FUSION-IO®

Optimizing I/O Operations via the Flash Translation Layer

Gary Orenstein, VP of Products, @garyorenstein

FLASH PATH FORWARD

- Flash: new media or new architecture
- Flash Translation Layer Best Practices
- Optimization examples
  - File Systems
  - Caching
  - Database
- Developer opportunities

FLASH: NEW MEDIA OR NEW ARCHITECTURE

Is GPS technology a new map or new architecture?

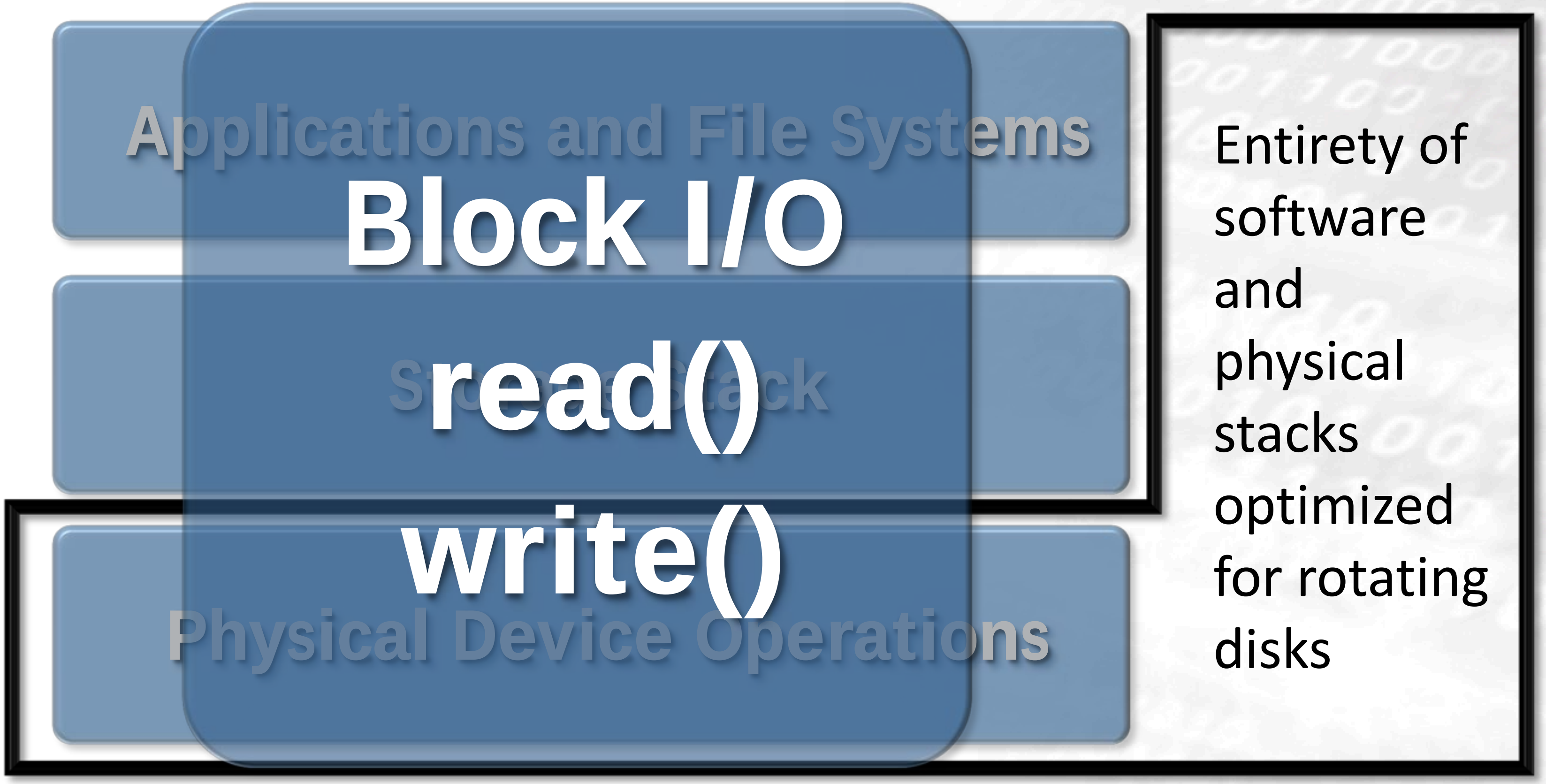
