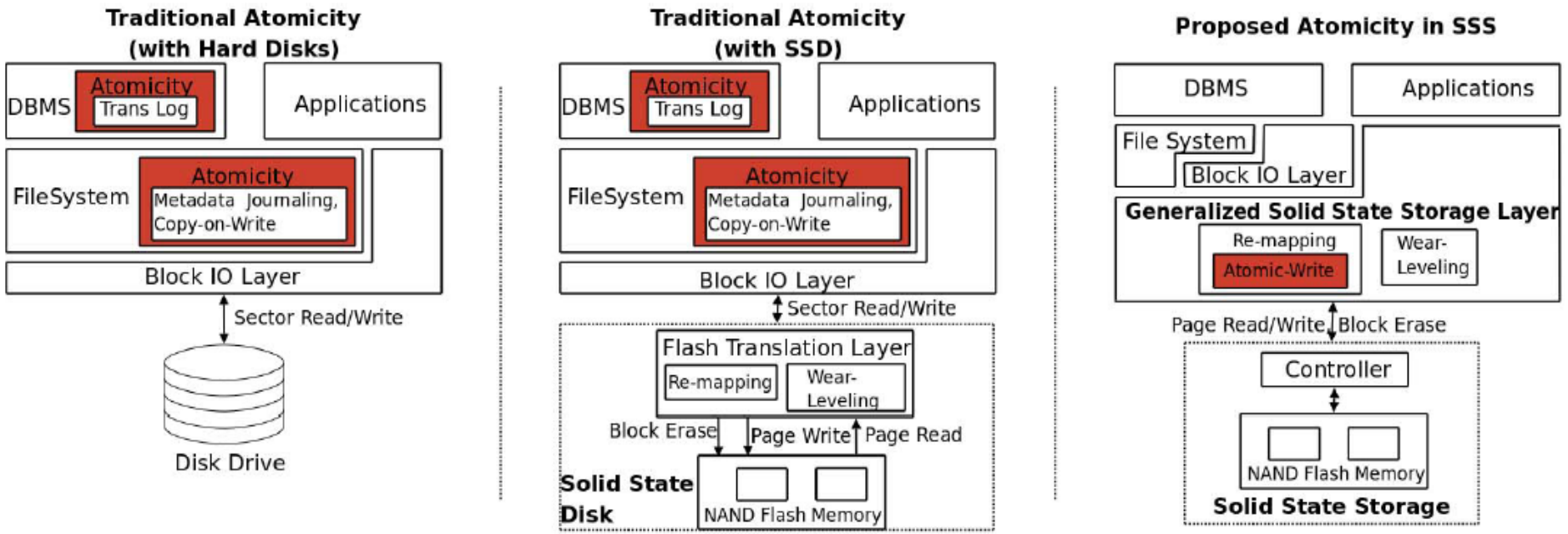
Background

- For decades, ***the interfaces for accessing persistent storage*** have remained unchanged
  - ***Read, write, seek*** and so on are fundamental operations
  - Designed ***for hard disks***
- ***Non-volatile memory (flash SSD)*** becomes the mainstream storage device in various domains
  - ***Uses same I/O interfaces like read/write***
  - ***New additional layer*** (FTL) is implemented in SSD
- ***Additional I/O interfaces*** (except for *read* and *write*) can be defined that can leverage "the FTL of SSD" to provide "new and interesting storage semantics"
  - Can we define ***any other storage primitive*** except for read/write?

