Internal Parallelism of Flash Memory-Based Solid-State Drives

Presented by: Nafiseh Mahmoudi

Spring 2017

BIG Data Management and Mining Laboratory
UNIVERSITY OF COLORADO DENVER | ANSCHUTZ MEDICAL CAMPUS

Paper Information

Authors:

Feng Chen Louisiana State University, Baton Rouge, LA
Binbing Hou Louisiana State University, Baton Rouge, LA
Rubao Lee Ohio State University, Columbus, OH

Publication:

- ACM Transactions on Storage (TOS), 2016

Type:

- Research Paper

- High speed data processing demands high storage I/O performance.
- Flash memory based SSD, with no moving part have received strong interest.
- Internal parallelism is a unique and rich resource embedded inside SSDs.

Internal parallelism in SSD

- SSDs have several inherent architectural limitations:
  1. one single flash memory package can only provide limited bandwidth (e.g., 32 to 40MB/sec), which is unsatisfactory for practical use as a high-performance storage.
  2. Writes in flash memory are often much slower than reads.
  3. FTL running at the device firmware level needs to perform several critical operations.
- These internal operations can incur a high overhead.

- High parallelizing design yields 2 benefits:
  1. Transferring data in parallel can provide high bandwidth
  2. High latency operations can be hidden behind other concurrent operations.

CONTENTS

1. Critical issues for investigation
2. **The background**
3. introduces our experimental system and methodology
4. Detect the SSD internals
5. Present our experimental and case studies on SSDs.
6. The interaction with external parallelism

Critical issues for investigation

- Strive to answer several question

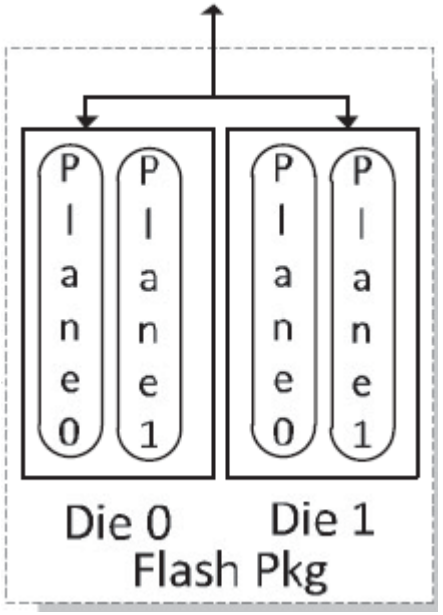
1. To fully understand internal parallelism of SSDs
2. Gain insight into unique resource of SSDs
3. Reveal some untold facts and unexpected dynamics in SSDs

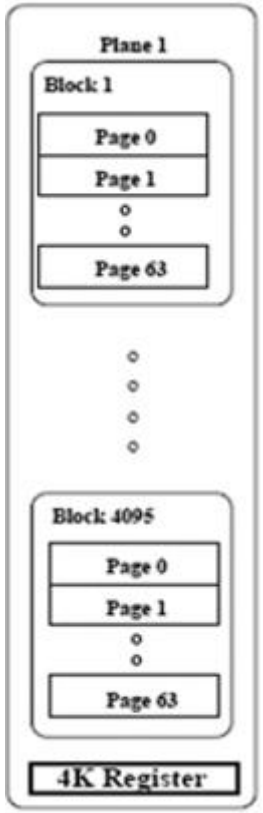
BACKGROUND OF SSD ARCHITECTURE

1. NAND Flash Memory
2. Flash Translation Layer (FTL)
3. SSD Architecture and Internal Parallelism
4. Command Queuing

1. NAND Flash Memory

- Nand flash classified into :
  1. SLC: stores only on bit
  2. MLC: can store 2 bits or even more in one cell
  3. TLC: stores 3 bits
- Nand flash memory package composed of :
- Each page has :
  - A data part (4-8KB)
  - Associated metadata area (128 bytes)
    - ✓ Used for storing ECC
- Flash memory has 3 major operation :
  - Read, write performed in units of pages
  - Erase





