Flash Data Reduction Techniques and Expectations

.. or how to fit two tons of fertilizer in a one ton truck.

Doug Dumitru

CTO EasyCo LLC

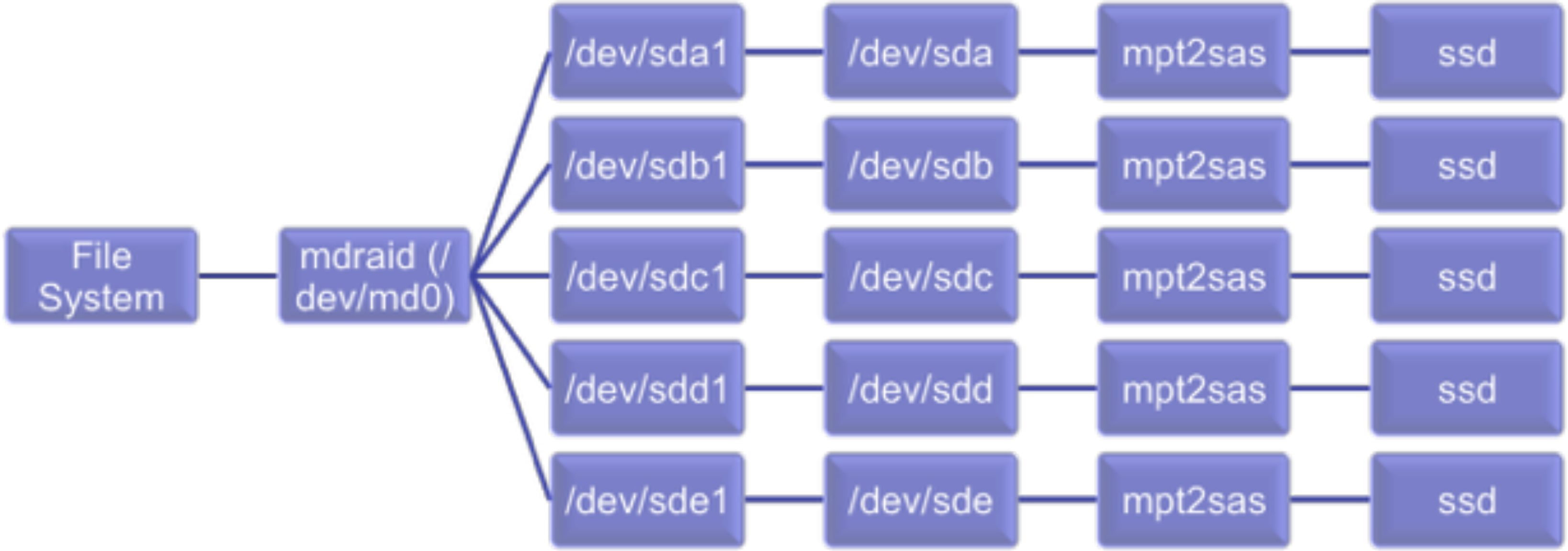
Block Device Basics

Block LBA Mapping

Most Block Devices have logic to handle Logical Block Addressing:

The RAW device starts at sector 0.
Partition tables are simple offsets to the sector number.

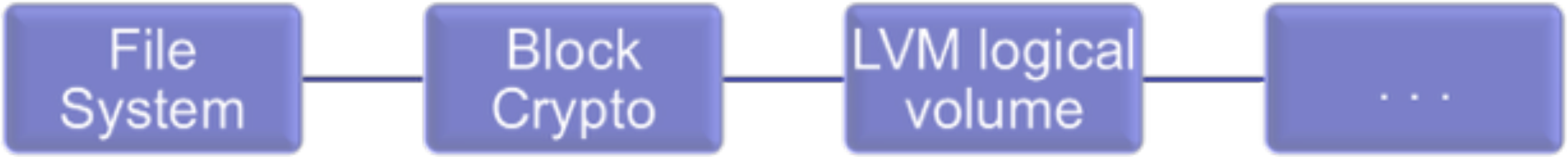
RAID mappings calculate where the block is stored across a collections of lower-level devices.



Mangling Block Contents

So far, the block stack has only dealt with “where” a block is stored.

Block stacks can also manipulate the contents of the blocks themselves.