Contributions

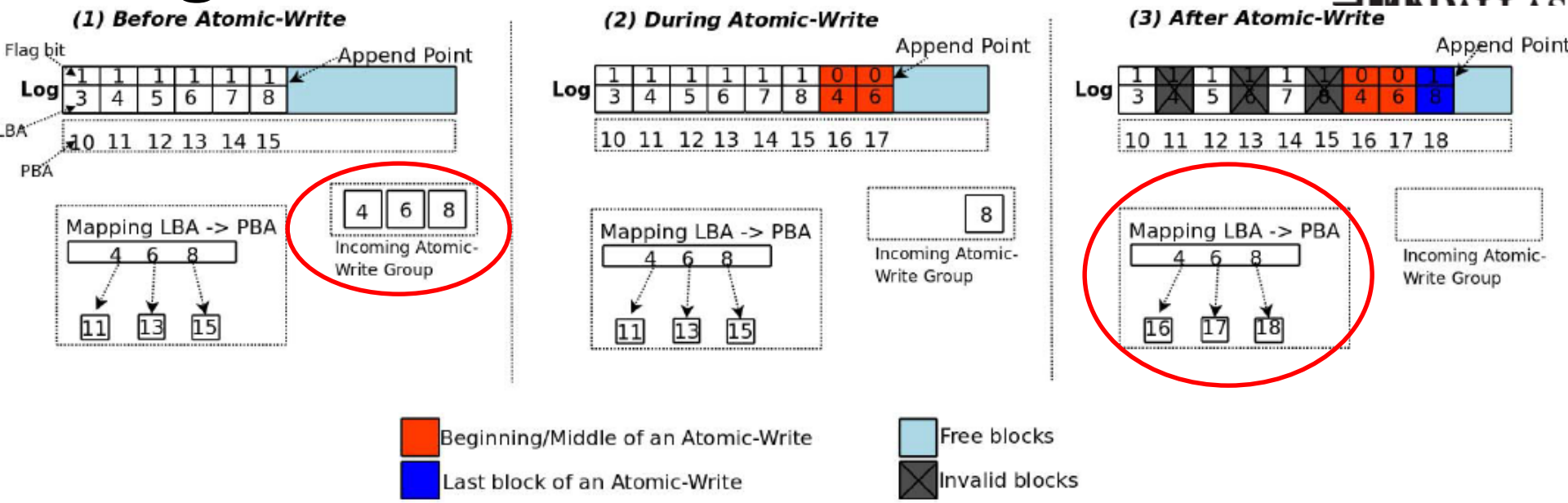
- **A new native storage interface, *"atomic-write"*, is proposed**
  - ***Multiple I/O operations can be issued as a single atomic unit***
  - ***Rollback is support*** for consistency when system failure
  - Exploits log-based ***mapping layer within FTL*** of SSD
- **Target application is database management system (DBMS)**
  - DBMS ***needs atomic-write*** to implement ACID property
- **A use-case of "atomic-write" is presented**
  - A DBMS application, ***MySQL,*** is modified to use "atomic-write"
- **Benefits of "atomic-write" are evaluated in the DBMS**
  - ***Performance*** improvement (latency, throughput)
  - ***Write volume*** reduction (wear-out of SSD)

DBMS's Write Atomicity



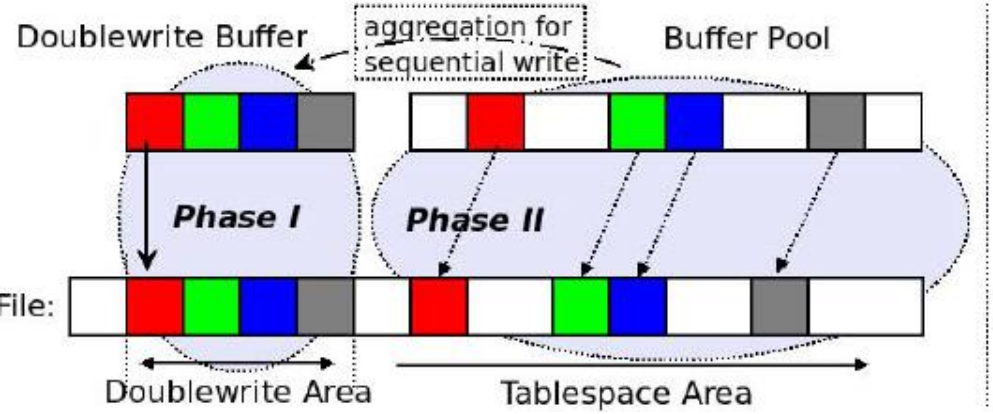
- **DBMS applications want to achieve write atomicity**
- **Traditional atomicity (with HDD or SSD)**
  - DBMS (transactional log), OS (metadata journaling, copy-on-write)
  - Complicated protocols have been designed in operating systems
- **Proposed atomicity using "atomic-write" as an I/O interface**
  - A new interface "atomic-write" can exploit the inner workings of FTL
  - Offloads write atomicity from other layers into FTL