2. Flash Translation Layer (FTL)

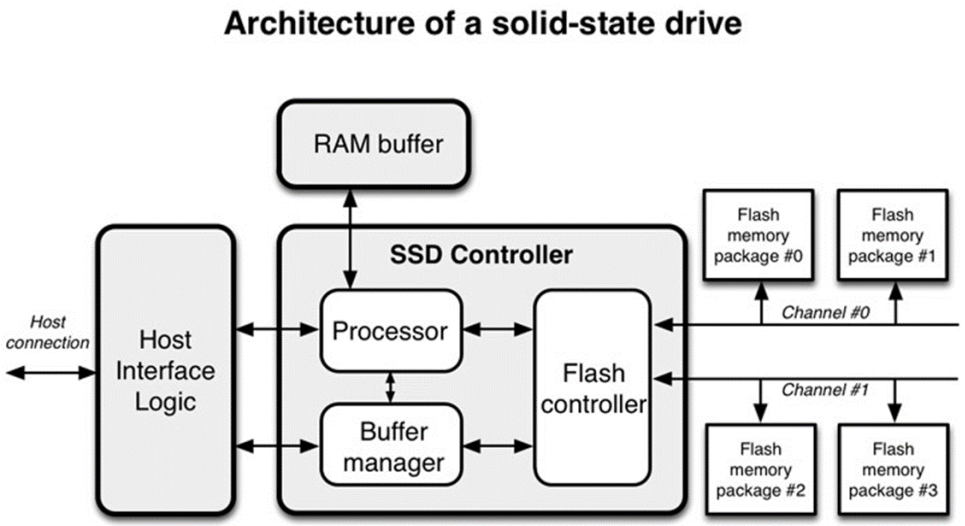
- FTL is implemented to emulate a hard disk.

- Three key roles of FTL:
  1. logical block mapping: maps LBA to PBA
  2. garbage collection: handles the no-in-place-write issue
  3. wear leveling: shuffles cold blocks with hot blocks

3. SSD Architecture and Internal Parallelism

- SSD includes four major components:
  - 1. host interface logic 2.Controller 3.Ram 4.Flash memory controller

- Different levels of parallelism:
  - Channel-Level Parallelism
  - Package-Level Parallelism
  - Die level parallelism
  - Plane level parallelism



4. Command Queuing

- Native command queuing(NCQ): a feature introduced by SATA II
- The device can accept multiple incoming commands and schedule the jobs internally
- Enables SSDs to achieve real parallel data access internally

1. Critical issues for investigation
2. Background of SSD architecture
- **Introduces our experimental system and methodology**

1. Detect the SSD internals
2. present our experimental and case studies on SSDs.
3. The interaction with external parallelism

MEASUREMENT ENVIRONMENT

1. Experimental Systems
2. Experimental Tools and Workloads
3. Trace Collection

1. Experimental Systems

- experimental studies have been conducted on a Dell Precision T3400. It is equipped with an Intel Core 2Duo E7300 2.66GHz processor and 4GB main memory. A 250GB Seagate 7200RPM hard drive is used for maintaining the OS and home directories . Use Fedora Core 9 with the Linux Kernel 2.6.27 and Ext3 file system.
- Select two representative, the two SSDs target different markets
  - One is built on MLC flash memories and designed for the mass market
  - The other is a high-end product built on faster and more durable SLC

Table I. SSD Specification

	SSD-M	SSD-S
Capacity	80GB	32GB
NCQ	32	32
Interface	SATA	SATA
Flash memory	MLC	SLC
Page Size (KB)	4	4
Block Size (KB)	512	256
Read Latency (μs)	50	25
Write Latency (μs)	900	250
Erase Latency (μs)	3,500	700

2. Experimental Tools and Workloads

- For studying the internal parallelism of SSDs
  - Use two tools (Intel Open Storage Toolkit, custom-built tool), to generate workloads with the following three access patterns.

