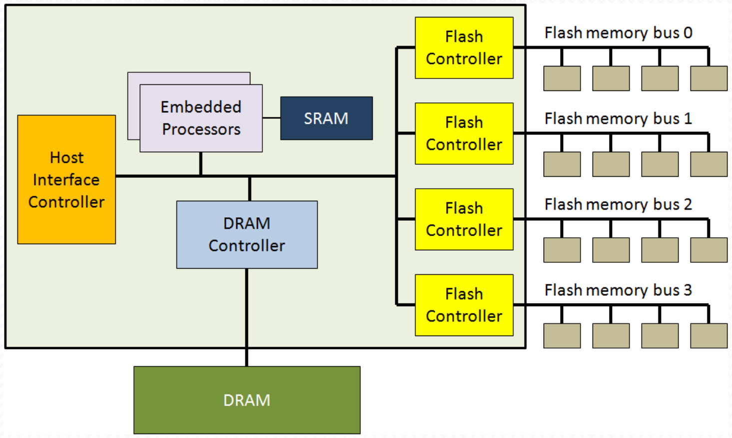
Enabling Cost-effective Data Processing with Smart SSD

Yangwook Kang, UC Santa Cruz
Yang-suk Kee, Samsung Semiconductor
Ethan L. Miller, UC Santa Cruz
Chanik Park, Samsung Electronics

iTRIS Center for Information Technology Research in the Interest of Society
SSRC Storage Systems Research Center
Baskin Engineering UC Santa Cruz

Efficient Use of Solid State Drives

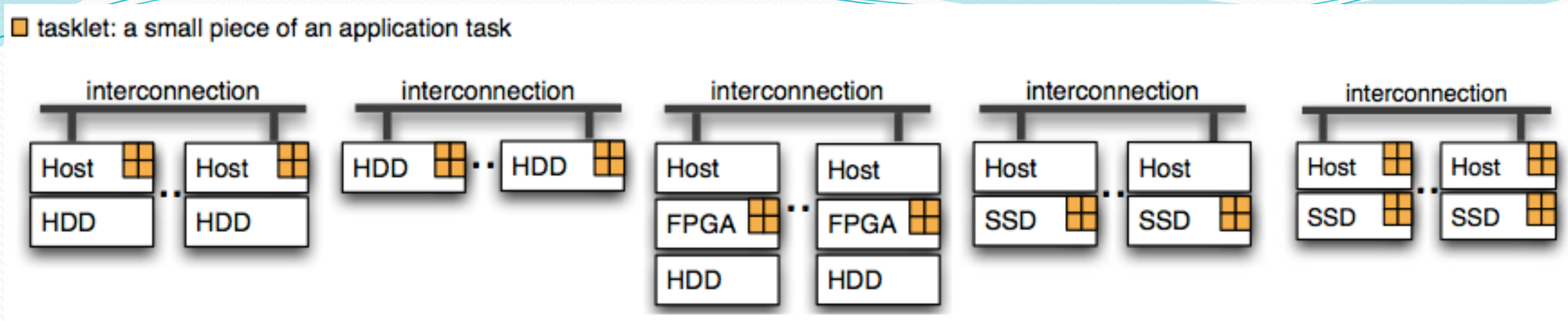
- Modern SSDs
  - Multiple processors / flash controllers
  - High internal bandwidth between Flash and DRAM
  - IOPS: 1k~85k 4K requests



- Assumptions on slow I/O devices are being challenged
  - OS storage stack is optimized for slow I/O devices
  - e.g. I/O schedulers, device queue management, I/O interrupts

- ***How can the system efficiently leverage the processing power and internal bandwidth of SSDs?***
  - *Alternative storage stack for SSDs*
  - *Offloading I/O tasks to SSDs*

