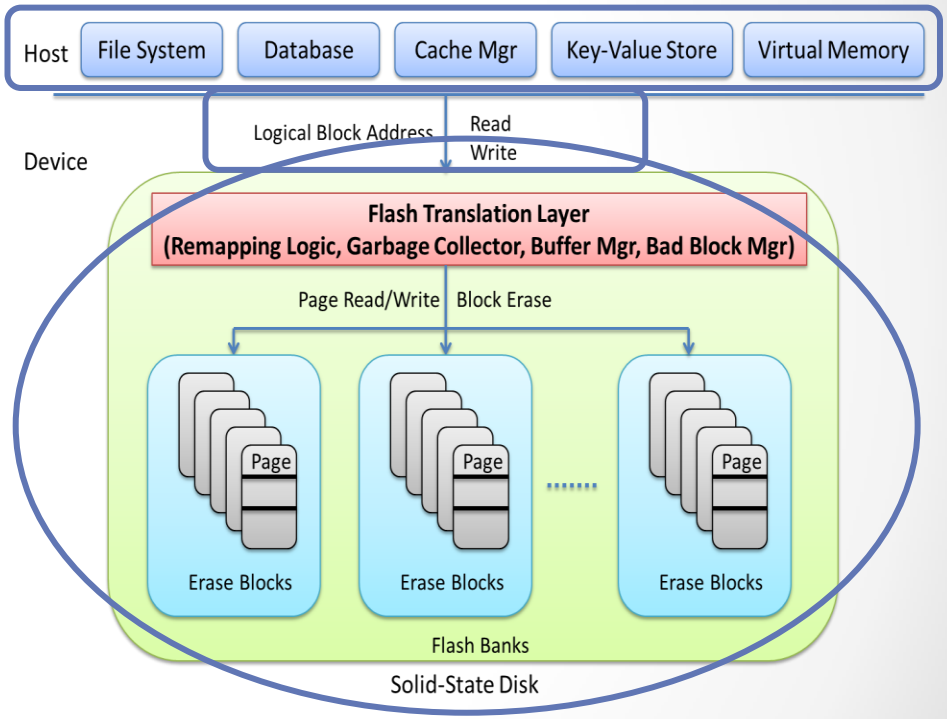
Getting Real: Lessons in Transitioning Research Simulations into Hardware Systems

Mohit Saxena, Yiying Zhang

Michael Swift, Andrea Arpaci-Dusseau and Remzi Arpaci-Dusseau

Flash Storage Stack Research

- SSD Design
  - Flash management
- OS & Applications
  - File systems, block cache, K-V stores
- Device Interface
  - read, write, trim



How to evaluate new SSD designs?

1. Modify SSD
   - Replace firmware/FTL
   - Are you a device vendor?

2. FPGA Prototype
   - Flexible and fast
   - Hard and time-consuming

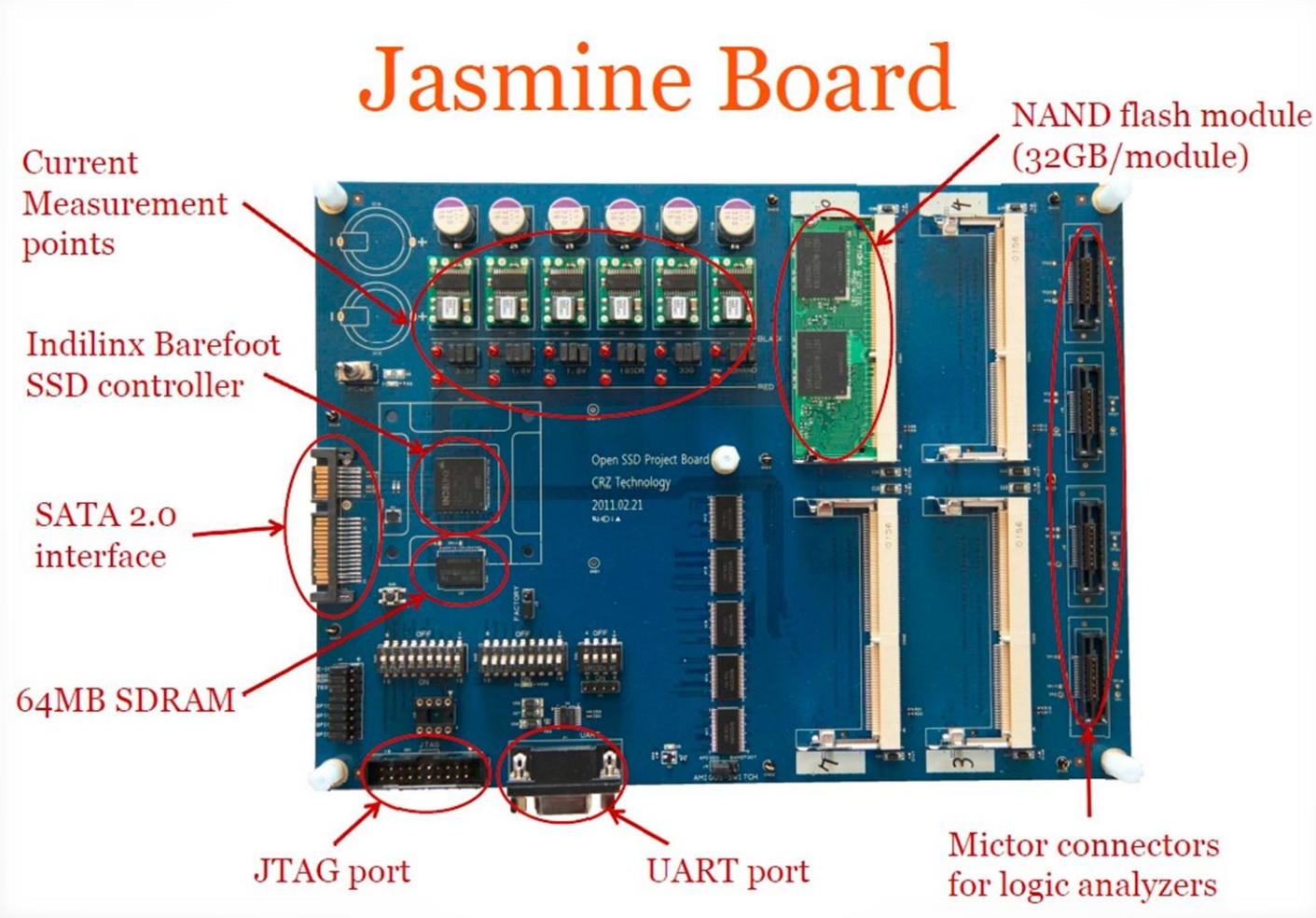
3. Simulator or Emulator
   - Replay block traces
   - Implement device models