CONVENTIONAL ARCHITECTURE

Applications and File Systems

Storage Stack

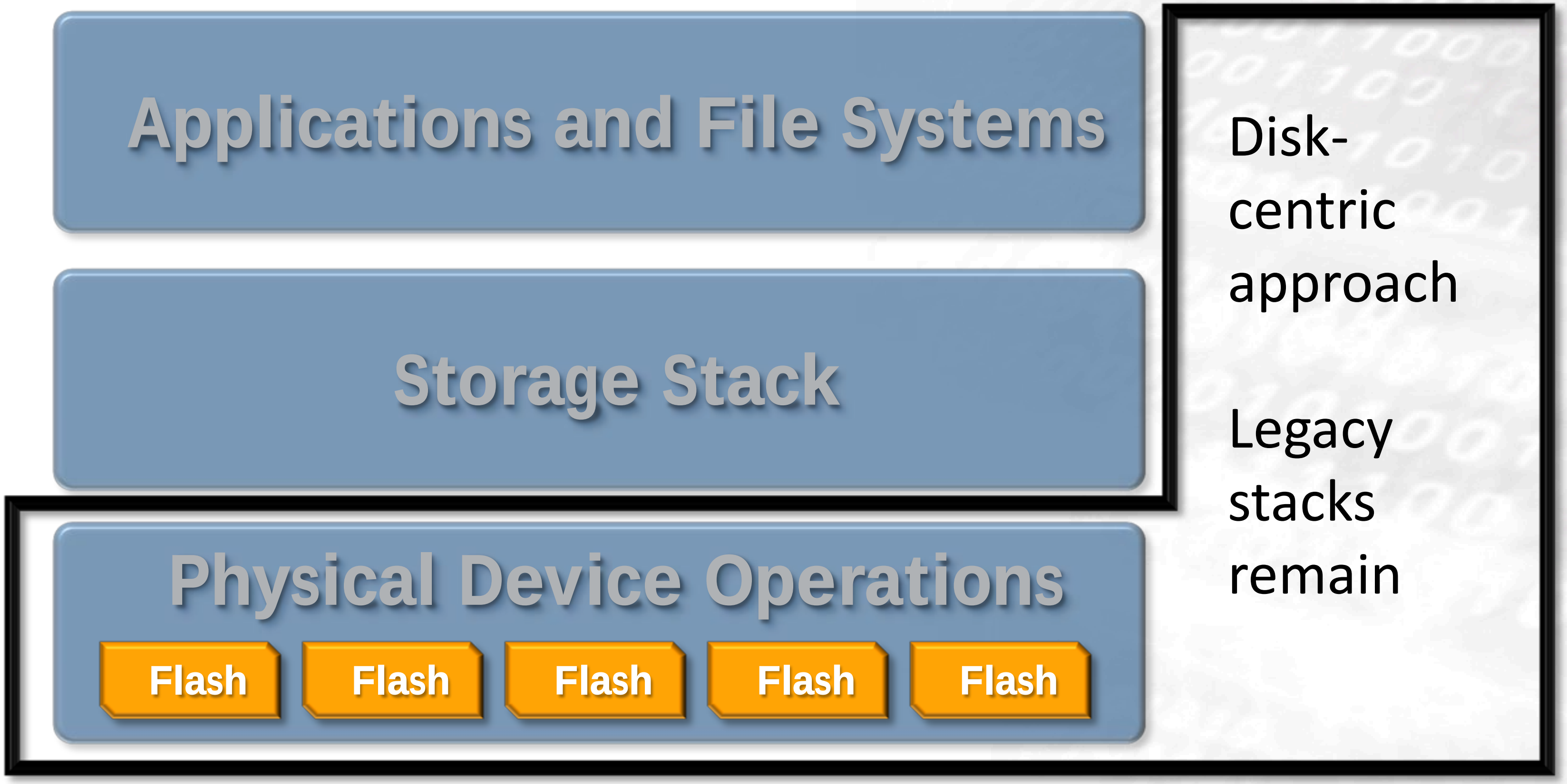
Physical Device Operations

STUCK ON DRIVES AND BLOCK I/O



Entirety of software and physical stacks optimized for rotating disks

FLASH ONLY AS A FAST DISK



Applications and File Systems

Storage Stack

Physical Device Operations

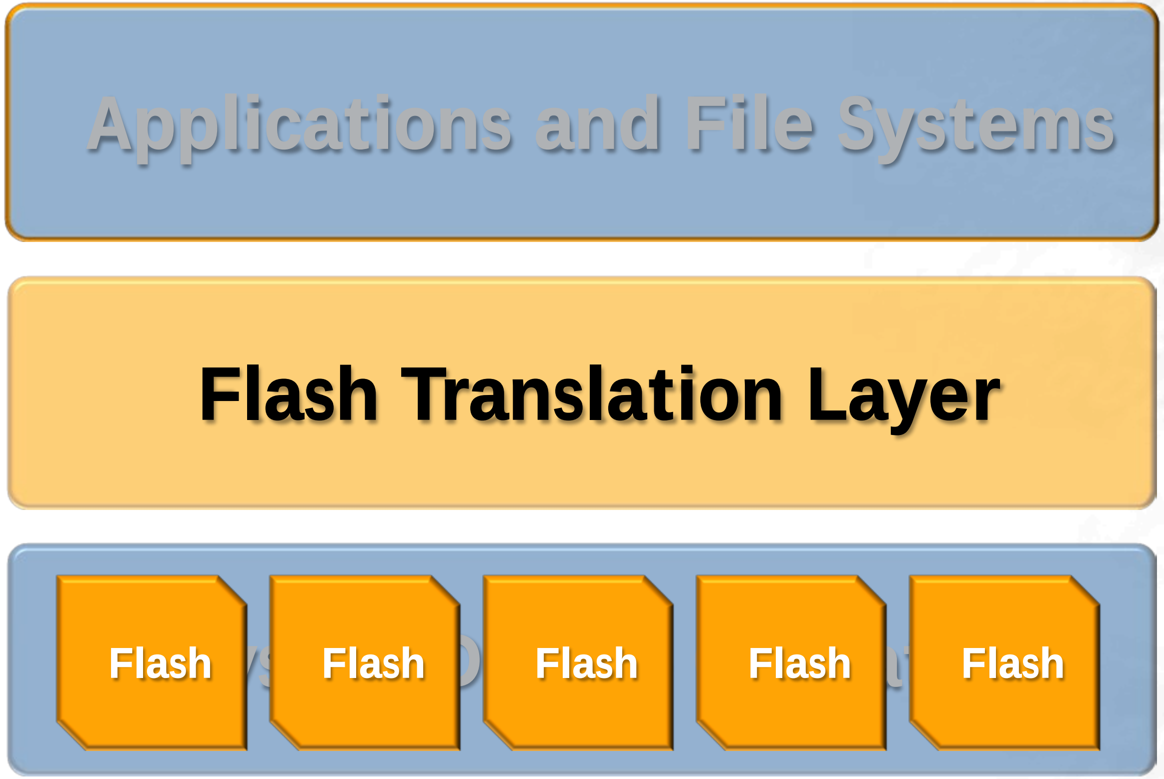
Disk-centric approach

Legacy stacks remain

BUT FLASH IS DIFFERENT

- Asymmetric read/write latencies
- Write-impact on durability
- Unique erase characteristics

FLASH AS A NEW ARCHITECTURE



Applications and File Systems

Flash Translation Layer

Flash Flash Flash Flash Flash

Flash-centric approach

Retain backwards compatibility with conventional block I/O

FLASH TRANSLATION LAYER 101

Input

Logical Block Address (LBA)