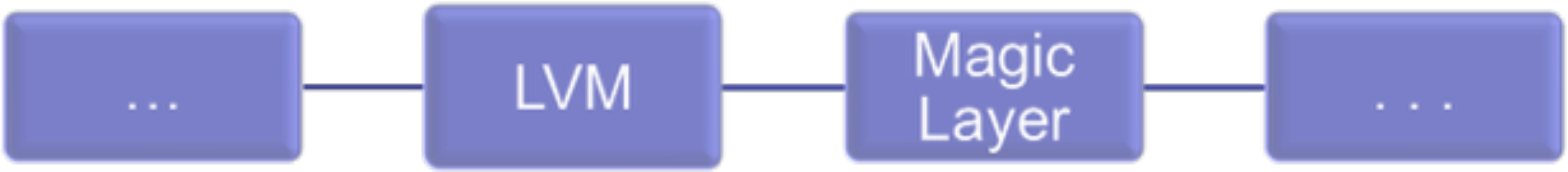


Magic Transformations

Now that we see how a block stack can re-arrange the location of blocks, and also manipulate the content of blocks, just how much trouble can we get into.

We have a few goals:

- Optimize performance and wear for Flash
- Optimize write behavior so that data is not lost or corrupted after a crash
- Implement “thin provisioning”
- Implement “block-level compression”
- Implement “block-level de-duplication”



Writing Linearly

To keep Flash happy, we want to write to the media in controlled linear segments. If you dig inside of Flash SSDs, the Flash itself only understands linear updates. It is the SSDs FTL (Flash Translation Layer) that allows SSDs to support random writes at all.

Our external "Magic Layer" should off-load the FTL so that this function is performed globally and not local to a single SSD.

Here we accumulate a group of block updates, and place them into a linear write segment:

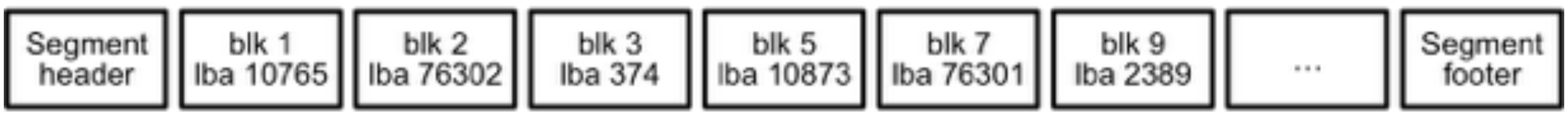
Segment header	blk 1 lba 89302	blk 2 lba 10472	blk 3 lba 9762	blk 4 lba 89103	blk 5 lba 21765	...	Segment footer

All of these blocks are the same size (4K). The header includes summary information, plus an array of the sector numbers associated with the blocks that follow.

Thin Provisioning

The header (and footer if the write segment is long enough) has information about each block that follows.

The header can also include information about blocks that have no data. This allows you to write blocks that are empty (all zeros) or full (all FFFFs) without having the block actually occupy any space on the target device.



Compressing

Now that we can write linearly, compressing is pretty easy to manage.

If we compress each inbound 4K block, we can store at least some of these blocks in less than 4K bytes worth of space. Then this layout looks like:

Segment header	blk 1 lba 392	blk 2 lba 9382	blk 3 lba 93	blk 7 lba 23	blk 8 lba 2	blk 11 lba 983	blk 13 lba 1045	blk 15 lba 221	…	Segment footer

With this layer, we have to track where a block is stored in more detail, but the write logic still maintains the advantages of 100% linear updates.

De-duplication

Now we de-dupe. With de-dupe, we look at the blocks contents and generate a unique ID using a function called a HASH.

With a unique ID, a write to a block that already exists is just like writing a zero or FFFF block.

The layer that remembers writes gets more complicated with de-dupe, but the write format remains the same.