1. Sequential: using a specified request size, starting from sector 0.
2. Random: Blocks are randomly selected from the first 1,024MB of the storage
3. Stride: starting from sector 0 with a stride distance 2 consecutive data access

- ✓ Each workloads runs for 30 seconds in default to limit trace size while collecting sufficient data.
- ✓ All workloads directly access the SSDs as raw block devices

3. Trace collection

- In order to analyze I/O traffic, use:
  - Blktrace to trace the I/O activity
- The tool intercepts I/O events in the OS kernel such as
  - Queuing
  - Dispatching
  - Completion
- Completion event: Report the latency for each individual request
- The trace data are first collected in memory and then copied in HDD
  - To minimize the interference caused by tracing

1. Critical issues for investigation
2. Background of SSD architecture
3. Introduces our experimental system and methodology
4. **Detect the SSD internals**
5. present our experimental and case studies on SSDs.
6. The interaction with external parallelism

4. UNCOVERING SSD INTERNALS

1. Internal parallelism is an architecture-dependent resource.

2. Present a set of experimental approaches to expose the SSD internals.

1. A Generalized Model
2. Chunk Size
3. Interleaving Degree
4. The Mapping Policy

- Domain: is a set of flash memories that share a specific set of resources. (e.g. channels).
  - A domain can be further partitioned into sub-domains (e.g. packages)
- Chunk: A chunk is a unit of data that is continuously allocated within on domain.
  - Chunks are interleavingly placed over set of N domains by following a mapping policy.
  - Chunk size: the size of the largest unit of data that is continuously mapped within individual domain.
- Stripe: A set of chunks across each of N domains are called stirpe

- A general goal is to minimize resource sharing and maximize resource utilization, three key factors directly determine the internal parallelism.

I. Chunk size: the size of the largest unit of data that is continuously mapped within an individual domain.

II. Interleaving degree: the number of domains at the same level. The interleaving degree is essentially determined by the resource redundancy (e.g., channels).

III. Mapping policy: the method that determines the domain to which a chunk of logical data is mapped. This policy determines the physical data layout.

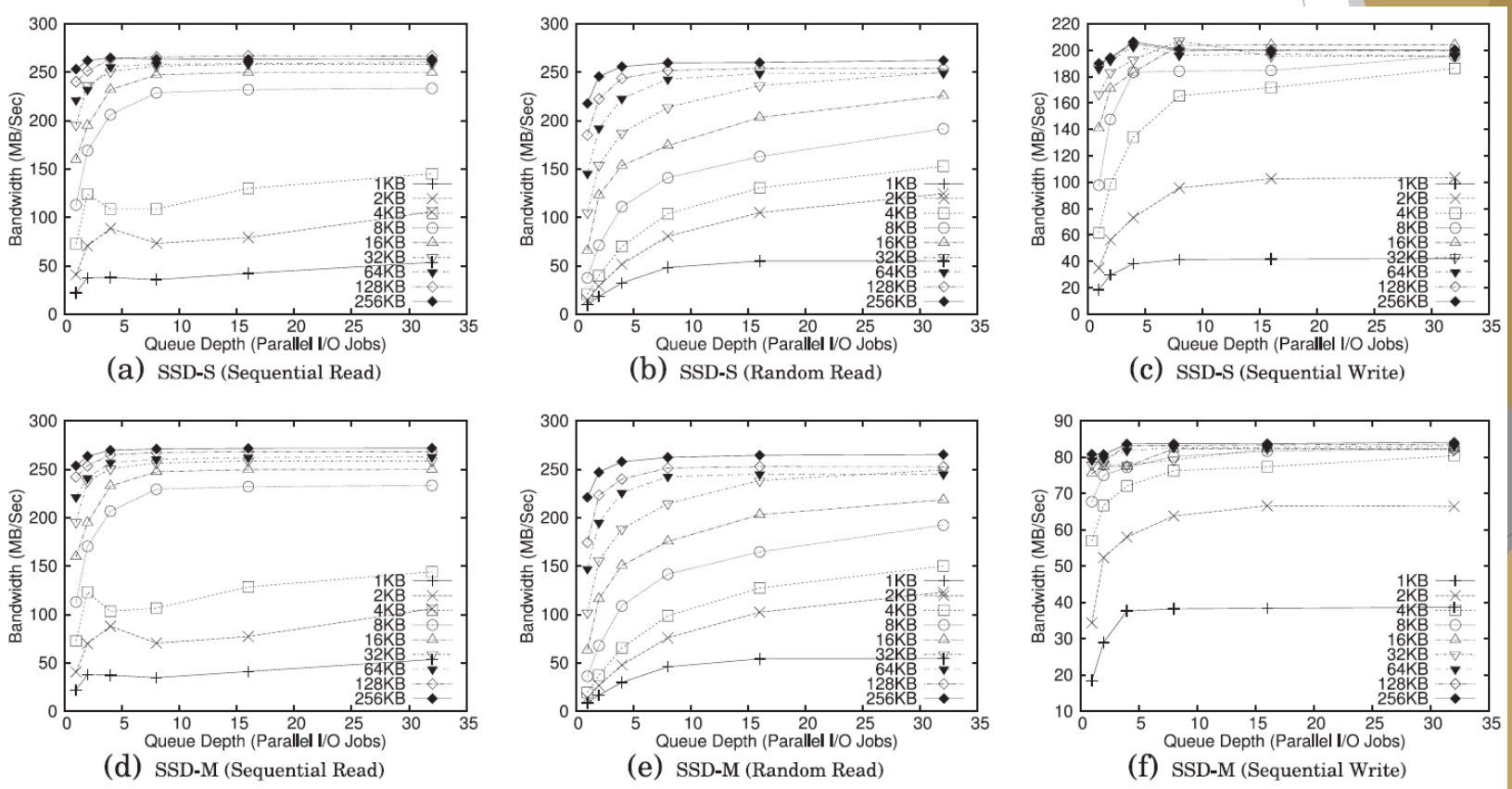
1. Critical issues for investigation
2. Background of SSD architecture
3. Introduces our experimental system and methodology
4. Detect the SSD internals
5. **present our experimental and case studies on SSDs.**
6. The interaction with external parallelism