Flash Translation Layer

Output

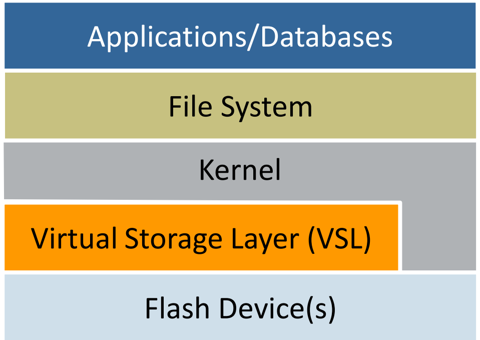
Commands to NAND flash

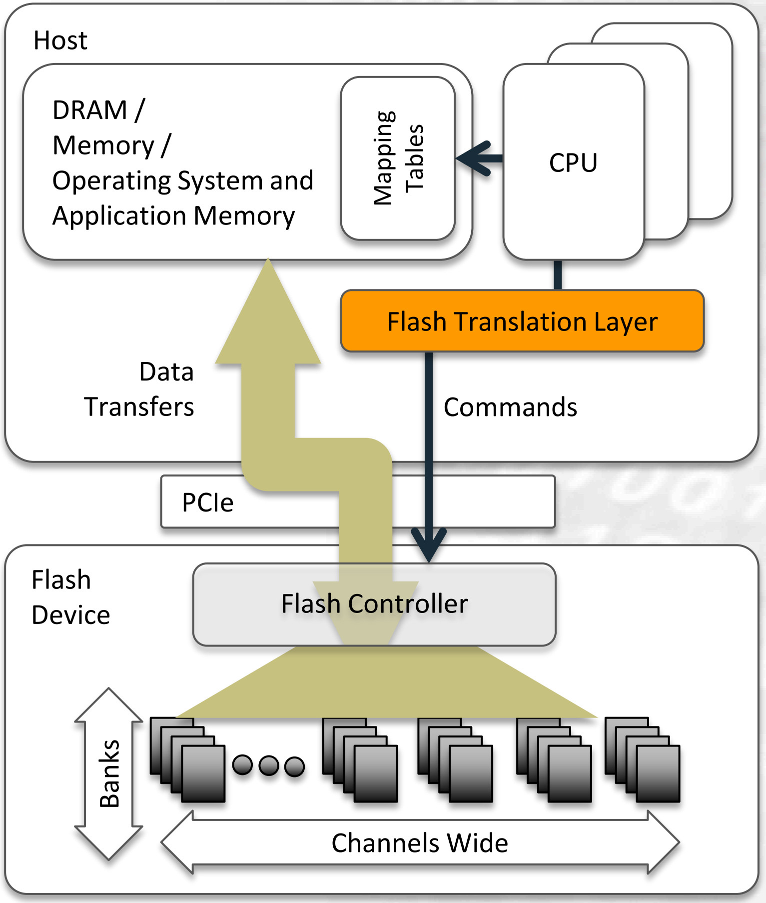
OPTIMIZING SOFTWARE STACKS FOR FLASH

- Virtualize the storage abstraction layer
- Provide a large virtual block address space
- Be backwards compatible with conventional block I/O
- Deliver new capabilities
  - Combine virtualization with intelligent translation and allocation strategies; hide bulk erasure latencies; perform wear leveling

OPTIMIZED FTL

- Sophisticated architecture
  - maximum performance
- Intelligent software
  - advanced features





BENEFITS OF VIRTUAL ADDRESS SPACE

Large, virtualized, block address space provides:

1. Client software direct access to flash memory
    - single level store fashion
    - across multiple flash memory devices

2. Frees applications and databases from details of virtual to physical flash memory pages

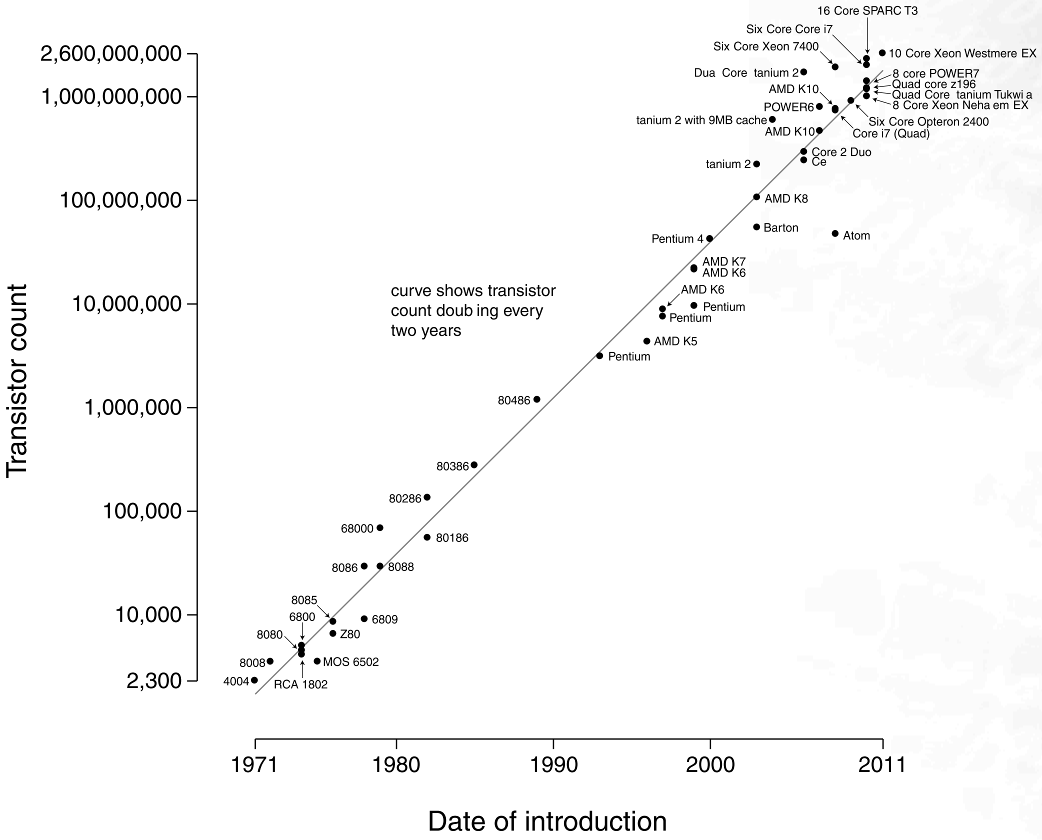
3. Flat, virtual block-addressed space is backwards compatible with conventional block I/O