Simulator/Emulator Limitations

- Real hardware performance
  - SSDs are complex: banks, packages, planes
- Interaction with software stack
  - SSDs are sensitive to OS & app behavior
- Trace replay
  - Unable to model request timing dependencies & new commands

FAST year	Unmodified SSDs	Simulator	Hardware
2011	2	5	**0**
2012	2	7	**0**
2013	2	3	**2**

Approach 4. SSD Prototyping Kit



OpenSSD Hardware Platform

In this Talk

- New SSD and Interface Designs
  - Research Simulation → Hardware Prototype
  - FlashTier's Solid-State Cache (SSC) [EuroSys '12]
  - Nameless Write SSD [FAST '12]

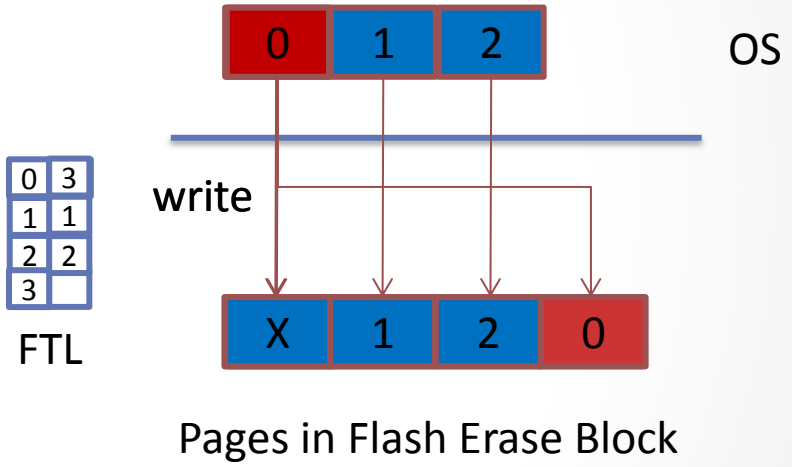
- Challenges, Experiences and Lessons
  - General insights applicable to other SSD designs and interfaces

Talk Outline

- Introduction
- Background
  - OpenSSD Hardware Platform
  - SSC and Nameless Write SSD Interfaces
- Prototyping Experiences
- Evaluation
- Conclusions

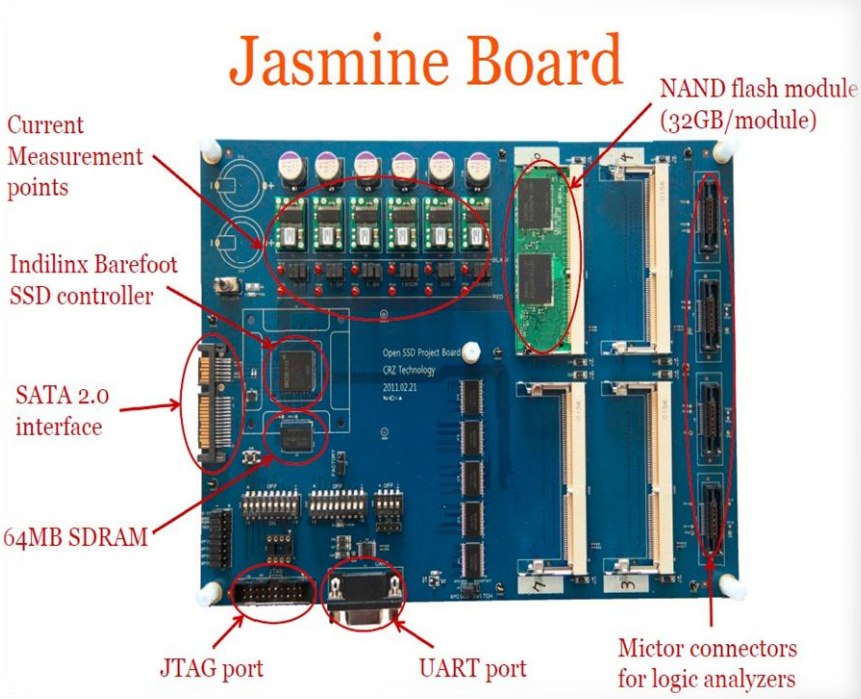
Flash SSD 101

- Remap in-place writes
  - Address translation
  - Log-structuring
  - Garbage collection

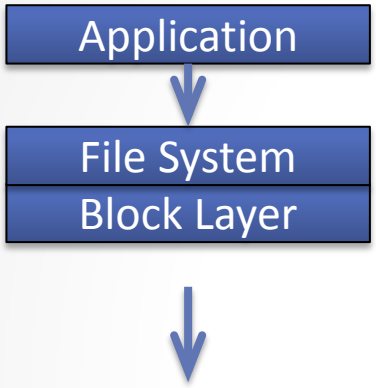


OpenSSD Hardware Platform

- Hardware: commodity
  - Indilinx Barefoot ARM SSD controller
  - 64 MB DRAM, 128 GB NAND Flash
- Software: reference
  - Open-source FTL
- Interface: standard
  - SATA read & write
  - UART serial debugging



Solid-State Disk (SSD)