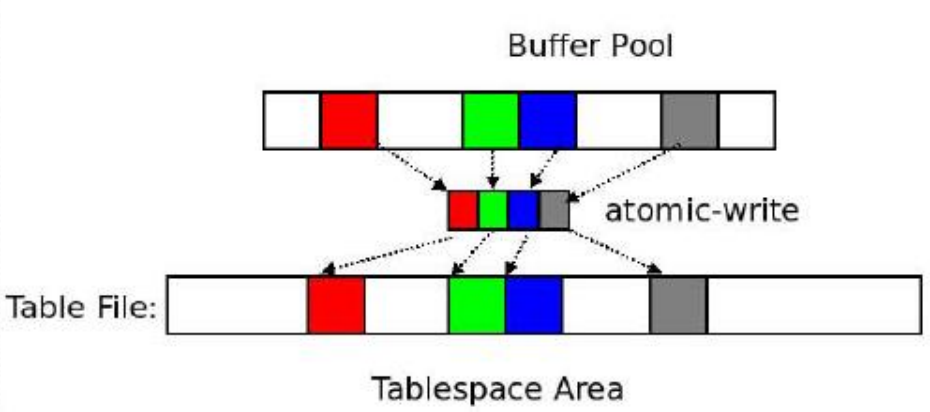
Log-based FTL



- **Log-based FTL**
  - All writes to the media are sequentially appended to "tail of the log"
  - FTL manages a mapping of LBA to PBA
  - ***"Out-of-place updates" allow "old version of data" to be used again, when system failure occurs during new writes***
- **A flag bit is added to the log for "atomic-write" support**
- **During an "atomic-write",**
  - First and subsequent writes are marked as "0", the flag bit is set to "1" only for the final block write