HOST-CENTRIC APPROACH

- Cooperate with hardware support
- Maintain virtual to physical mappings
- Handle multiple devices
- Log structured allocation strategy
  - Bulk erasure
  - Wear leveling
  - Bad page recovery
- Richer interface than currently available

BENEFIT OF THE X86 ECOSYSTEM

Microprocessor Transistor Counts 1971-2011 & Moore's Law



http://www.physicsforums.com/showthread.php?t=506828

File Systems

DFS: A FILE SYSTEM FOR VIRTUALIZED FLASH STORAGE

DFS: A File System for Virtualized Flash Storage

William K. Josephson

Lars A. Bongo

David Flynn

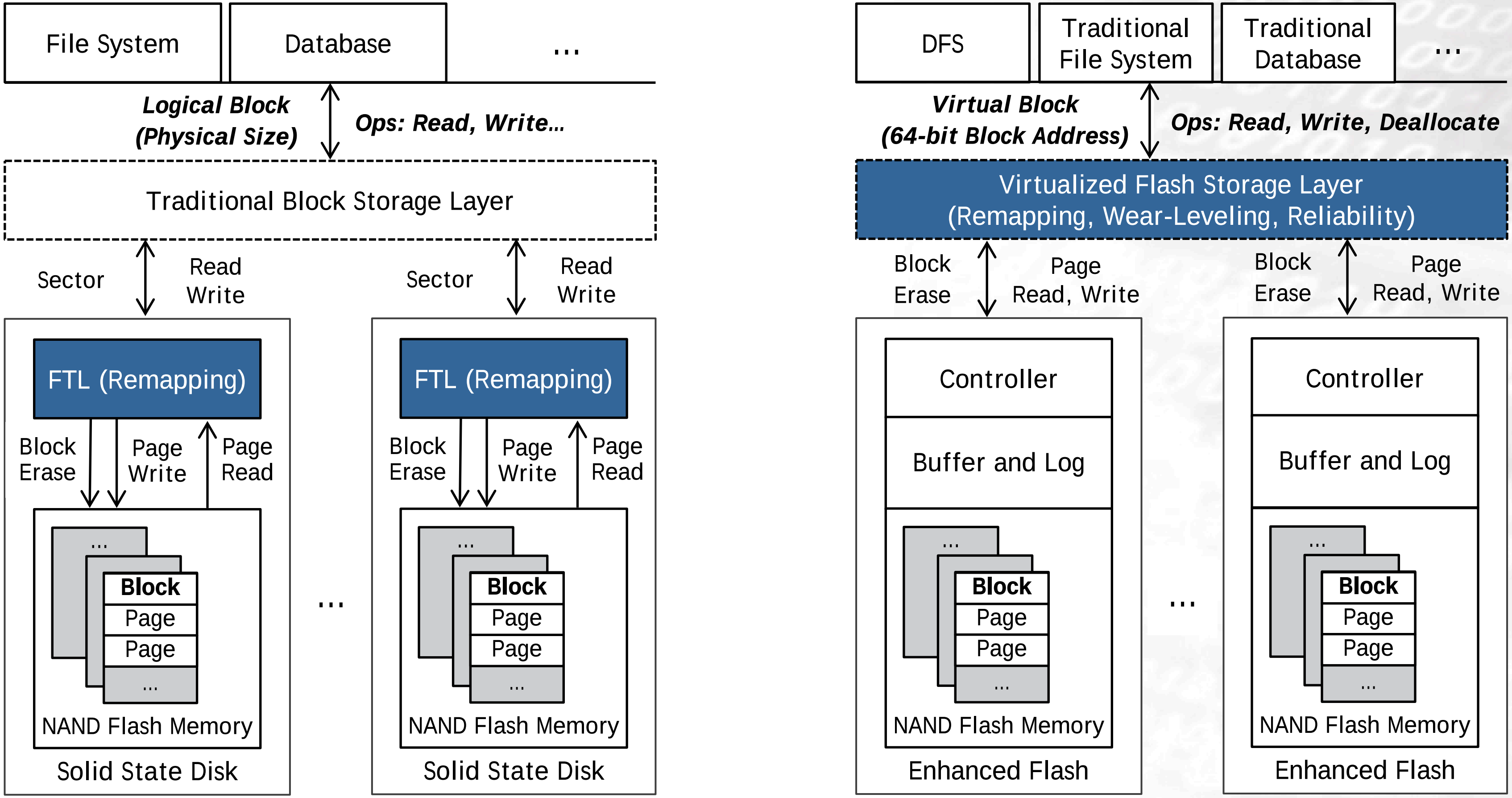
Kai Li

Abstract

1 Introduction

http://www.usenix.org/event/fast10/tech/full_papers/josephson.pdf

FLASH STORAGE ABSTRACTIONS



DFS

- Full fledged UNIX file system
- Employ virtualized flash storage layer’s
  - Large virtualized addressed space
  - Direct flash access
  - Crash recovery mechanisms