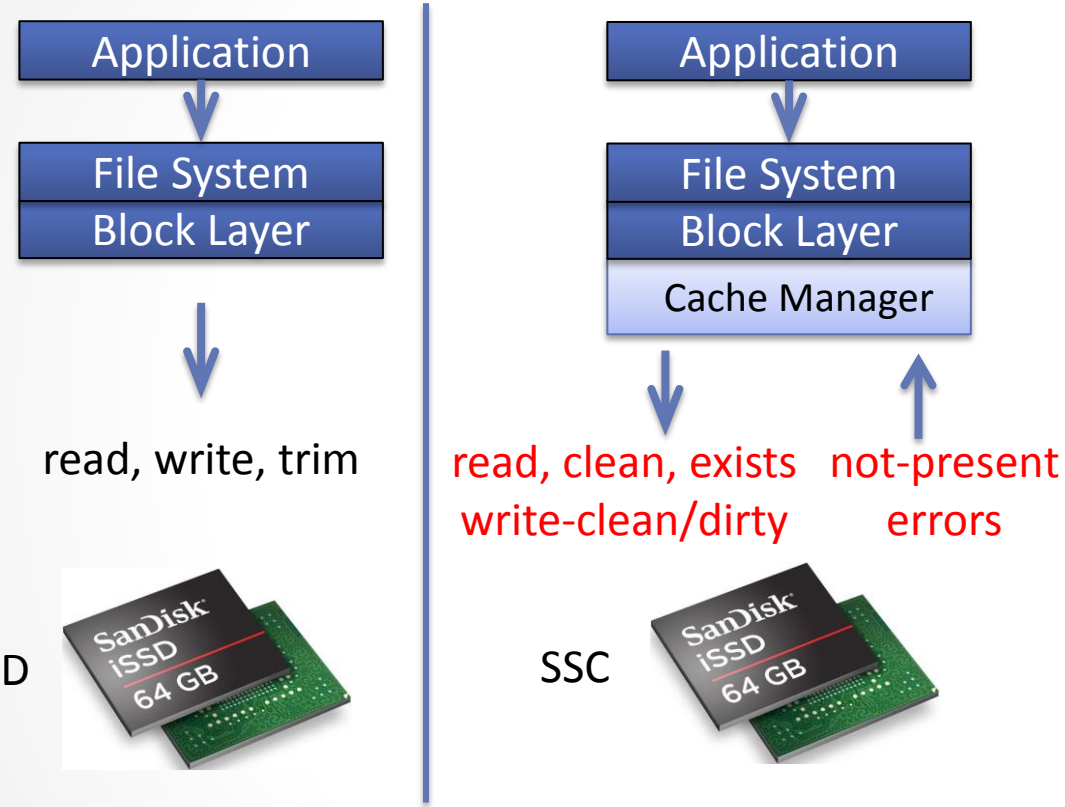


SSD

SSD Block Interface

- Emulate disk: persistent block store

Solid-State Cache (SSC)

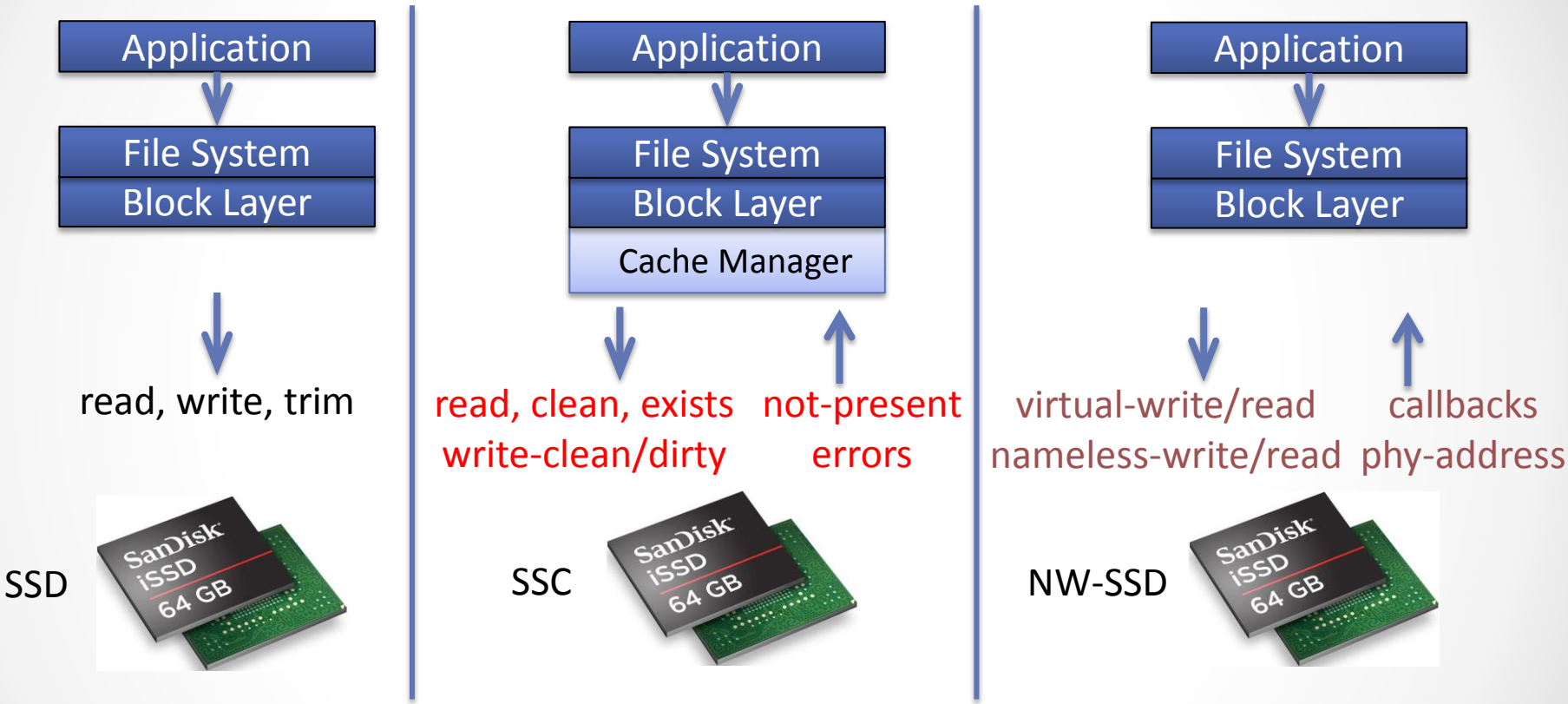


SSC Caching Interface

- Distinguish **clean** from **dirty** data
- Return not-present **read errors** for evicted blocks

➢ Fast and reliable SSC

Nameless-Write (NW-SSD)



Nameless-Write SSD

- **nameless-write/read** : physical address
- **migration callbacks** : relocated blocks

➢ Cheap and fast SSD

Summary of Interface Changes

New Interface Extensions	SSC	Nameless-Write SSD
Forward Commands	• write-dirty • write-clean • exists • clean	• nameless-write • physical-read • virtual-write • virtual-read
Return Values	• not-present read errors	• physical addresses
Device Responses		• migration-callbacks

