TRIM: Behavior of Subsequent READs

David L. Black, EMC
Fred Knight, NetApp

September 2008
T10/08-347r0

Topic: Thin Provisioning and TRIM

- Thin Provisioning
  - Some logical blocks have no corresponding physical blocks.
  - Physical blocks assigned as I/O occurs, typically on write
- Proposed TRIM command (T10/08-149r1)
  - Data in affected blocks no longer needed
  - Device server **may** reassign the underlying physical blocks.
- Question #1: What happens when TRIM'ed blocks are READ?
  - Assume that device server reassigned underlying physical blocks.
  - Assume that the READ does not assign physical blocks.
- Question #2: What happens when a TRIM'ed blocks is READ multiple times without any WRITE or other operation that sets its value?
- T10/08-149r1 has good answers to these questions:
  - READ after TRIM: unchanged data or a constant value
  - READ operations always return consistent data.

Problem: The ATA TRIM command

- Question #1: What happens when TRIM'ed blocks are READ?
  - ATA TRIM Answer #1: The result is "indeterminate". It may be data that was previously written somewhere else on the drive.
  - **This is a potential security problem.**
- Question #2: What happens when a TRIM'ed block is READ multiple times without any WRITE or other operation that changes its value?
  - ATA TRIM Answer #2: The READ operations may return different values for the same block.
  - **This is a SERIOUS data corruption problem!!**

- Apparent Rationale: ATA SSD drive behavior
  - Never-written blocks are mapped/remapped to physical blocks
  - READ returns whatever's in the currently mapped physical block
- These drives are broken
- SCSI standards (e.g., SAT) must not allow this misbehavior

(#1) Security and Drive Partitioning

- Question #1: What happens when TRIM'ed blocks are READ?
  - ATA TRIM Answer #1: The value is "indeterminate".
- Drives often partitioned by higher layer functionality
  - Examples: Disk array, hypervisor I/O stack, volume manager
  - Data isolation often expected (or required) across partitions
- Security Problem: TRIM pass-through to drive
  - Alice and Mallory share a partitioned drive
  - Alice uses her drive for sensitive data that Mallory would like to see
  - Partitioning layer prevents this, but not when TRIM is passed through
- Mallory's Attack:
  - Mallory TRIMs some of his partition, TRIM is passed to drive.
  - Mallory READs his TRIM'ed blocks looking for interesting data.
  - Alice overwrites some data (**Alice doesn't have to use TRIM!**).
  - Some of Alice's overwritten data may be returned to Mallory. (ouch!)

(#2) Stored Data Corruption

- Question #2: What happens when a TRIM'ed block is READ multiple times without any WRITE or other operation that changes its value?
  - ATA TRIM Answer #2: The READ operations may return different values for the same block.

- This is dangerous!
  - Can cause data corruption for important uses of storage
- Three examples (there are more):
  1. Filesystem repair (e.g., fsck, CHKDSK)
  2. Mirroring (e.g., RAID 1)
  3. Single parity-based RAID (e.g., RAID 3, 4, 5)
- Each has possible event sequences that corrupt stored data
  - Failures involved (e.g., OS crash, drive failure)
  - Recovery assumes data does not change between reads when no writes occur between the reads

Data Corruption: Filesystem (FS) repair

- Every open systems filesystem has a repair application
  - Examples: fsck, CHKDSK
- Sooner or later something goes wrong (e.g., crash)
  - Leaves FS stored data structures in an inconsistent state.
  - Journalled filesystems deal with common failure scenarios
  - That leaves uncommon failures
- Eventually Murphy’s Law strikes and FS repair is needed
  - Unexpected stored metadata changes: FS repair can corrupt data

Data Corruption: Filesystem (FS) repair

- Corruption example based on FS indirect blocks
  - Indirect Block: holds set of pointers to actual data blocks in FS
- Data corruption sequence: Add an indirect block
  1. Write to initialize indirect block fails.
     - The returned ASC is not understood.
  2. OS reads the indirect block to check value – it's ok.
     - Drive supplies an old initialized indirect block.
  3. Indirect block integrated into FS data structures
  4. Operating system crash, run FS repair application.
  5. FS repair application reads the indirect block.
     - **Data change**: drive supplies a different old indirect block that contains block pointers.
  6. Result: data inserted into corresponding file (oops)

Data Corruption: Mirroring

- Two-way mirrored drive pair (RAID 1)
  - Each write done to both drives, completes when both drives respond
  - Intent log maintained to record outstanding incomplete operations
- Consider data value that exists in unused physical blocks.
  - All zeroes for this example, but actual value doesn't matter.
- Data Corruption sequence: Write zeroes to a mirrored block.
  1. Mirror logic writes zeroes to both drives.
     - Write to drive 0 succeeds, write to drive 1 fails (nothing happens).
  2. Operating system crash
  3. On reboot, mirrored pair of drives is rebuilt.
     - Intent log flags an incomplete mirrored write, but does not have the data
  4. Read both drives: Write appears to have completed.
     - Drive 1 supplies an unused block that's full of zeroes.
  5. **Data change**: Drive 1 changes the value of its block (to garbage)
     - Reading this block from the mirrored pair is now inconsistent. (oops)
     - RAID repair (scrubber) may copy the garbage to drive 0. (ouch!)

Data Corruption: Single Parity RAID

- Single parity block covers multiple data blocks
  - RAID 3, 4 and 5
  - RAID 6 has a second set of parity blocks.
    - Not considered here for simplicity
- Initialize drives by reading data blocks and writing parity blocks.
  - Data blocks are not written as part of this.
- Consider a single set of RAID blocks, 3+1 (4 drive) RAID group.
  - Three data blocks on drives 0, 1 and 2, parity block on drive 3
- Data corruption sequence: Drive failure
  1. Write application data block to drive 2.
     - XOR write to drive 2, XOR write of that result to parity on drive 3
  2. **Data change**: Data block on drive 1 changes (to some garbage)
  3. Drive 2 fails and cannot be accessed.
  4. RAID rebuild of drive 2 data block will rebuild garbage. (ouch!)
     - XOR of drive 1 changes gets XORed into rebuilt drive 2 data.

Proposed Solution: Security

- Question #1: What happens when TRIM'ed blocks are READ?
  - ATA TRIM Answer #1: The result is "indeterminate". It may be data that was previously written somewhere else on the drive. **[WRONG ANSWER]**
- Acceptable outcomes of READ after TRIM:
  - TRIM has no effect on data stored in the block
    - Drive continues to store data, no change to drive behavior
  - The data is read as a pre-specified constant
    - e.g., all zeroes, all ones
- 08-149r1 proposal specifies these acceptable outcomes.

Proposed Solution: Data Corruption

- Question #2: What happens when a TRIM'ed block is READ multiple times without any WRITE or other operation that changes its value?
  - ATA TRIM Answer #2: The READ operations may return different values for the same block. **[VERY WRONG ANSWER!!]**
- This is unacceptable and has to be forbidden!
- Proposal for SBC-3 standard:
  - For each logical block operated on by a TRIM command:
    - **if** the TRIM command is followed by a READ command with no intervening WRITE commands or other commands that change the user data contained in the logical block,
    - **then** the data for that block returned by that READ command shall be returned for subsequent READ commands until the device server processes a WRITE command or other command that changes the user data contained in the logical block.
- Q: Does this also belong in the model clause?