Data Reduction Challenges

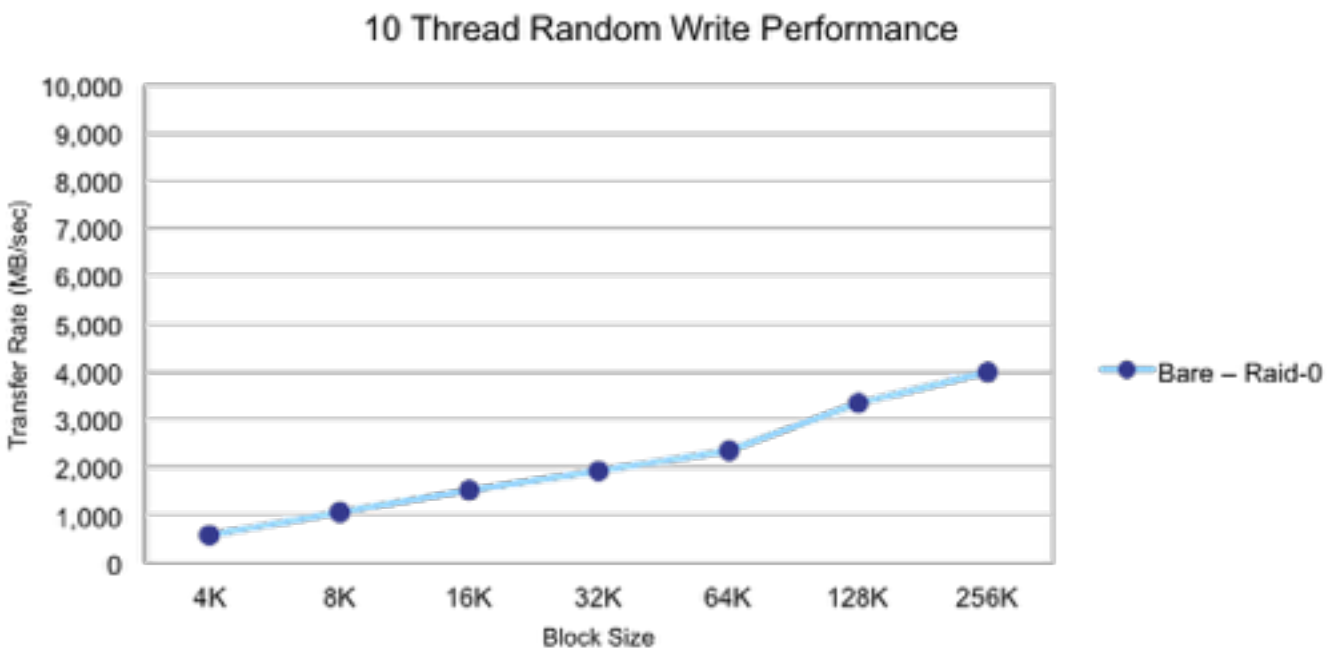
- Mapping requires some place to “remember” where blocks are stored
  - This can be a lot of memory
  - Thin provisioning, compression, and de-dupe put further demands on memory
  - Eventually, you have to store this data “on disk” which hurts performance.

Data Reduction Challenges

- Compression takes CPU Cycles
- De-duplication HASH functions take CPU Cycles
  - De-duplication requires read validation to keep hash collisions from corrupting data.