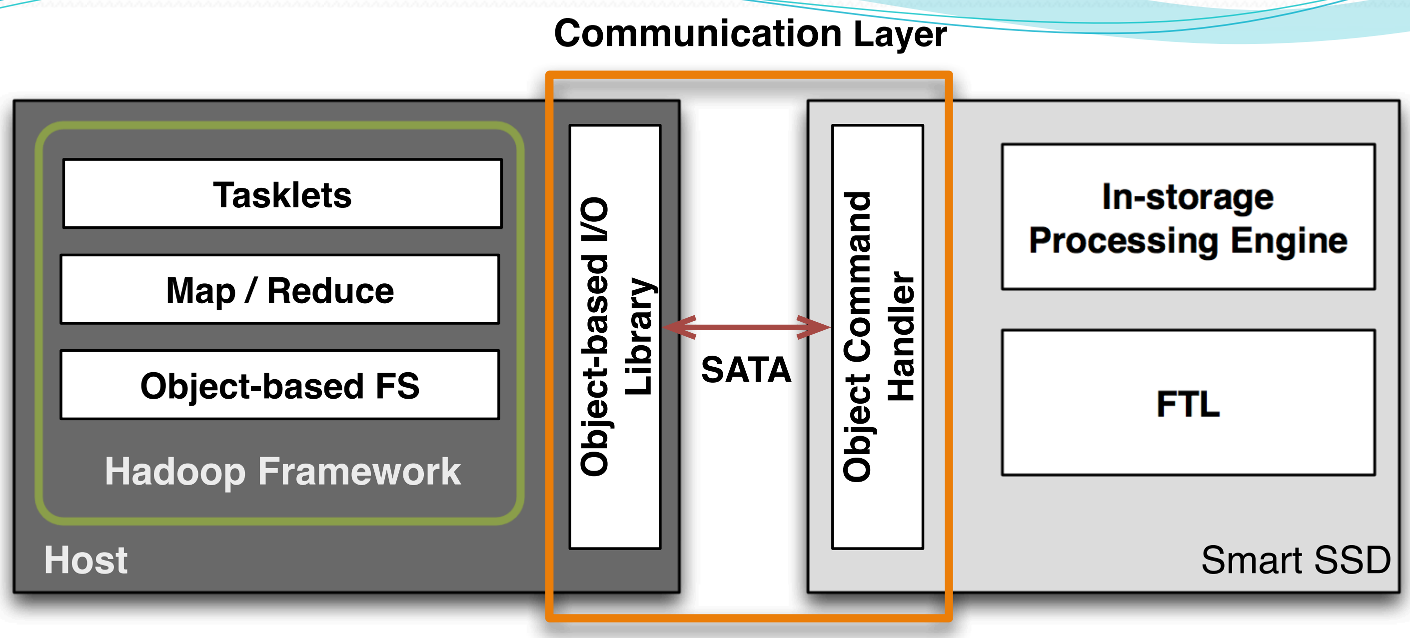
Intelligent Storage Devices



Host-only | Active Disks | FPGA-aided HDDs | Flash controller-based in-storage processing | Smart SSDs

- Problems with HDD-based in-storage processing
  - Limited processing capabilities
  - Requires additional hardware components such as memory and FPGA (commodity hard drives could not be used)
  - Reduces traffic between a host and a device but provides little I/O performance improvements
- Smart SSD
  - Designed to use embedded processors and DRAM in SSDs to process I/O tasks
  - Uses devices' internal knowledge to optimize an execution plan
  - Achieve low energy consumption by not using power-hungry host resources

Smart SSDs



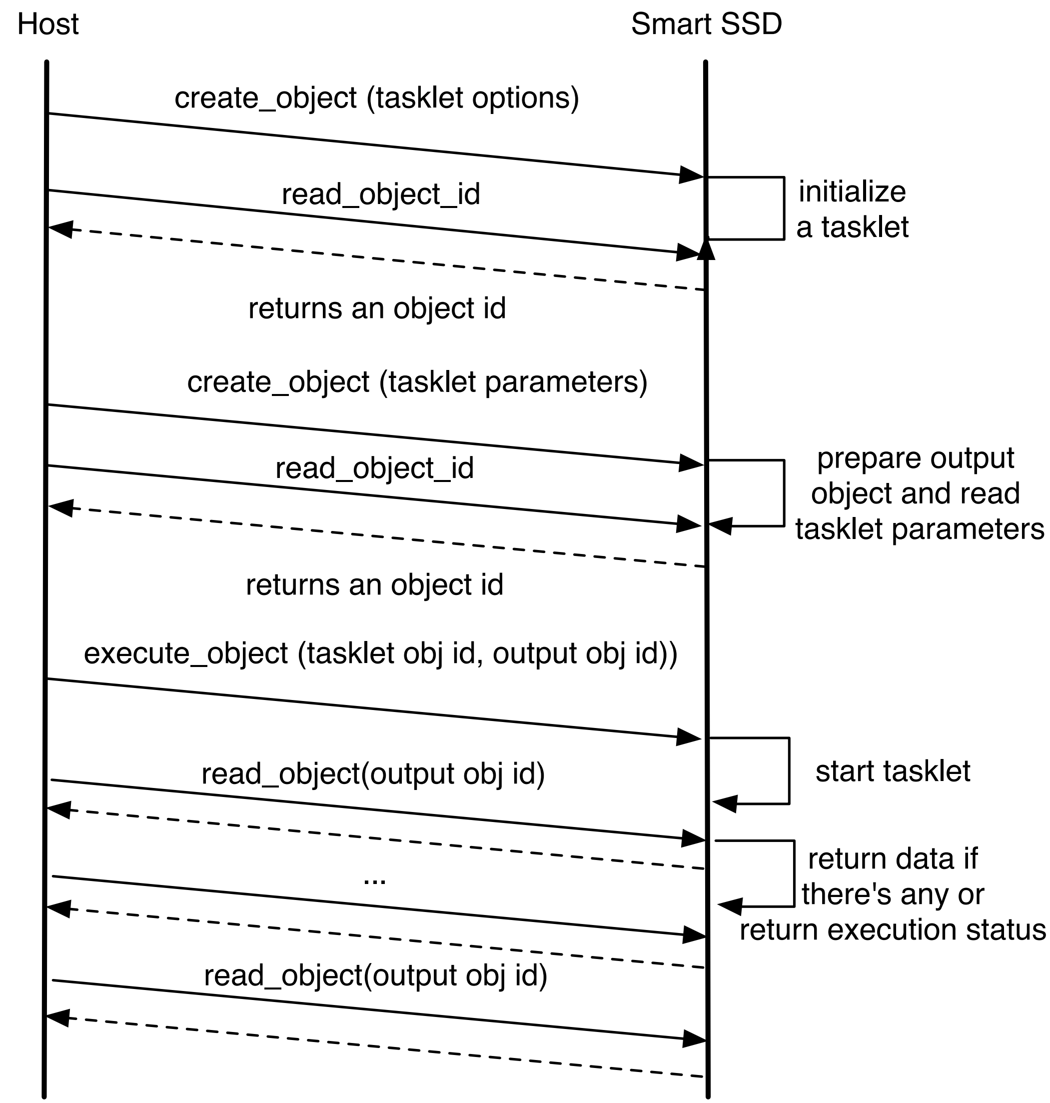
- In-Storage Processing Engine
  - Uses existing hardware and SATA protocols
  - Allows hosts and devices to work together on the same job
- Object-based Communication Layer
  - Designed to provide an universal interface to applications and operating systems, independent to the underlying protocols such as SAS, SATA, and PCI-e
- Application Interface
  - Map/Reduce programming interface for user applications