Talk Outline

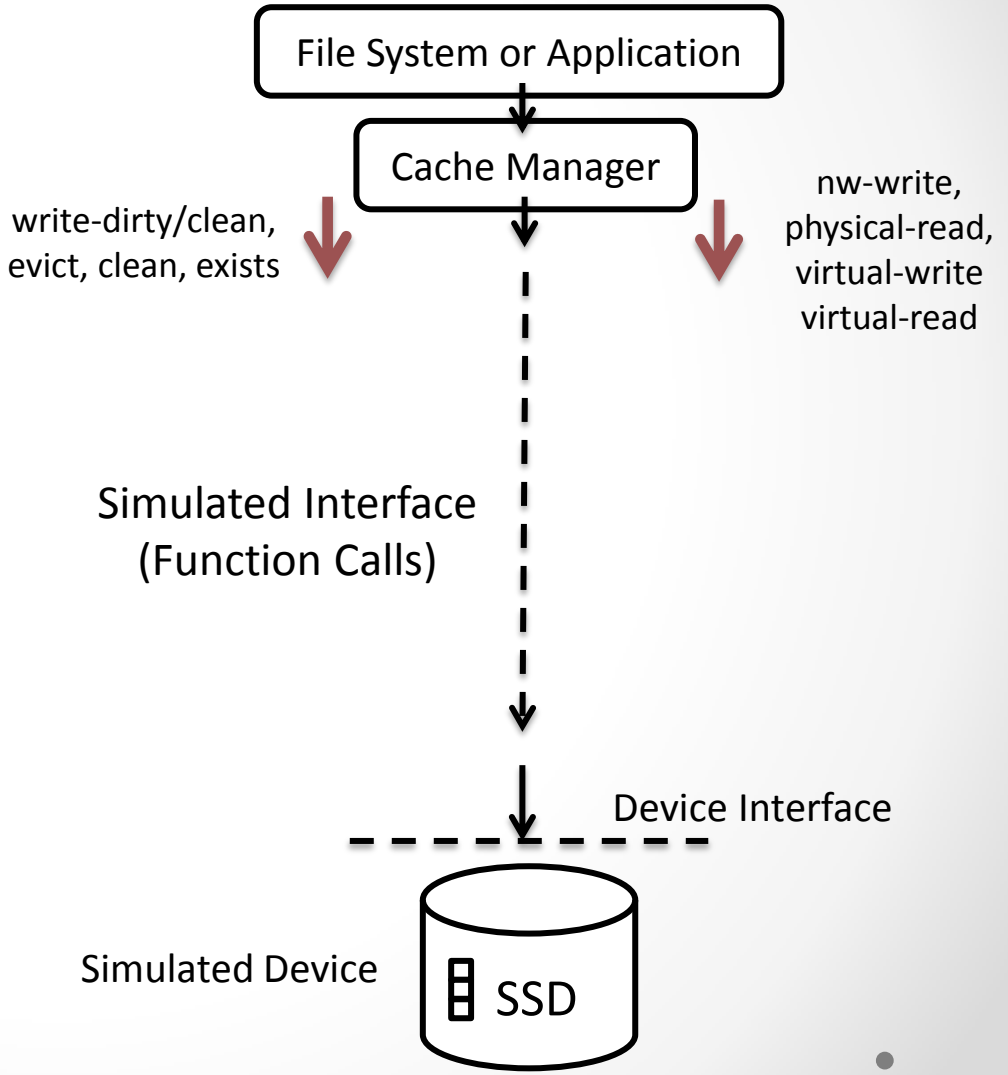
- Introduction
- Background
- Prototyping Experiences
  - Challenges
  - Solutions
  - Lessons
- Evaluation
- Conclusions