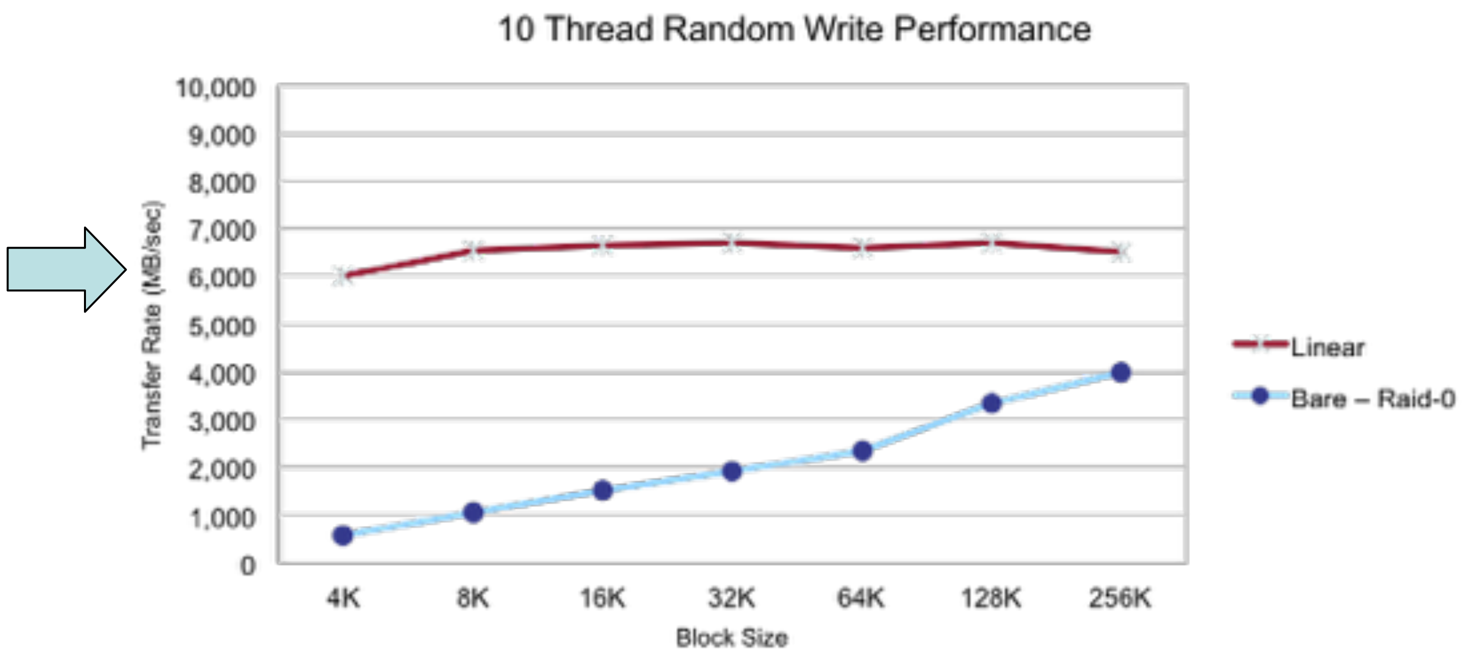
SSD Array Update Performance

- RAID-0 Base Line Performance



SSD Array Update Performance

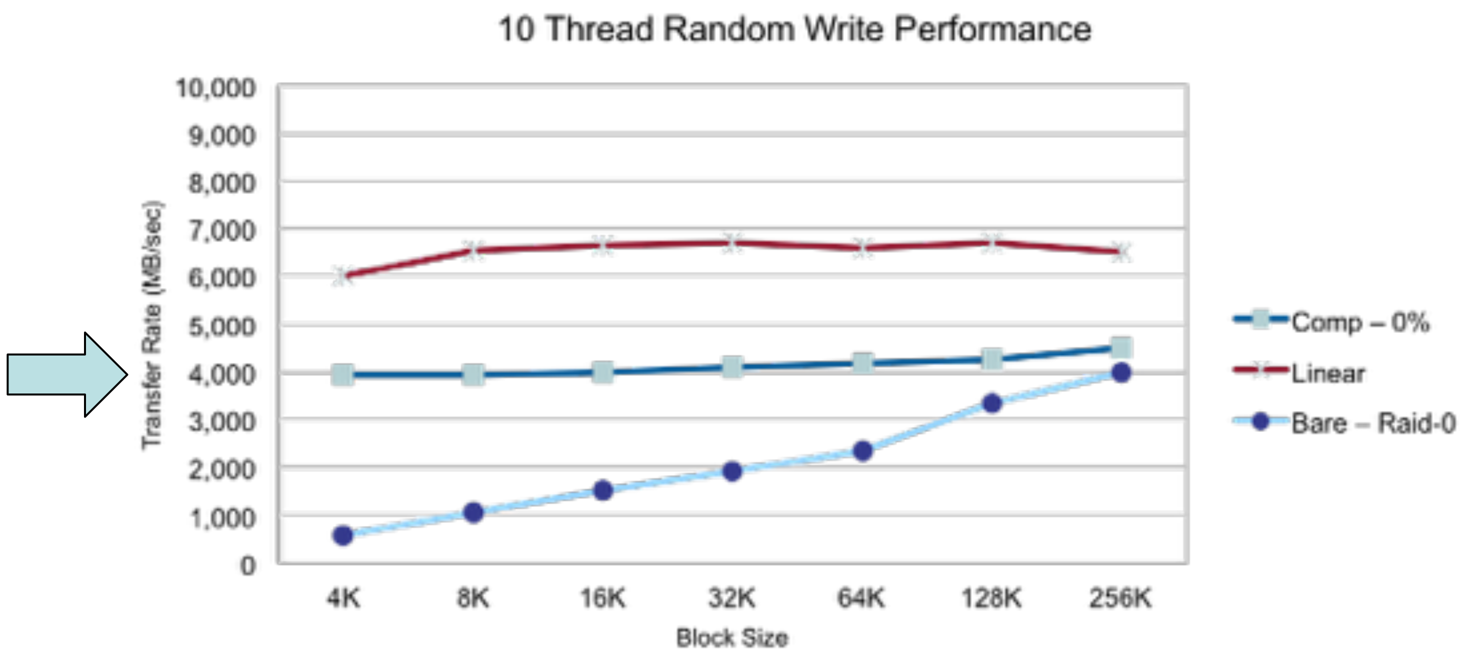