DFS PERFORMANCE

Device	Read IOPS	Write IOPS
Conventional SSD and FTL	33,400	3,120
Optimized SSD and virtual flash storage layer	98,800	71,000

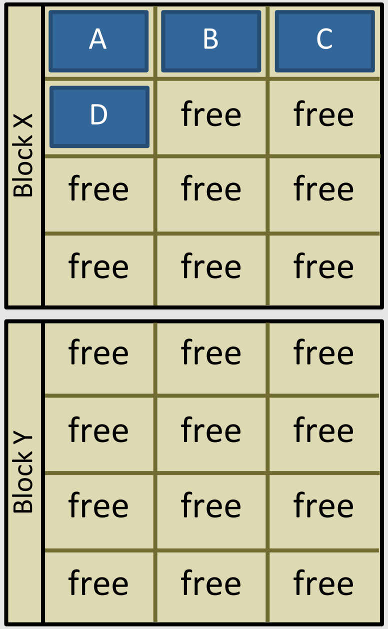
http://www.usenix.org/event/fast10/tech/full_papers/josephson.pdf

DFS SIMPLICITY – LINES OF CODE

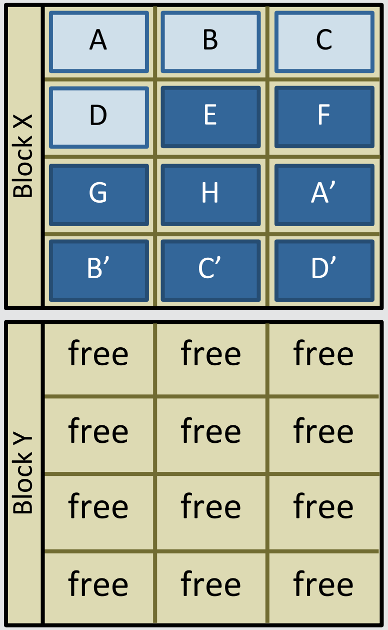
Module	DFS	Ext3
Headers	392	1583
Kernel Interface (Superblock, etc.)	1625	2973
Logging	0	7128
Block Allocator	0	1909
I-nodes	250	6544
Files	286	283
Directories	561	670
ACLs, Extended Attrs.	N/A	2420
Resizing	N/A	1085
Miscellaneous	175	113
Total	3289	24708

Caching

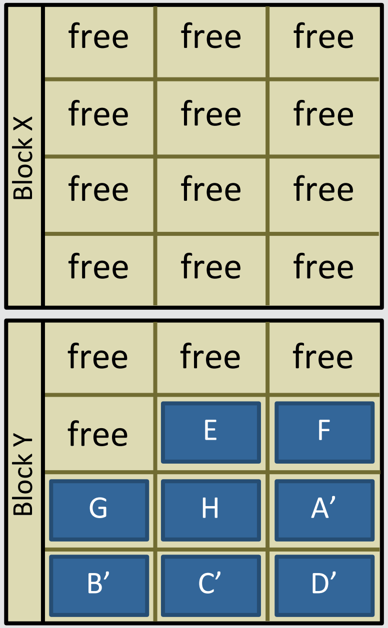
ALL FLASH TRANSLATION LAYERS DO GARBAGE COLLECTION



1. Four pages (A-D) are written to a block (X). Individual pages can be written at any time if they are currently free (erased)