Prototyping Challenges

1. New Forward Commands
2. New Device Responses
3. Real Hardware Constraints for SSC and NW-SSD

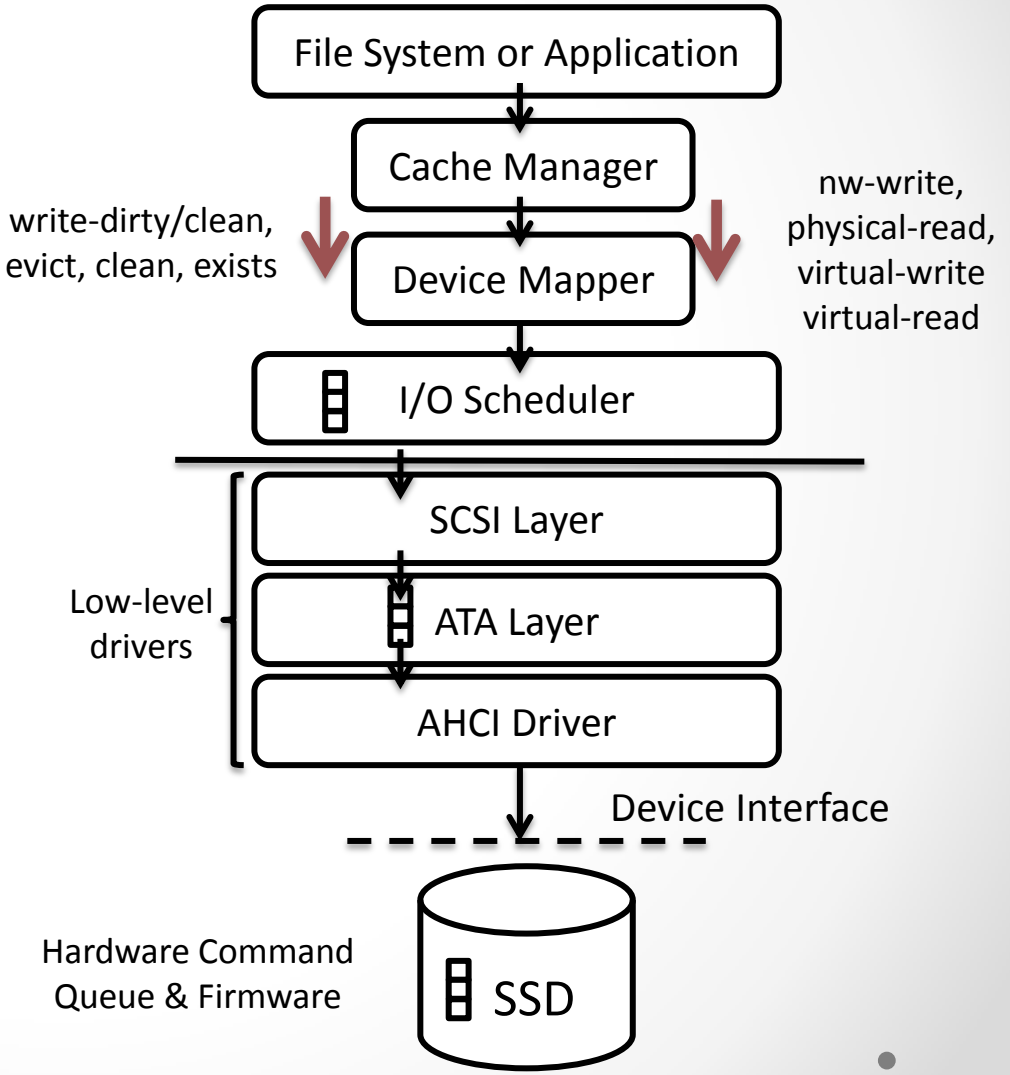
1. New Forward Commands

- Assumption
  - File system & cache manager directly interface with device



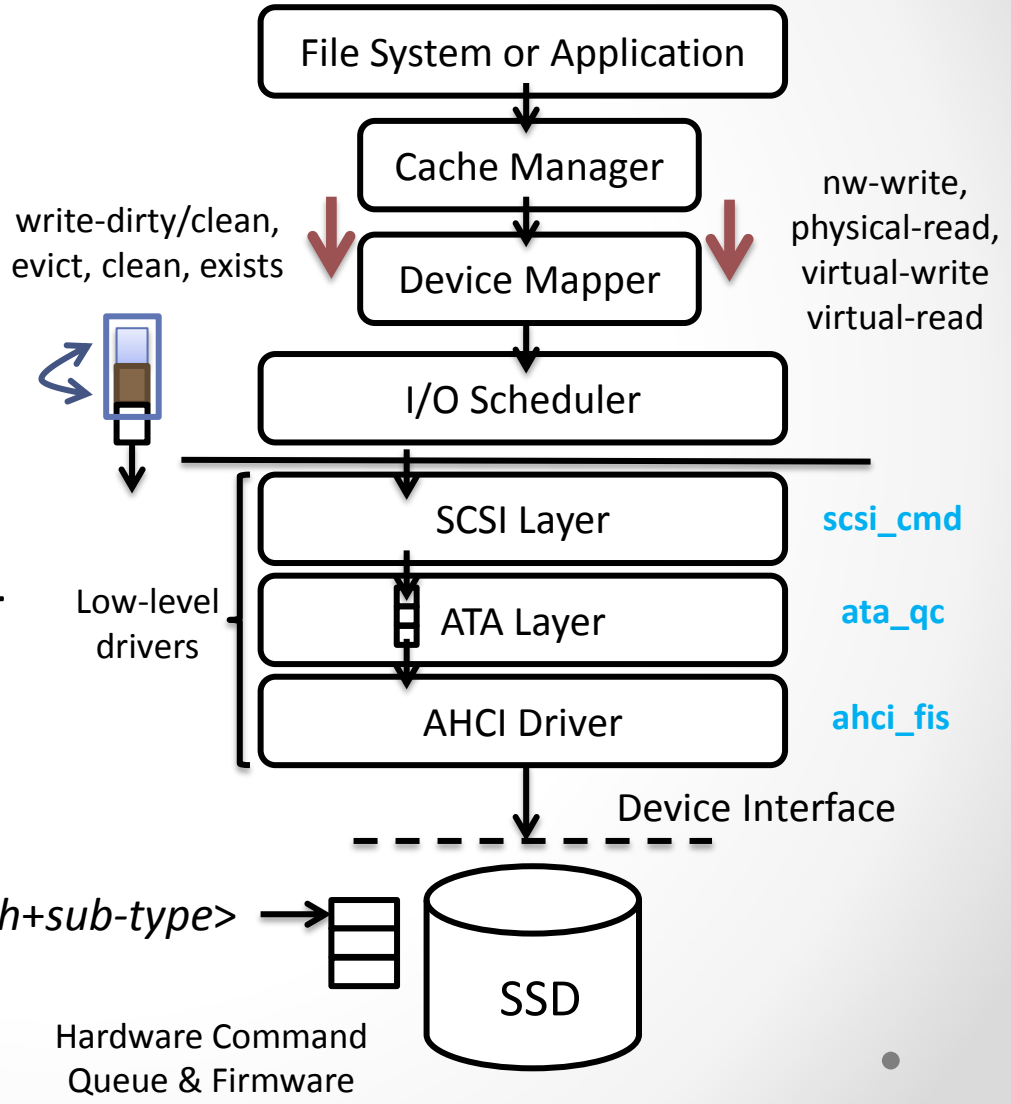
1. New Forward Commands

- Assumption
  - File system & cache manager directly interface with device
- Reality
  - Several intermediate OS, firmware & hardware layers



Implementing Forward Commands

- Solutions
  - Disallow merging & re-ordering
  - Add new command sub-types
  - Mask command sub-type in firmware for hardware queues

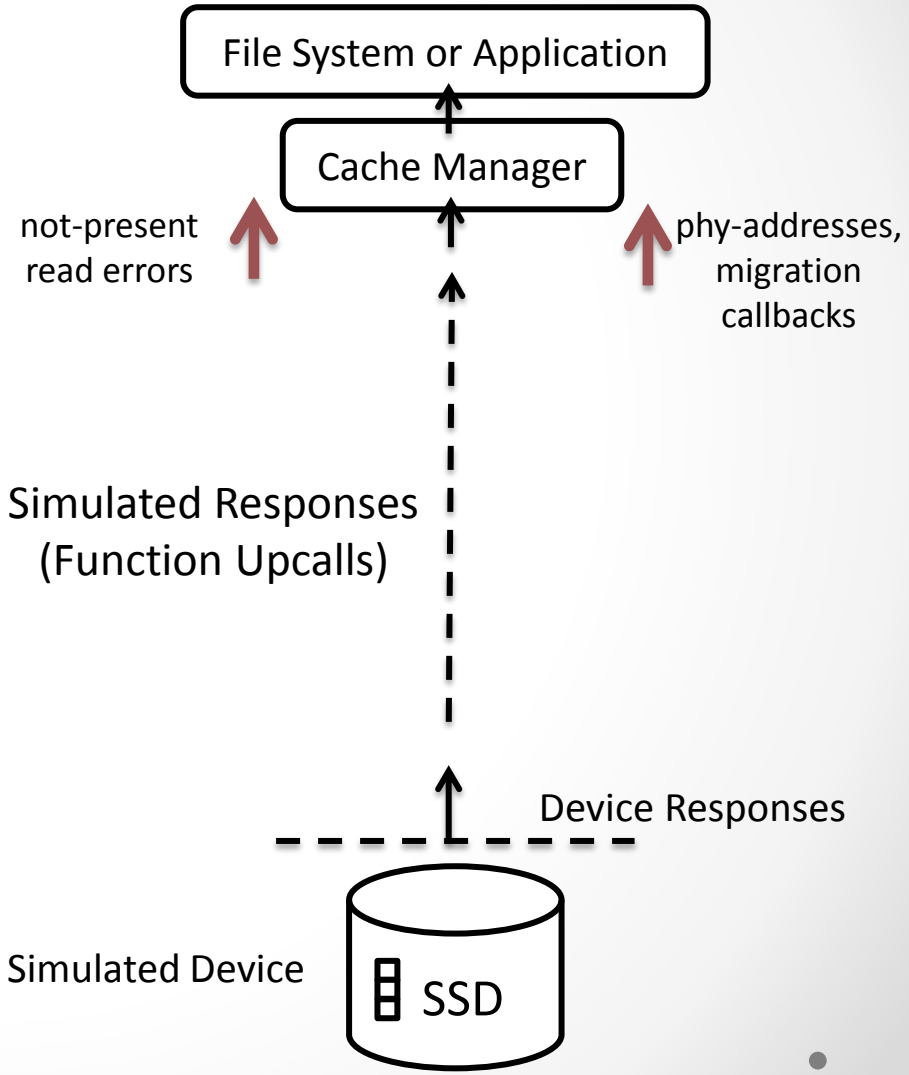


Lesson 1
Designing New Forward Commands

- Research lesson: consider all layers of OS
  - I/O scheduler: merging and re-ordering
  - Storage device drivers: adding sub-types
- Research lesson: consider complete SSD ecosystem
  - Firmware: encoding sub-types
  - Hardware: accelerating new command queues