- RAID-0 Linear Writing



SSD Array Update Performance

- RAID-0 Linear Writing w/ Compression

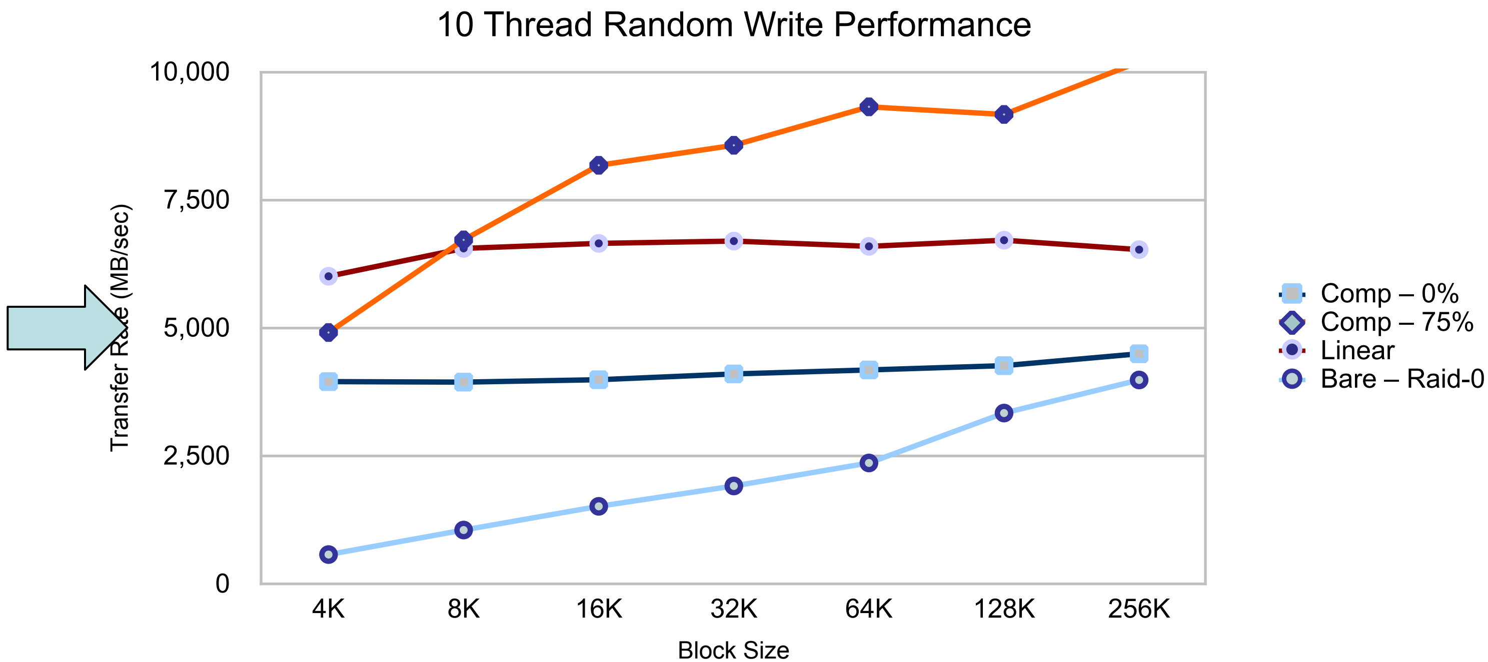
uncompressible data



SSD Array Update Performance