In-Storage Processing Engine

- An event-driven processing framework to execute tasklets
- Each tasklet implements the following event functions
  - On_Create
  - On_Execute
  - On_DataAvailable
  - On_Read
- Internal read requests:
  - have a lower priority than normal read operations
  - are divided into multiple smaller read requests based on the current load.
  - On_DataAvailable is called whenever each read request is done, to generate partial results

Host-Device Interface
Object Interface on SATA

- Designed to provide an universal APIs regardless of the underlying protocols
- Tasklets are executable objects
- 3 commands are provided
  - create_object
  - execute_object
  - read_object
- Implementation issues
  - No bi-directional communication
  - Device-initiated connection

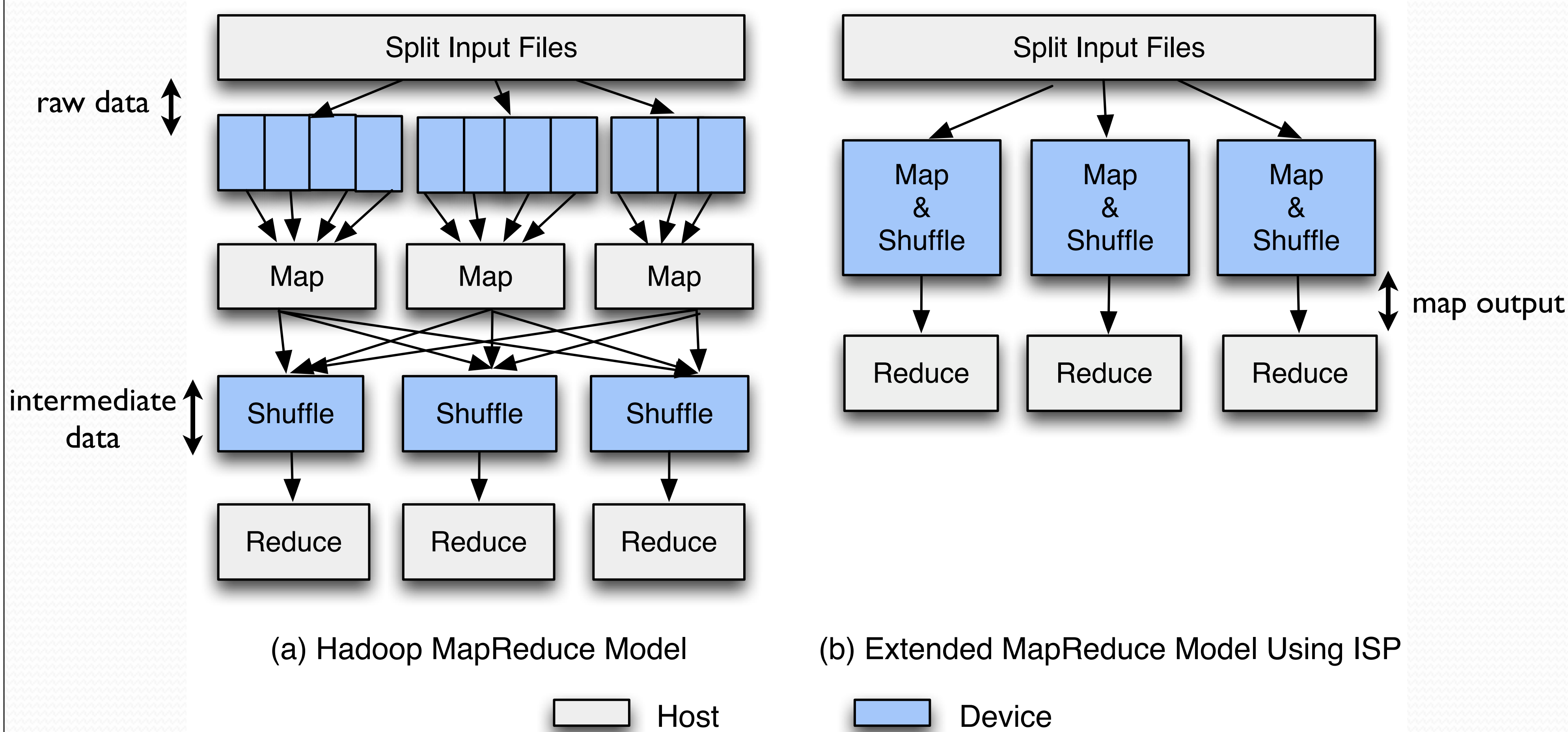


Application Interface for Smart SSD

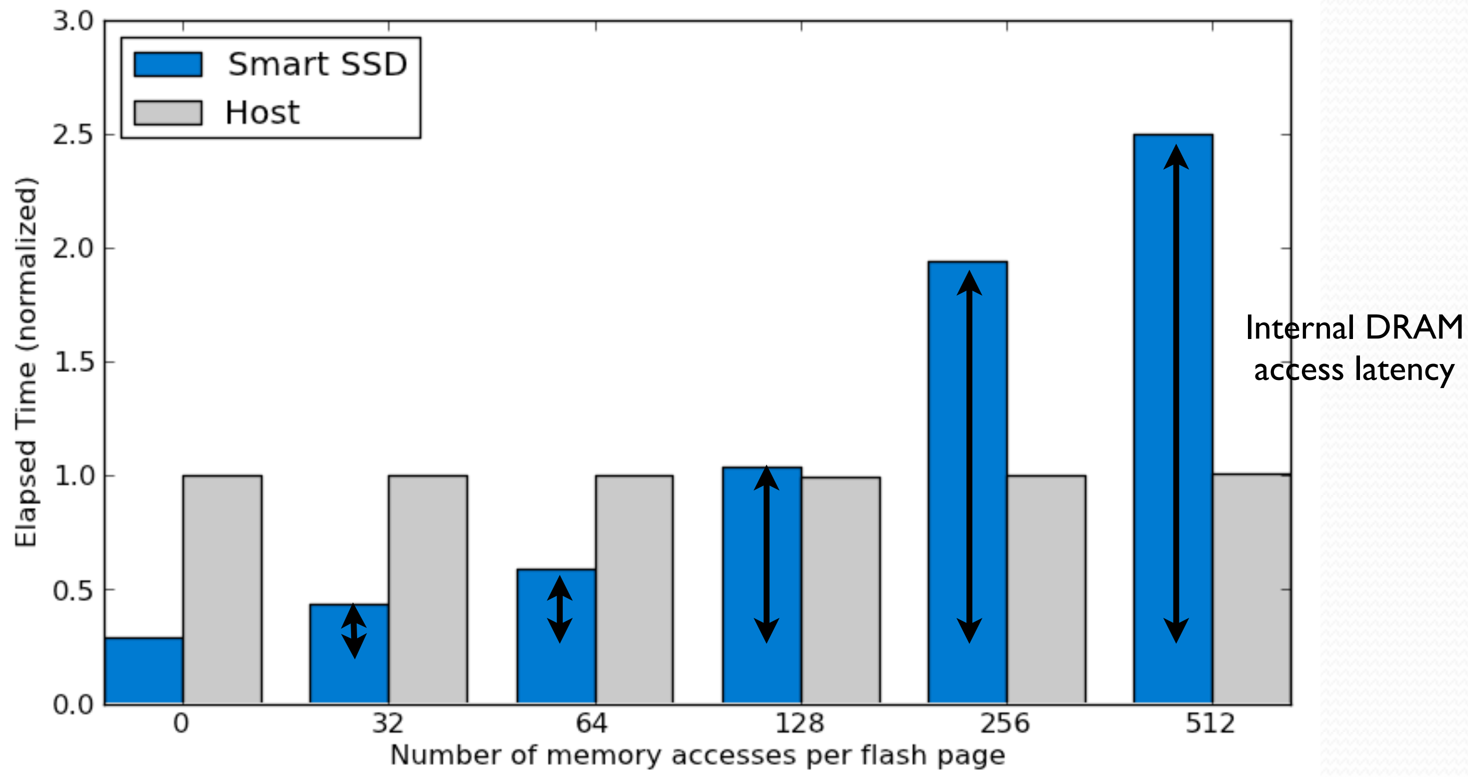
- Direct access
  - Use the low-level protocol directly
  - can be used by SSD-aware storage(file) systems

- MapReduce Model
  - Simple programming interface to applications
  - Hide the detailed communication from applications
  - Generate independent sets of data that can be assigned to multiple Smart SSDs / tasklets concurrently
  - Tasklets can be mappers, reducers or both