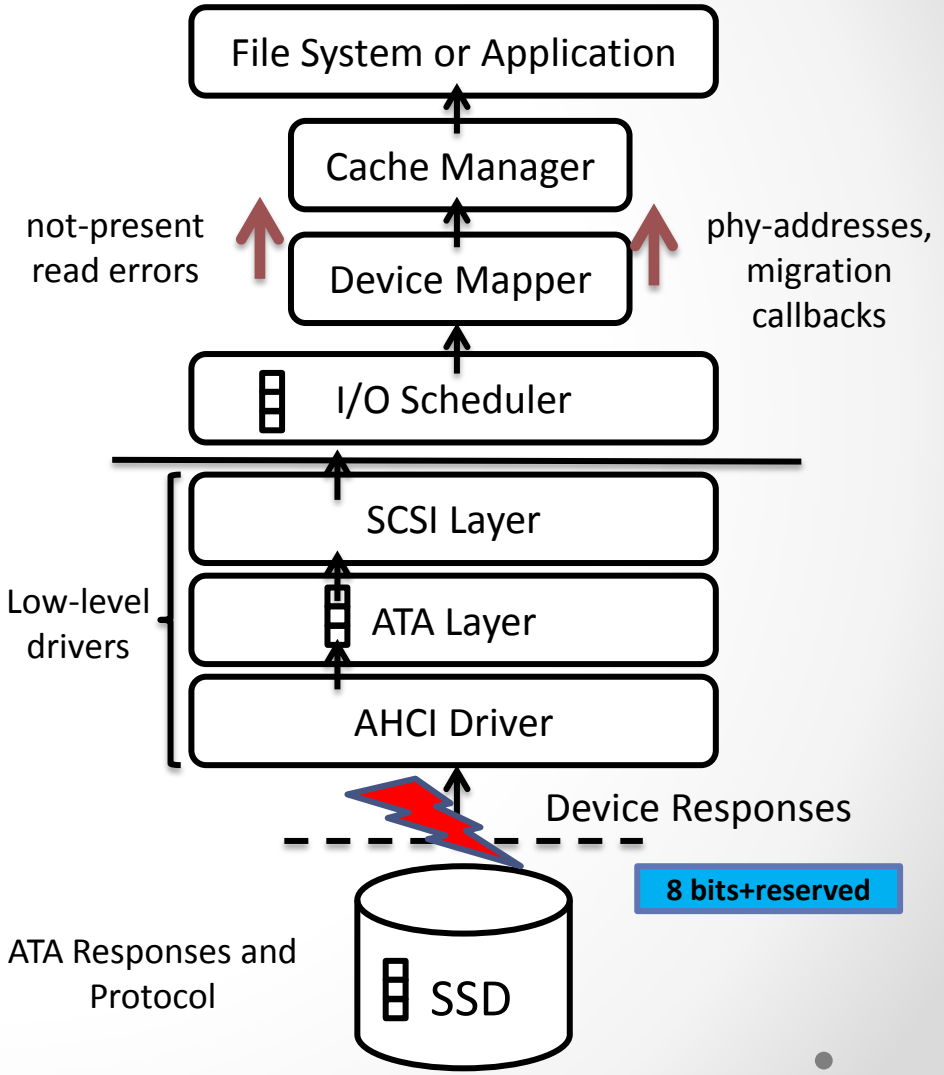
2. New Device Responses

- Assumption
  - Device can directly send new responses to cache manager and file system



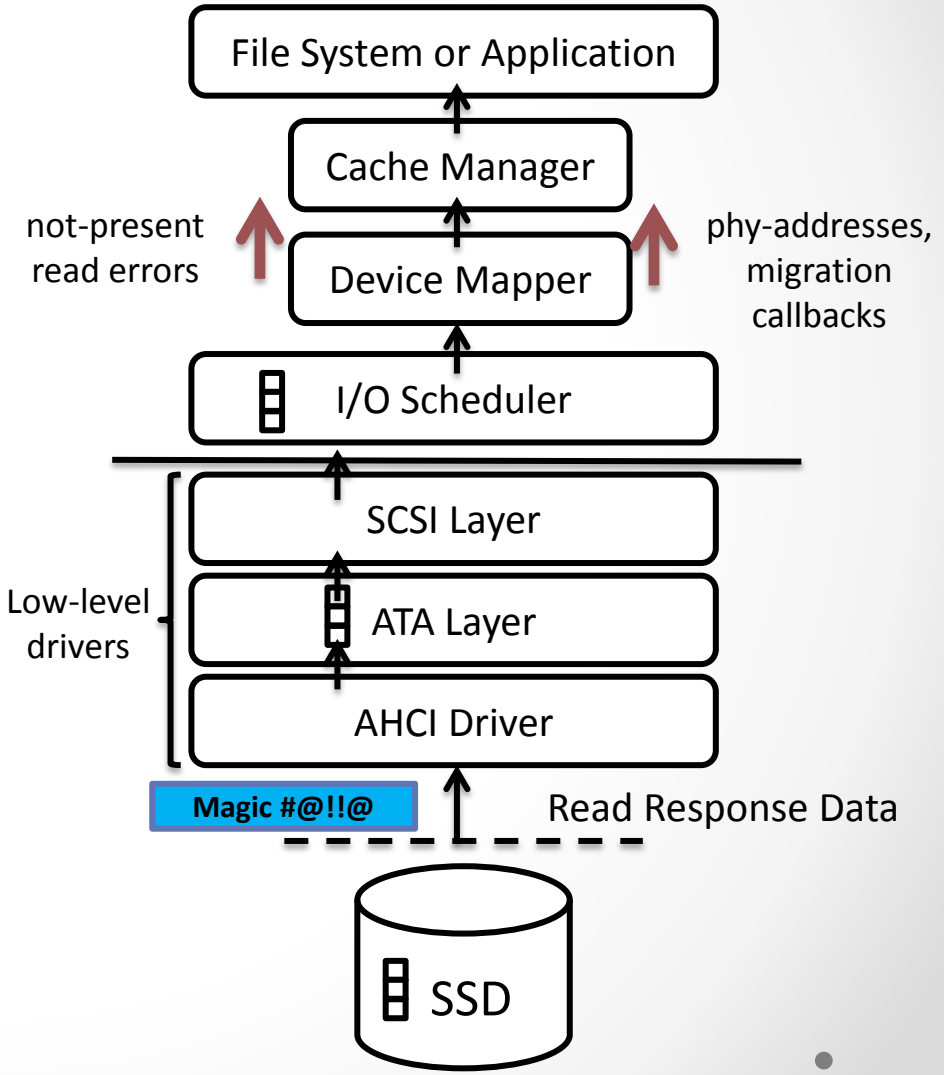
2. New Device Responses

- Reality
  - New errors do not propagate up beyond device drivers
  - Write responses can not encode physical addresses
  - Migration callbacks do not fit standard protocols