- RAID-0 Linear Writing w/ Compression

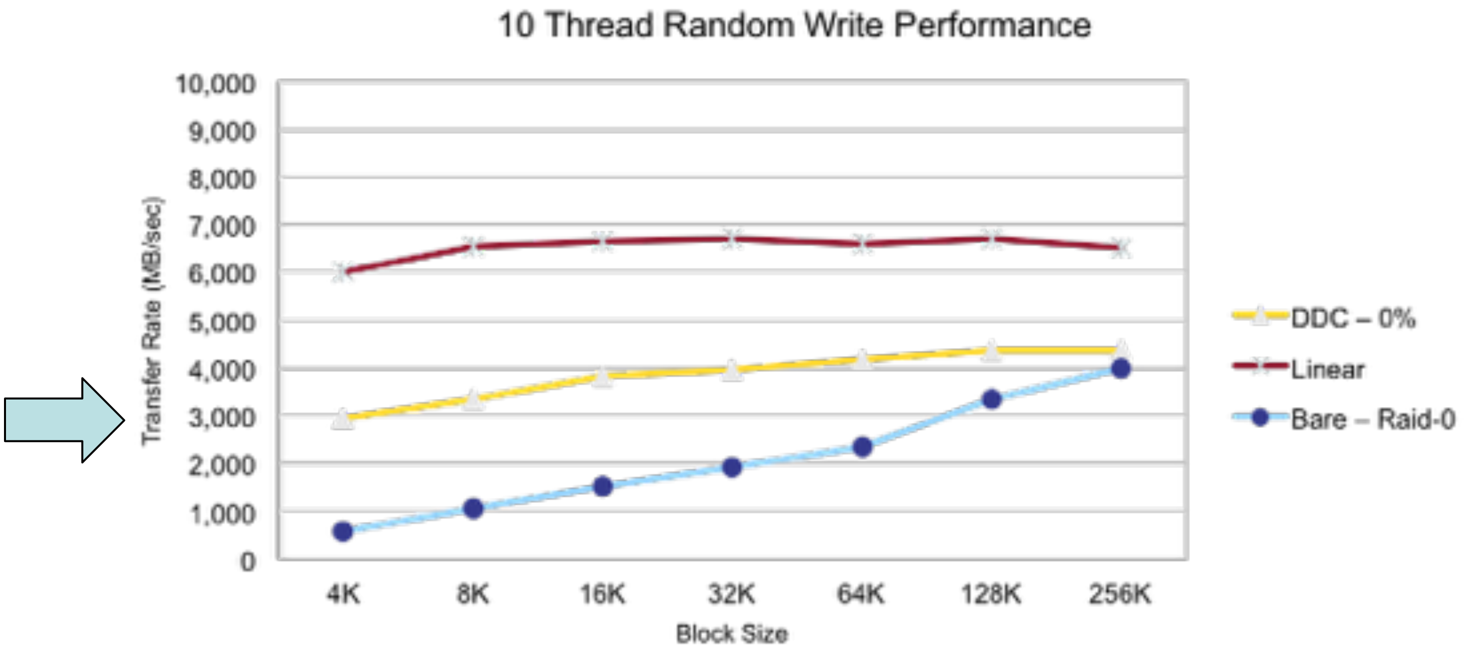
75% compressible data



SSD Array Update Performance

- Linear Writing w/ Compress and De-dupe