PERFORMANCE STUDIES

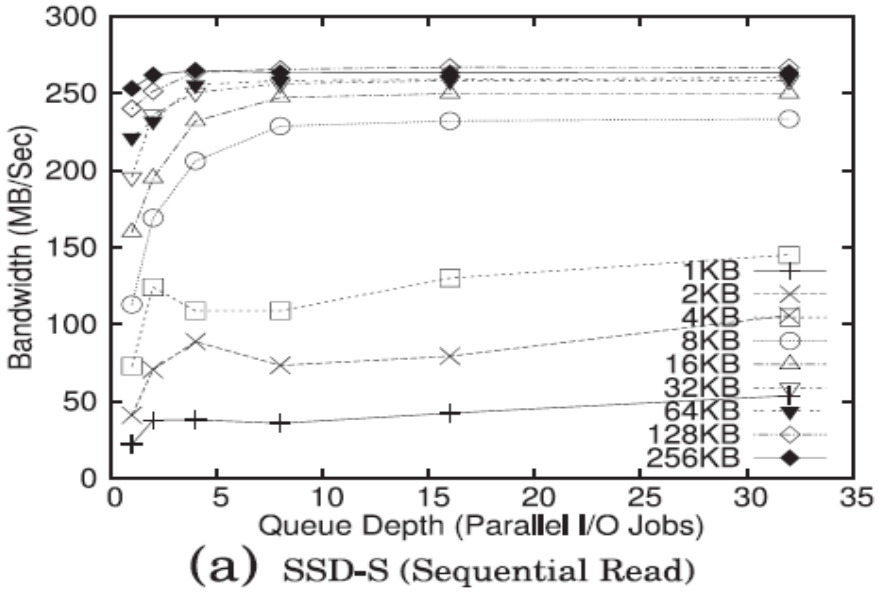
1. What Are the Benefits of Parallelism?
2. How Do Parallel Reads and Writes Interfere with Each Other and Cause Performance Degradation?
3. How Does I/O Parallelism Impact the Effectiveness of Readahead?
4. How Does an Ill-Mapped Data Layout Impact I/O Parallelism?
5. Readahead on Ill-Mapped Data Layout