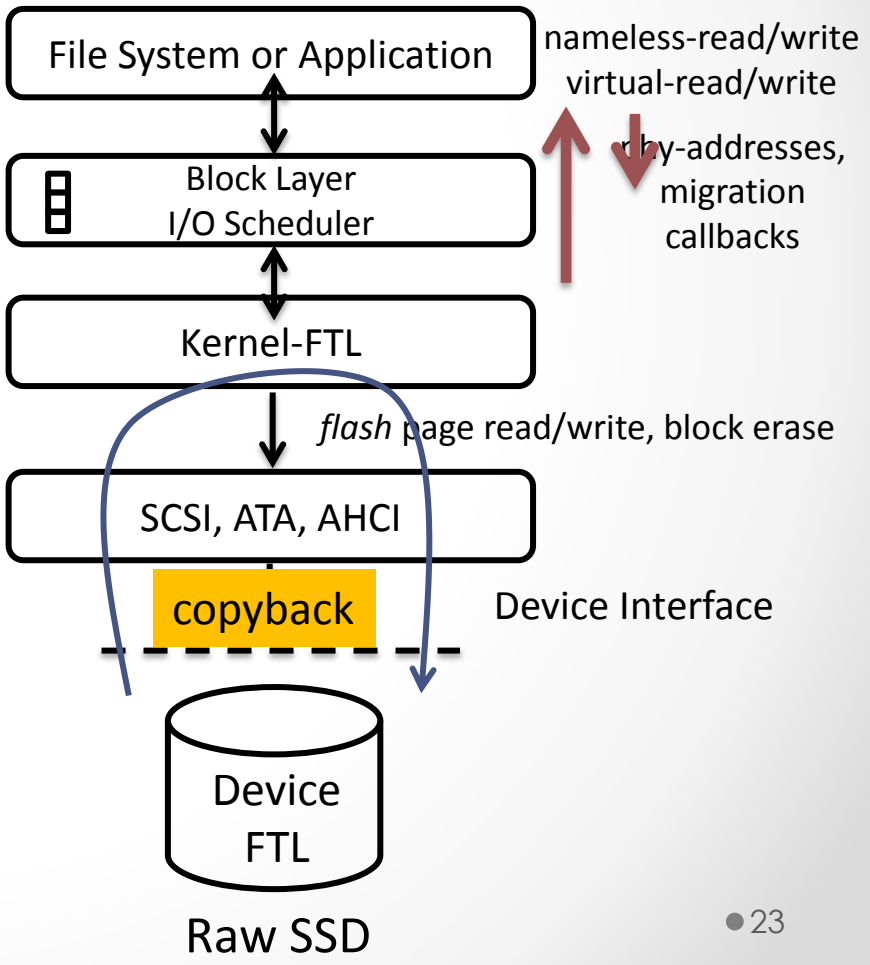
Reverse Path Errors

- Solution
  - not-present errors overloaded on read response data from device to OS



Split-FTL Design

- Solution
  - Kernel and Device FTLs
  - Forward commands transformed to raw flash commands from Kernel-FTL
  - Physical addresses & migration callbacks returned from kernel-FTL to file system
  - Garbage collection: copyback from/to host



Lesson 2
Designing New Device Responses

- Research lesson: consider all OS layers
  - Storage device drivers: handling of frequent benign errors
  - Device & file system: race conditions for callbacks
- Prototyping lesson: simplicity and correctness
  - Kernel-FTL: simpler block layer OS interface
  - Device-FTL: enforce correct erase-before-overwrite

Talk Outline

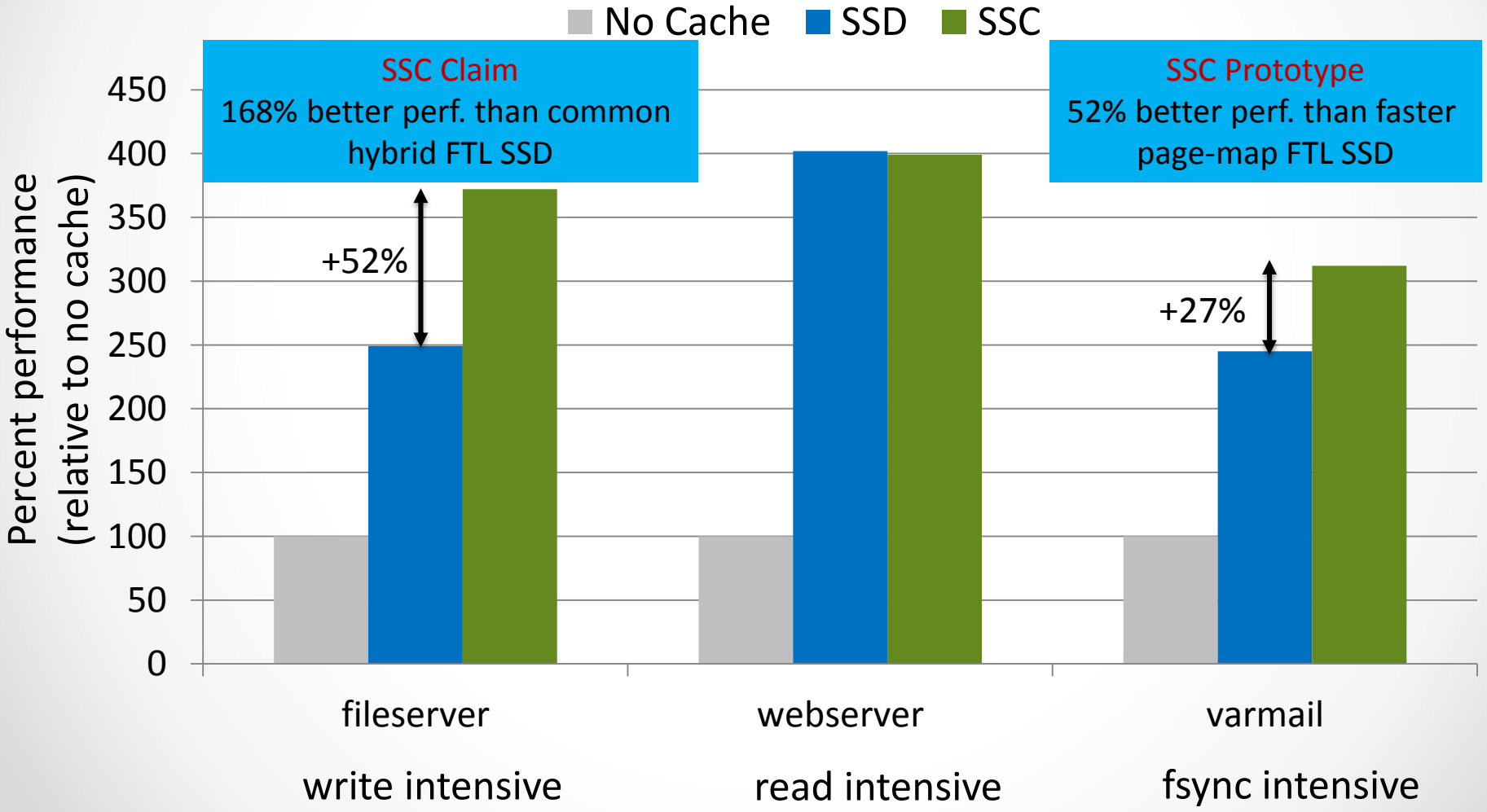
- Introduction
- Background
- Prototyping Experiences
- Evaluation
  - Validate Performance Claims
- Conclusions