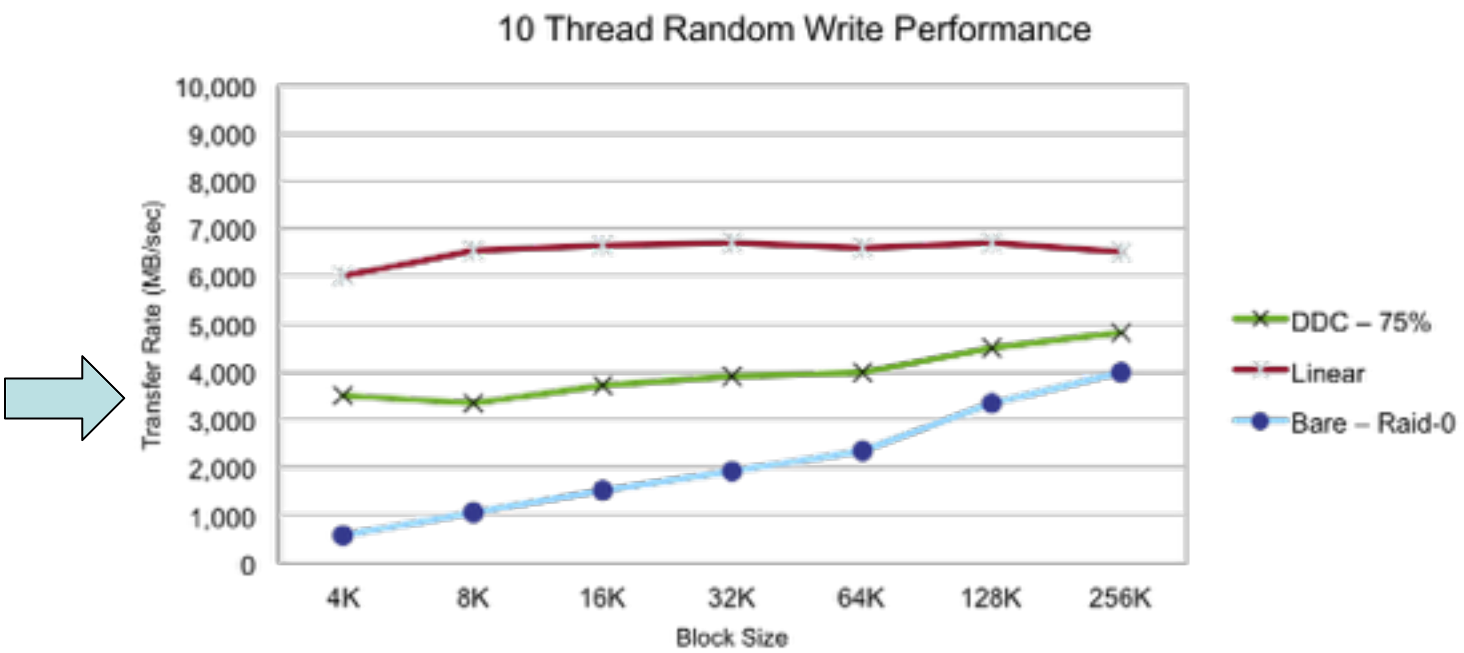
uncompressible data



SSD Array Update Performance

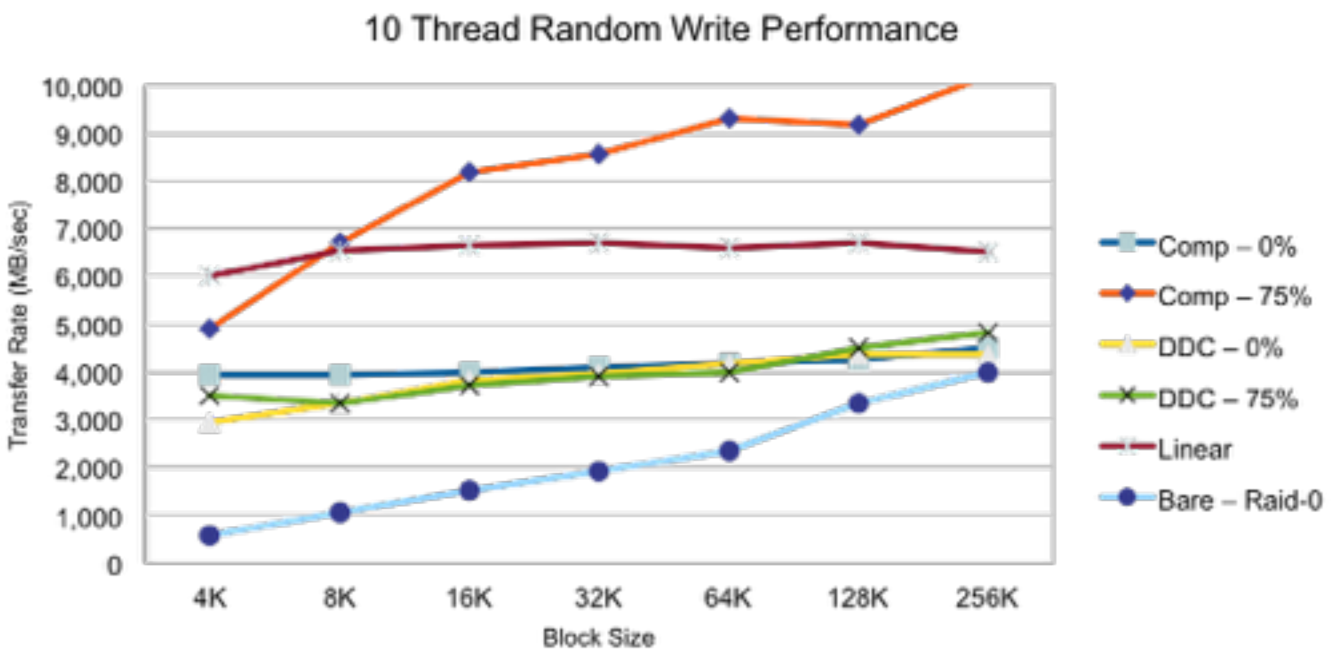
- Linear Writing w/ Compress and De-dupe