Extended MapReduce Model for In-storage Processing

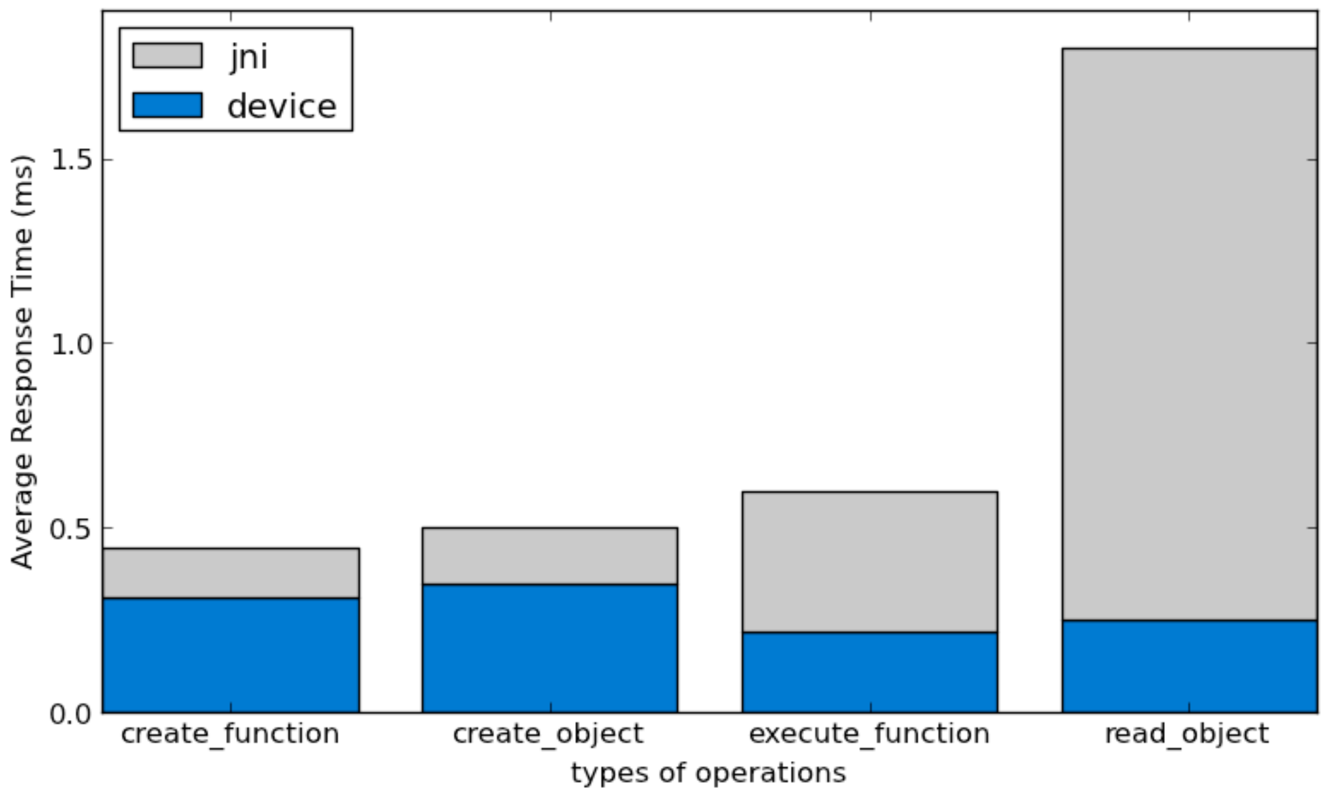


Evaluation
Tasklet Execution Performance



- Internal read performance - *1.6x faster* than the host-device bandwidth
- For each access, it reads one integer, and does one integer comparison
- However, the performance degrades as the number of DRAM accesses and comparisons increase

Evaluation
Host-Device Communication Overhead



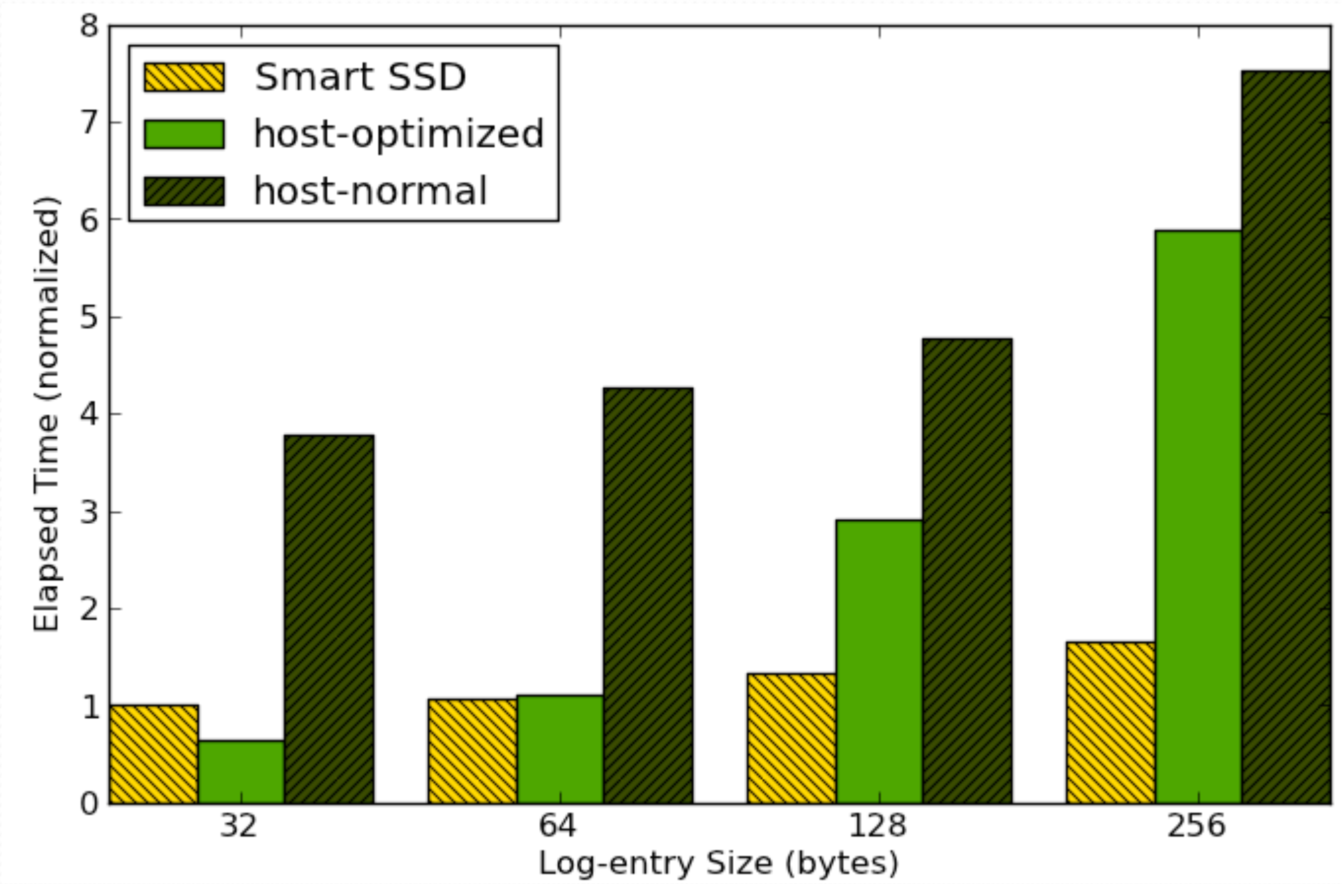
- create_function, create_output_object takes more time than other functions, because it consists of two separate commands
- read_object takes more time in JNI part, due to the memory copy between c and java and algorithms for determining the polling interval