Methodology

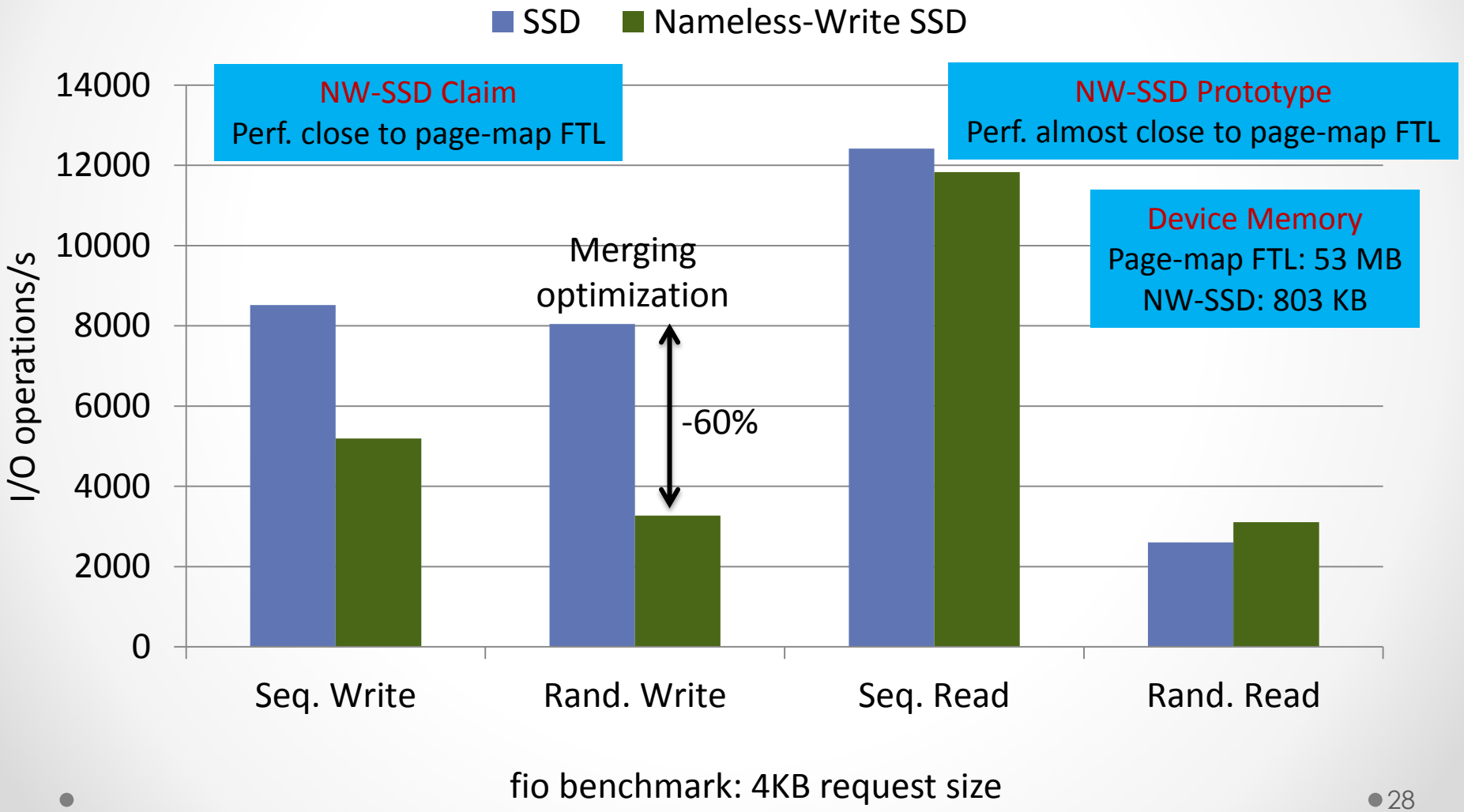
- Systems for comparison
  - SSD: Facebook FlashCache using SSD with GC
  - SSC: modified Facebook FlashCache using SSC with silent eviction
  - Nameless-Write SSD interface performance
- Workload: *filebench* with read/write/fsync

Workload	Ratio of operations
fileserver	1:2 reads/writes
webserver	10:1 reads/writes
varmail	1:1:1 reads/writes/fysnc

SSC Prototype Performance



Nameless-Write SSD Performance



Conclusions

- OpenSSD is a valuable tool for evaluating new SSD designs
- Prototyping and research lessons applicable to other SSD designs
- First high-performance open-source FTL
  - http://www.cs.wisc.edu/~msaxena/new/ftl.html
  - OpenSSD prototype on display at poster session today

Thanks!

Getting Real: Lessons in Transitioning Research Simulations into Hardware Systems

Mohit Saxena, Yiying Zhang
Michael Swift, Andrea Arpaci-Dusseau, Remzi Arpaci-Dusseau