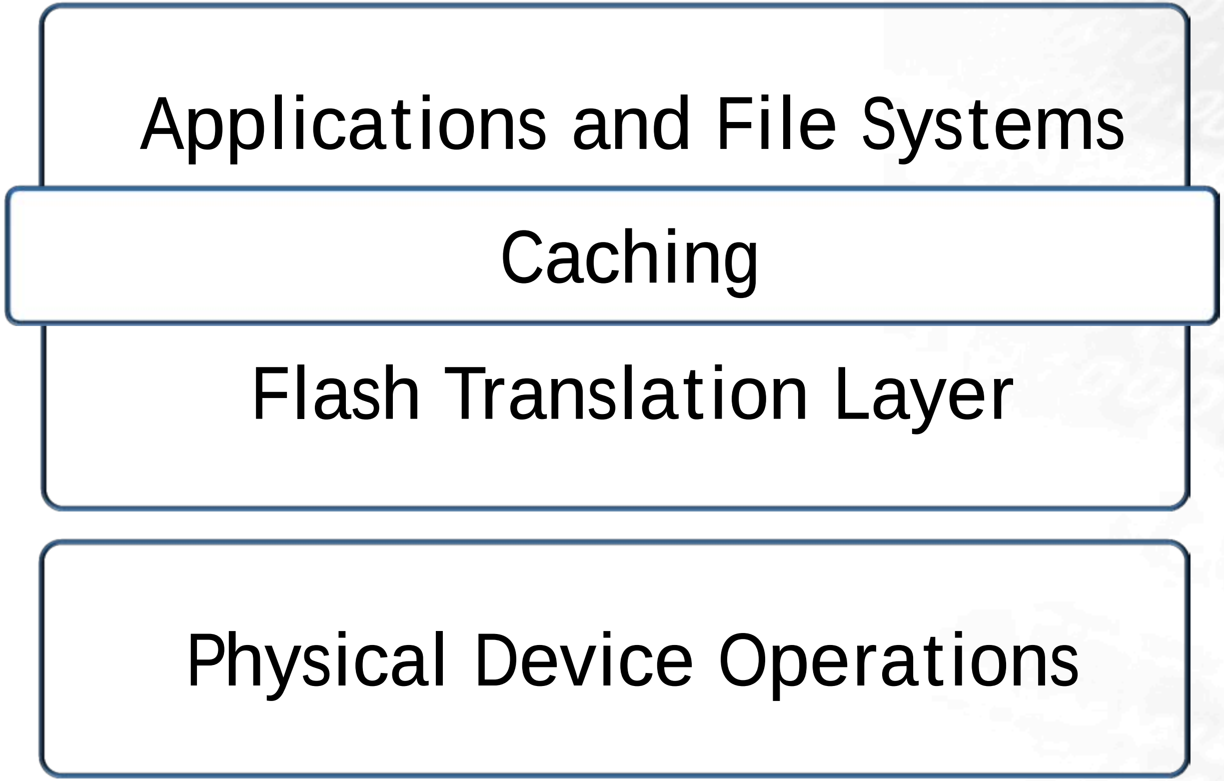
2. Four new pages (E-H) and four replacement pages (A’-D’) are written to the block (X). The original A-D pages are now invalid (stale) data, but cannot be overwritten until the whole block is erased.



3. In order to write to the pages with stale data (A-D) all good pages (E-H & A’-D’) are read and written to a new block (Y) then the old block (X) is erased. This last step is *garbage collection.*

http://en.wikipedia.org/wiki/Write_amplification#cite_note-L_Smith-5

INTELLIGENT CACHING LAYER



Applications and File Systems

Caching

Flash Translation Layer

Physical Device Operations

INTEGRATION BENEFITS

- Flash translation and garbage collection is complex
- TRIM demonstrates that flash needs information from upstream stack to perform efficiently
- TRIM is only a first step

- Caches maintain intelligence on data
- Uncoupled caching and FTL layers could be working against each other
- Linking cache intelligence to FTL can improve FTL efficiency, write performance, endurance

Databases Atomic Writes

BEYOND BLOCK I/O: RETHINKING TRADITIONAL STORAGE PRIMITIVES

Beyond Block I/O: Rethinking Traditional Storage Primitives *

Xiangyong Ouyang†1, David Nellans†, Robert Wipfel†, David Flynn†, Dhabaleswar K. Panda1

† FusionIO and 1The Ohio State University

Abstract

1 Introduction

http://www.cse.ohio-state.edu/~zhang/hpca11-submitted.pdf

IT IS ABOUT TRANSACTIONS

- Building block of applications and databases

TRANSACTION SEMANTICS

- Data Integrity
- Concurrency
- Crash Recovery

TOKEN ACID SLIDE ☺

- **Atomicity**
  - Database modifications must follow an "all or nothing" rule
- **Consistency**
  - Any transaction the database performs will take it from one consistent state to another.
- **Isolation**
  - Other operations cannot access data that has been modified during a transaction that has not yet completed
- **Durability**
  - Ability to recover the committed transaction updates against any kind of system failure (hardware or software).
- http://en.wikipedia.org/wiki/ACID

TRANSACTIONAL SEMANTICS APPLY

Across:

- Applications
- File Systems
- Databases
- Web Services
- Search Engines
- Mission Critical Computing