Evaluation Log-Analysis Example

- Trace
  - 1998 World Cup website access log
  - 7,000,000 entries with different sizes ranging from 32 bytes to 256 bytes
- Scenarios
  - Number of accesses per region (requires 4 byte read per log entry and sum)
  - Top 5 file types accessed per region (requires 8 byte read per log entry and sorting)
- Applications
  - host-normal: normal log-analyzer using MapReduce
  - host-optimized: modified version of host-normal, not generating the intermediate results
  - Smart SSD: in-storage processing

Evaluation
Log-Analysis Example

- Scenario 1. counting the number of accesses per region

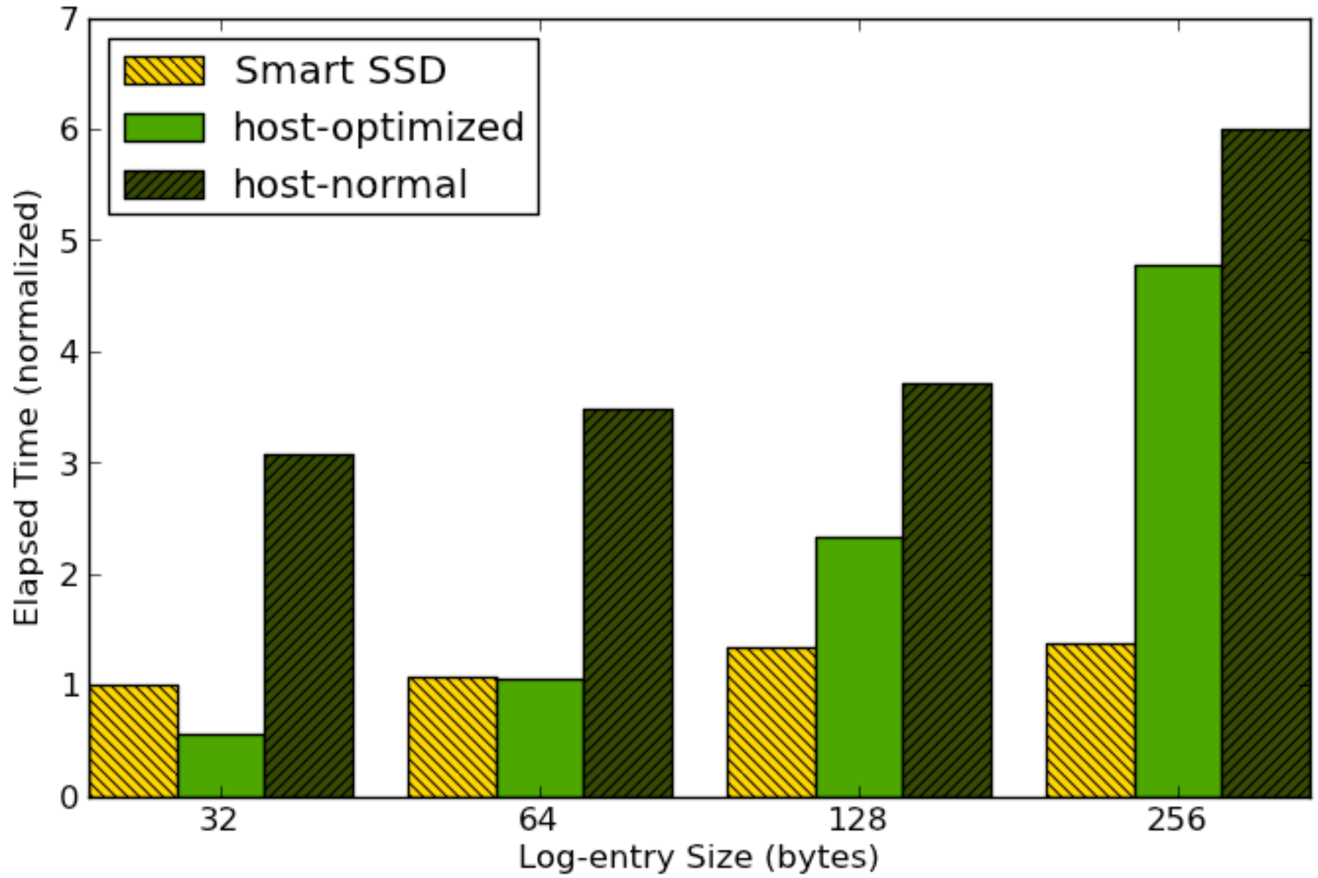


Baskin Engineering UC SANTA CRUZ

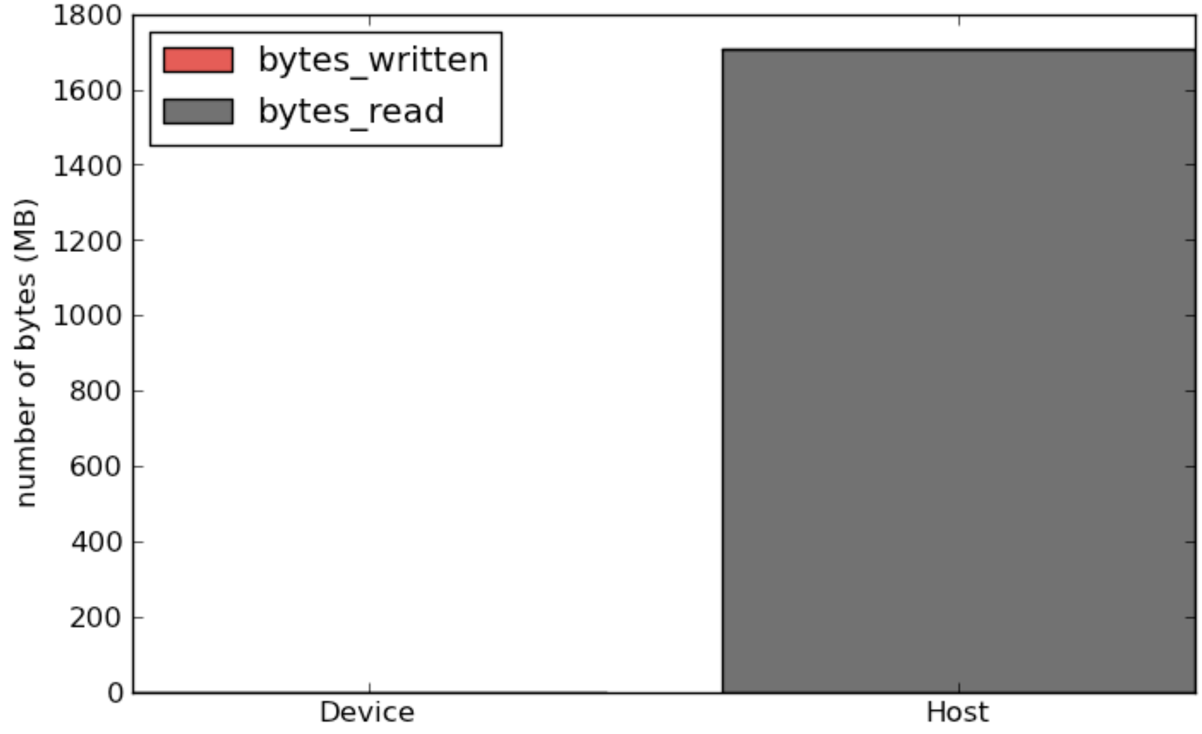
ssrc

Evaluation
Log-Analysis Example

- Scenario 2. top 5 file types accessed per region

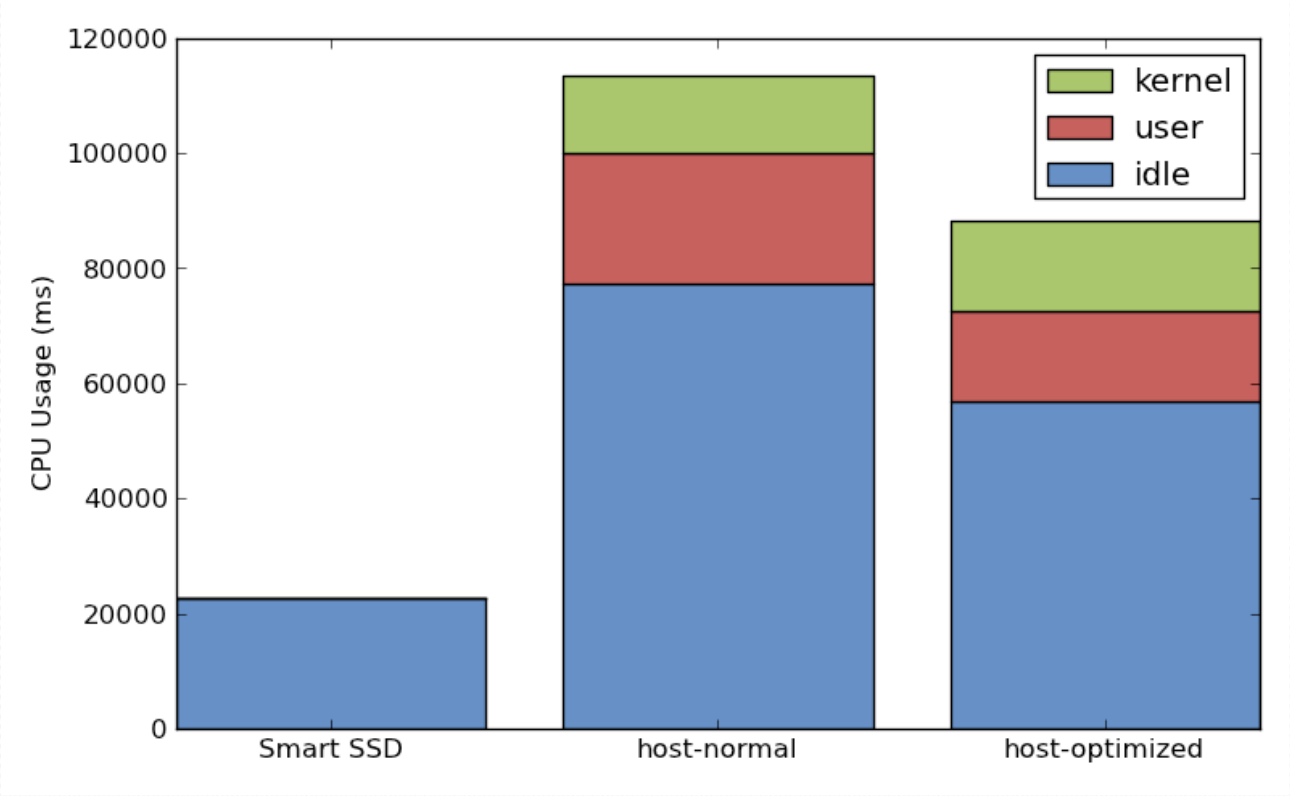


Evaluation
Disk I/Os



- Smart SSDs do not transfer any raw data to the host

Evaluation
CPU Usage

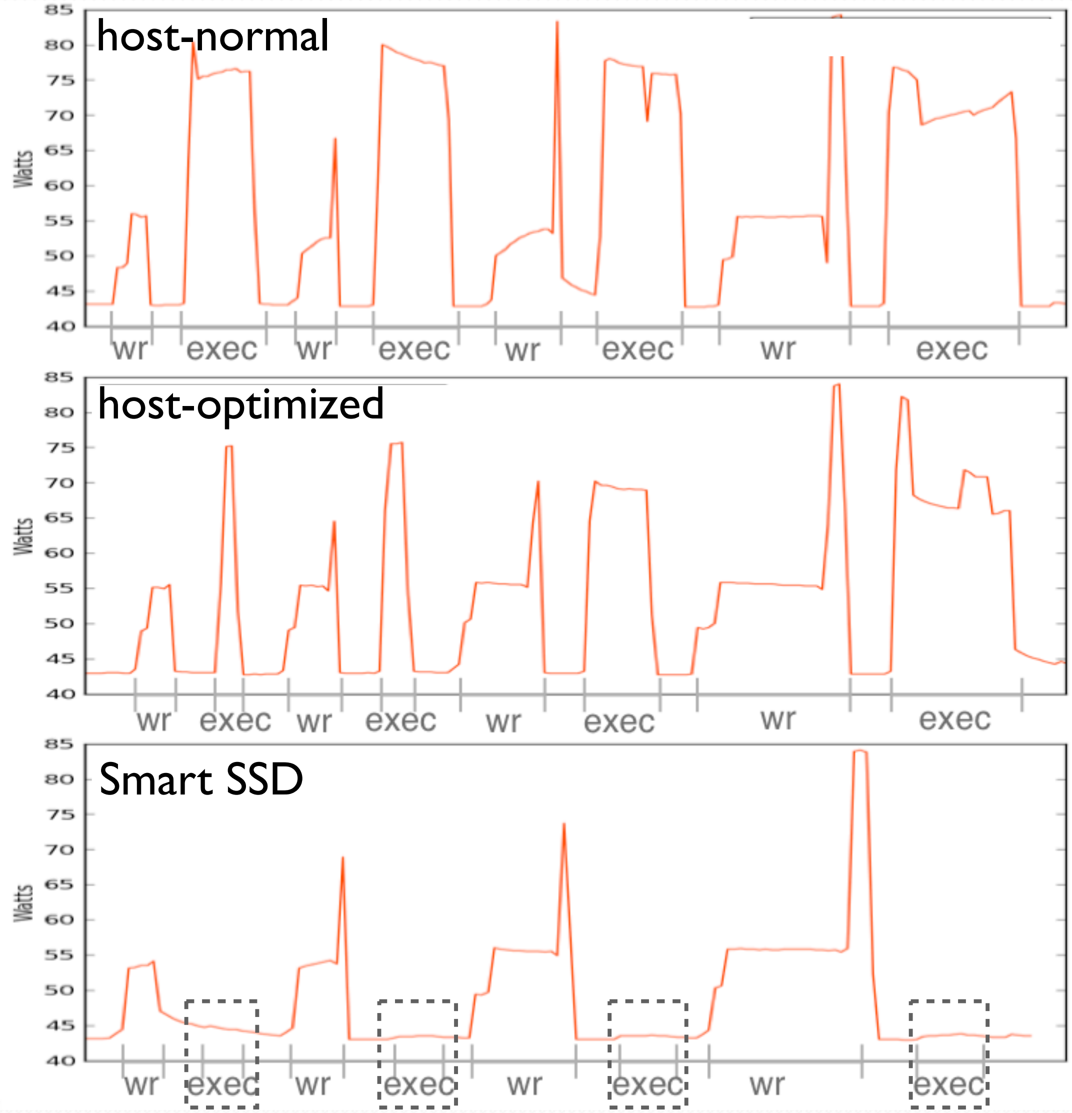


- With Smart SSDs, CPUs are mostly in the idle state waiting for the results