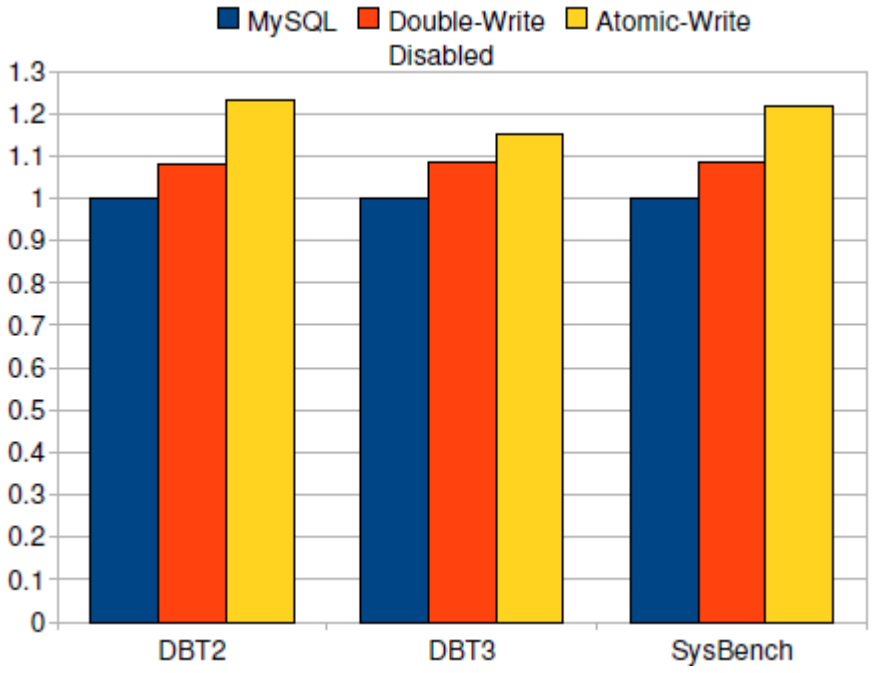
MySQL Optimization with Atomic-Write

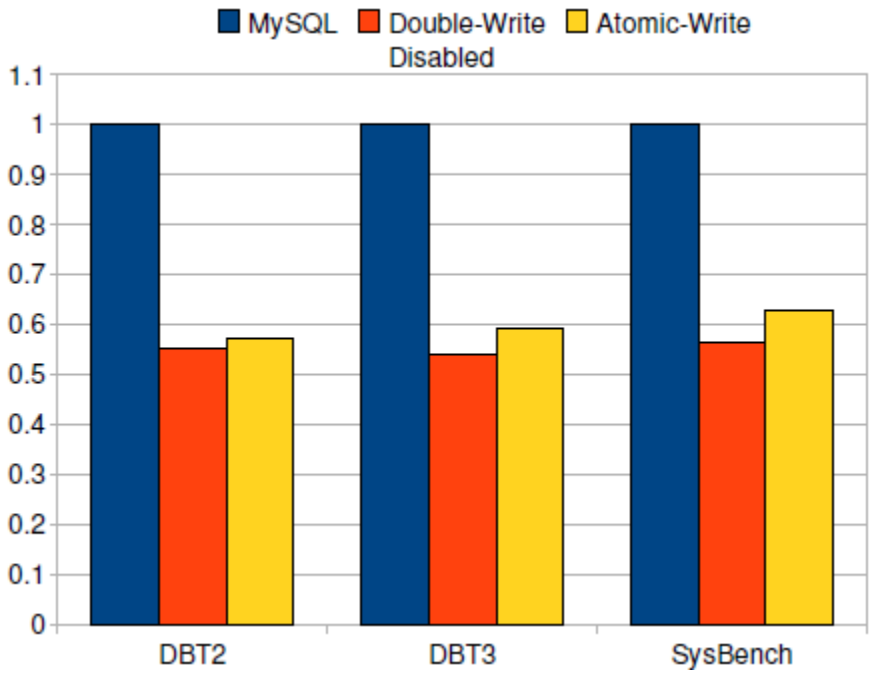




- **Double-write of MySQL (original operations, left figure)**
  - ***Phase I:*** copies discrete data pages from memory buffer pool into a dedicated memory area (doublewrite buffer), then sequentially and synchronously written to media
  - ***Phase II***: re-writes the same individual data pages to their final locations in the media
- **Replacing double-write with "atomic-write"**
  - MySQL is modified to use "atomic-write" supported by SSD
  - "Atomic-write" removes performing Phase I
  - ***"Write-twice" to "write-once" can improve performance and durability***

Evaluations





- **Fusion-io** 320GB MLC NAND flash SSD
- "Atomic-write" is implemented in Fusion-io driver
- MySQL 5.1.49 is extended with "atomic-write"
- Comparison of ***MySQL***, ***double-write disabled***, and ***atomic-w***
- ***"Throughput"*** and ***"the size of data written"*** are shown

Conclusions

- ***SSDs open opportunities for higher order primitives*** (other than read/write) in storage interfaces
- ***Atomic-Write*** allows multiple write operations to be done as an atomic unit, which can be implemented ***by exploiting the existing FTL*** of SSD
- ***Upper layers with atomicity requirements can get benefits***
  - Existing complex mechanisms can be eliminated
  - Performance / durability can be improved