ATOMIC WRITES

- Batch multiple I/O operations into a single logical group
- Multiple I/Os are persisted as a whole or rolled back upon failure

ATOMIC WRITES TODAY

- Handled by
  - Applications
  - Databases
  - File Systems
- Guarantee the consistency and integrity of data

- Databases support atomic write through
  - Logs
  - Locks
  - Buffers
  - Process Management

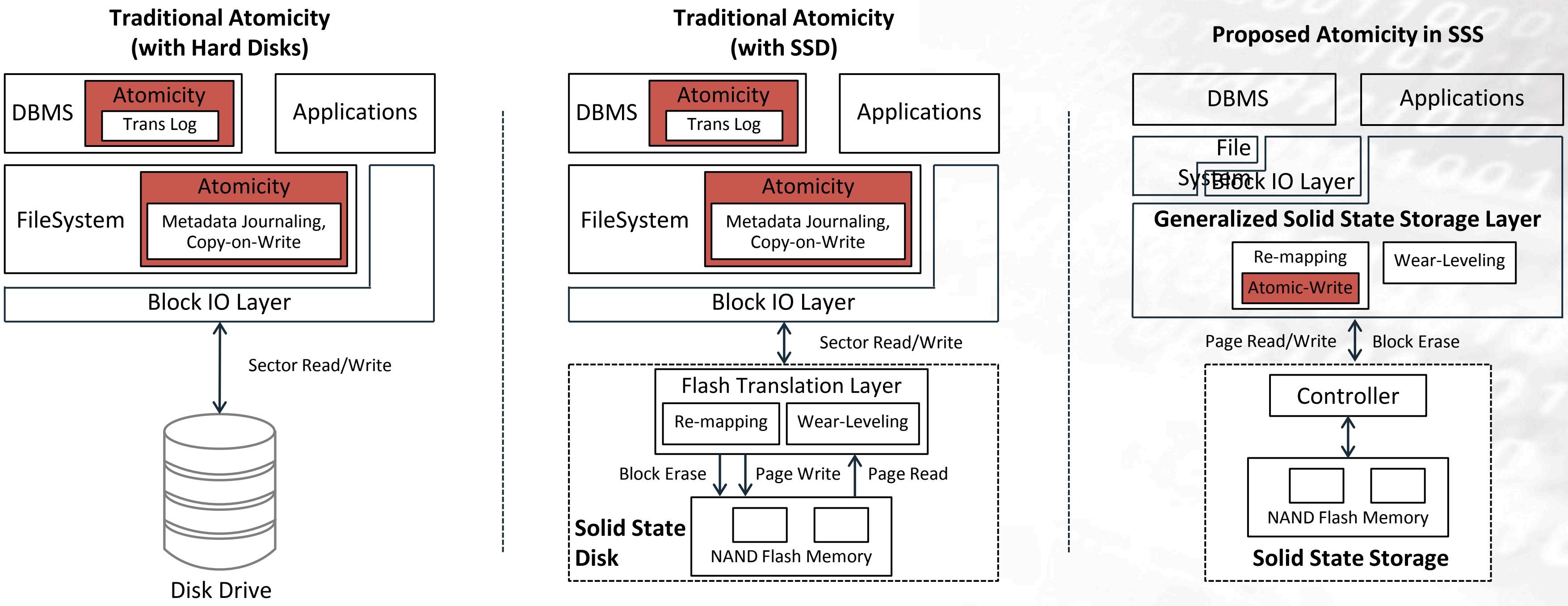
NOT ALL APPLICATIONS THE SAME

- Many do not fit the RDBMS model perfectly
- Opportunities exist to
  - Optimize efficient access
  - Provide more control
  - Improve application specific data layout
  - Improve application specific data access mechanisms

ATOMIC WRITE POTENTIAL

- Leverage underlying, log-based Flash Translation Layer
- Reduce load on applications and databases
- Simplify Atomic Write execution

ATOMIC WRITES – OPTIMIZED



Moving the Atomic-Write Primitive into Storage Stack

INITIAL DEMONSTRATION

- MySQL and InnoDB

- Early testing
  - 33% speedups to TPC-C and TPC-H
  - Reduced write bandwidth requirement by 43%
  - Increased endurance with write reduction

Call to Action

FLASH MERITS A NEW SOFTWARE ARCHITECTURE

- Host-based FTLs integrate and scale with applications, examples include
  - File Systems
  - Caching
  - Databases
- Power of FTL no longer restricted by traditional block interfaces
- Opportunity for performance, simplicity and reliability improvements

For more

go@fusionio.com

www.fusionio.com

THANK YOU