- In host-normal and host-optimized, CPUs wait for I/O requests (kernel time), and then process the data (user time)

Evaluation
Energy Efficiency

- Host-normal and host-optimized:
  - host system uses 35 ~ 40 Watts
- Smart SSD:
  - consumes 0.9 Watts to 1.2 Watts

50% reduction in overall energy consumption



Baskin Engineering UC SANTA CRUZ

ssrc

Evaluation
Data Filtering

- Problem of long DRAM read latencies can be alleviated by using filtering instead of searching
  - Stop processing after the first match
- Host systems can assign data filtering tasks to the device before entering long computations
  - taking advantage of energy-efficient ISP
- Data filtering with 1GB data, 60% selectivity, 256 accesses per page => 40% faster than searching

Suggestions for Future Smart SSD Architecture

- Reduce DRAM Access Latency
  - High bandwidth between DRAM and CPU
  - L1/L2 caches for embedded processors

- I/O Latency
  - Use application processors to enable background processing

- Tasklet Programming
  - Define a general set of APIs that expose the functionality of the firmware and hardware to tasklets
  - Add support for a script language and Sandboxing

Summary

- Explored the potentials and limitations of the Smart SSD model on the current SSD architecture
- Based on a prototype on a real SSD hardware and firmware, we measured performance and energy consumption of a Smart SSD
- Smart SSDs can help achieve both high performance and energy efficiency at a low cost through in-storage data processing
- Problems with the current SSD architecture
  - High DRAM access latency
  - Lack of cpu caches
  - No dynamic memory allocations

Thank you

Questions?

Extended Hadoop Framework

- Object File System
  - A Hadoop file system that supports Smart SSDs
    - Provides open/read/write to a raw Smart SSD
    - Directory support
  - ISP Support
    - Internally handles object-based communication with a device
  - Block management
    - Manages logical block numbers (unlike pure Object-based file systems)
- MapReduce Framework
  - ‘DeviceMapper’ and ‘DeviceJobClient’ are added