75% compressible data



SSD Array Update Performance

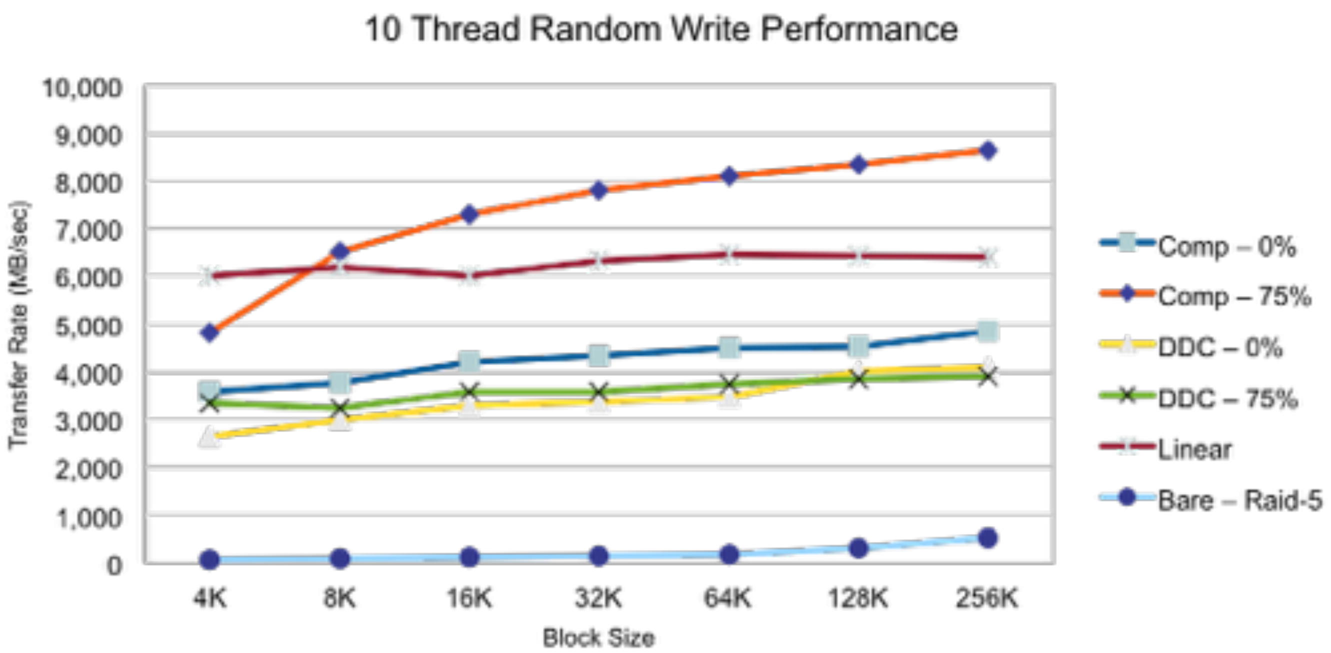
- RAID-0 Tests Compared



RAID for SSD Protection

- RAID-5 Tests

destroys native performance, but leaves linear writing intact.



Summary

- Block Layer Software can give you
  - Superior write performance
  - Superior flash wear
  - Improved crash data protection
  - Thin provisioning
  - In-line compression
  - In-line de-duplication

Will your Dataset “Reduce”

- Not all datasets compress
- Not all datasets have duplicate blocks
- Best applications
  - Virtual server farms
  - VDI
  - Databases
- Worst applications
  - Media and Entertainment
  - Encrypted / compressed files

Test your Data

- EasyCo has a simple, read-only tool that will scan block volumes and report compressibility and de-dupe potential.
  - Can be run on a single volume
  - Can be run on multiple volumes, even on different systems, and the results consolidated.
- http://easyco.com/dedupe-calc.htm

Summary

- Data reduction can lower storage costs while still maintaining excellent performance:
  - Drives pure flash storage costs below $0.20/GB
  - Keep symmetric read/write IOPS above 500K