1. Benefit of I/O parallelism in SSD

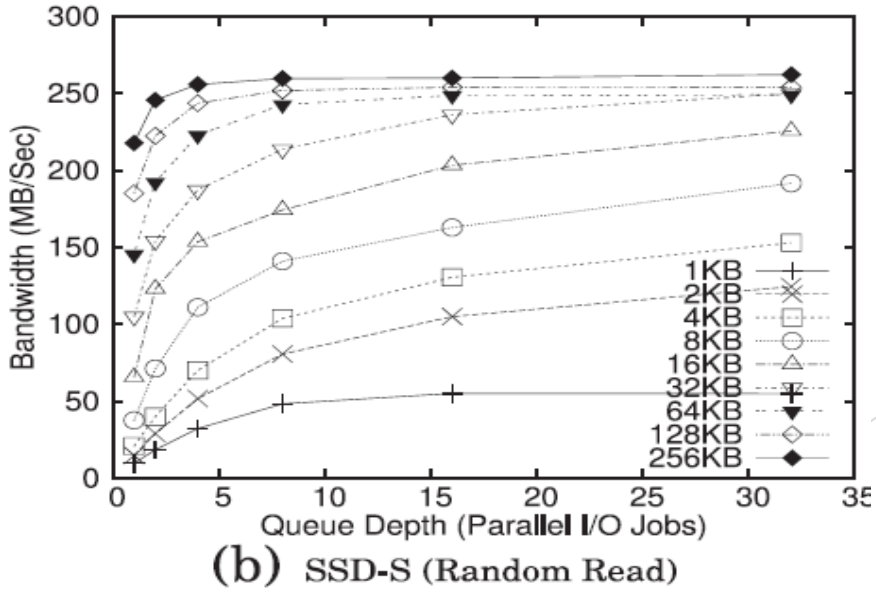
- Run four workloads with different access patterns
  - Sequential read
  - Sequential write
  - Random read
  - Random write



- Small and random reads yield the most significant performance gains.
- A large request benefits less from parallelism
  - because the continuous logical blocks are often striped across domains and it already benefits from internal parallelism
- In order to exploit effectively internal parallelism, we can either increase request sizes or parallelize small requests
- Highly parallelized small random access can achieve performance comparable to large sequential access without parallelism
- Writes on SSD-M, quickly reach the peak bandwidth
- Less difference on the two SSDs for reads.



(a) SSD-S (Sequential Read)



(b) SSD-S (Random Read)

- The performance potential of increasing I/O parallelism is physically limited by the redundancy of available resources
  - when the queue depth reaches over 8–10 jobs, further increasing parallelism receives diminishing benefits.
  - When the queue depth exceeds 10, a channel has to be shared by more than 1 job.
  - Over-parallelizing does not result in undesirable negative effects
- The performance benefits of parallelizing I/O also depends on the flash memory mediums
  - MLC and SLC flash memories provide performance difference, especially for writes.
  - MLC-based lower-end SSD, quickly reach the peak bandwidth (only about 80MB/sec)
  - the SLC-based higher-end SSD, shows much higher peak bandwidth (around 200MB/sec) and more headroom for parallelizing small writes.

Parallel Reads and Writes Interfere

- Both operations share many critical resources (ECC)
  - ✓ Parallel jobs accessing such resources need to be serialized
- Reads and writes have a strong interference with each other
- The significance of this interference highly depends on read access patterns

I/O parallelism impact on effectiveness of readahead

- **sequential** reads on SSD can outperform random reads with no parallelism
  - because modern SSDs implement a readahead mechanism at the device level to detect sequential data accesses and prefetch data into the on-device buffer
- parallelizing multiple sequential read streams could result in a sequence of mingled reads, which can affect the detection of sequential patterns and invalidate readahead.
- There is a tradeoff between increasing parallelism and retaining effective readahead.

Thank you