A Look Inside NetApp Inline Data Compaction

Skip Shapiro
Technical Marketing Engineer, All Flash FAS and ONTAP Flash
NetApp

July / August 2016

In NetApp® ONTAP® 9, we added a new storage efficiency feature called *inline data compaction*. Because this concept is new to most people, I thought I would take some time in this article to explain how compaction works and how it interacts with our other storage efficiency features.

Inline data compaction acts on data chronologically, that is, as the data arrives at the storage controller. While the data is still in the controller memory, we take data chunks that would each normally consume an entire 4KB block on physical storage and we compact the chunks. With compaction, more than one chunk fits into a single 4KB physical block. Think of it as a suitcase-packing, trunk-packing, knapsack-packing type of problem. We're able to take I/Os that are probably padded with a lot of zeros, or empty space, remove the empty space, and take advantage of it. That's what compaction does.

Inline data compaction occurs during the process of creating the ONTAP consistency point (CP) write. It's like doing disassembly, part of the Tetris operation: I have these small I/Os. Can I quickly combine two or more of those I/Os into one physical box before I put it down on the storage media? Multiple patents have been filed for our inline data compaction. The method that we use is innovative. Compaction is enabled by default for NetApp All Flash FAS systems, and it's an optional feature that you can turn on for FAS systems, with either HDD-only aggregates or NetApp Flash Pool™ aggregates. In either case, there is no additional cost for compaction; it's in the core of ONTAP, just like compression and deduplication.

Compaction is an additive storage efficiency component. It's orthogonal to deduplication and is very much a complement to inline adaptive compression. With inline adaptive compression, we create compression groups if the data is more than 50% compressible. After data is compressed, we look for opportunities to further compact compressed data by pairing multiple small chunks or large chunks with small chunks into a single physical block. Because it's part of the CP process, compaction alone requires very little CPU overhead, 1% to 2% maximum. You shouldn't ever be in a situation in which a controller with 1% to 2% additional CPU usage puts you over the top. In that case, the system's already overloaded.

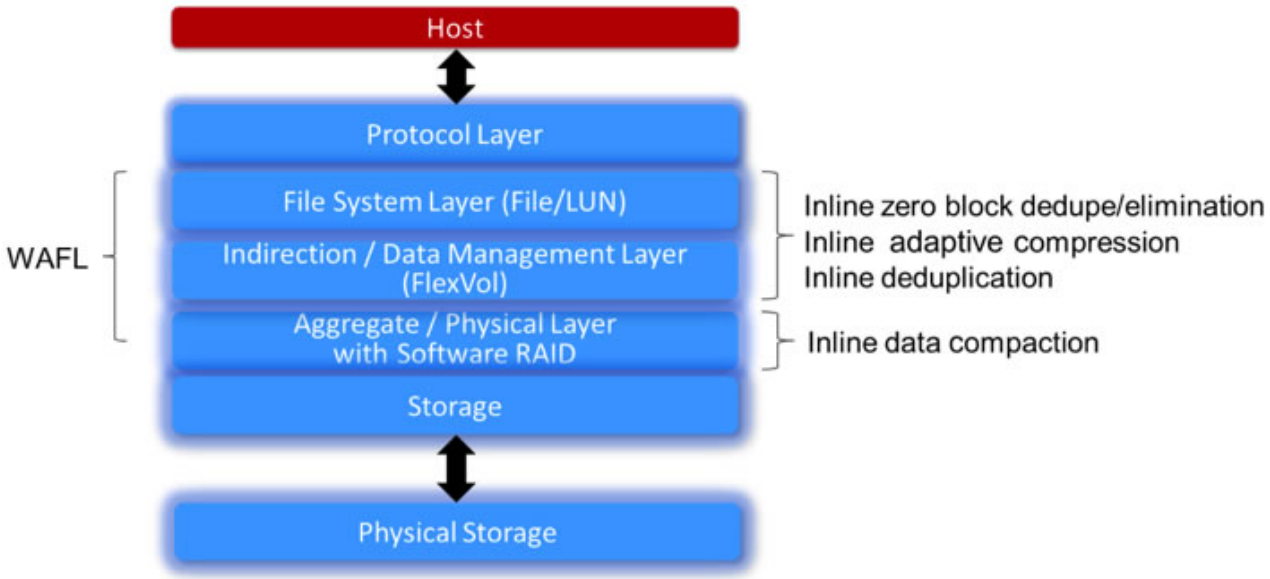
Although compression and compaction work very well together, it is not required that they operate together. A volume with many small files, for instance, might be a good candidate for compaction, but it wouldn't benefit much from compression.

Logically, as data enters the storage controller, the following is how the storage efficiency process works (assuming that all efficiencies are enabled):

1. The first thing that happens is that we detect any blocks with all zeros in them. For those blocks, we write nothing, we just update metadata. Essentially these blocks are merely a reference count.
2. Next, we apply inline adaptive compression. This process is very efficient at determining whether the block is compressible by 50% or more. We don't waste CPU cycles on trying to compress stuff by less than 49% (where we would get no compaction savings).
3. Then inline deduplication happens. This feature was introduced in ONTAP 8.3.2, in which only the data in memory was compared and deduplicated inline. With ONTAP 9.0, we expanded the size of the fingerprint hash store to include data that has been recently written to storage media. If you want to maximize the savings that result from deduplication, our best practice is to also do background (postprocess) deduplication on some schedule.
4. The last thing that we do is data compaction. Any data that was compressed by adaptive compression, or that has not been examined for compression, is eligible to be compacted. So, small, uncompressed files or data that is compressed by roughly 75% or more by inline adaptive compression is eligible. The compaction process pairs and fits two or more of those chunks into a single 4KB physical block before sending that block down to storage. The higher the compression rate is and the smaller the files are, the higher the compaction rate is. So, we can get a great multiplicative effect when combining small blocks, compression, and compaction.

You probably will not know what impact the compaction is going to have until you start to use it. Inline compaction is a heuristic process within a CP. It samples the first 100 I/Os that come in to see what the compaction rate is, then it applies that rate to the next 100 that come in. If the compaction rate changes upward, the process says, "Okay, I will assume a higher rate going forth." If it's getting a lower rate of compaction, it decreases the rate. When the next CP comes, the process starts all over again. It's an iterative process that seeks the lowest CPU usage but the highest compaction rate.

Figure 1) NetApp inline storage efficiencies are applied as data moves through the ONTAP stack.



Source: NetApp, 2016

You could use inline data compaction alone (without compression or deduplication) if you wanted to. Let's say that you had a small file environment only, that is, files that are on the order of 2KB or smaller. Those small files are not compressible, and it's unlikely that you will see much in the way of deduplication savings. Inline data compaction, however, will result in space savings when two or more files can be stored in a 4KB physical block.

In terms of logical data replication, if the source and destination volumes have the same efficiency policies enabled, space savings are preserved, without any rehydration. An example is an All Flash FAS system with all storage efficiencies enabled—which is the default configuration—replicating to a FAS system that is being used as a NetApp SnapVault® destination. If you want to preserve the space savings from deduplication, adaptive compression, and inline data compaction, you must have all of those policies enabled on the destination system as well.

Conversely, if the destination FAS system has none of the space savings policies enabled and you're replicating from an ONTAP 9 All Flash FAS system, the data is inflated when it is written to the destination. You have no compressed blocks, no deduplicated blocks, and no compacted data, either. So, if you want to preserve those space savings, source and destination volumes must have the same storage efficiency policies. By the way, our best practice is to enable all efficiencies on the destination volumes when replicating from an All Flash FAS source system; that way, you preserve the space savings on the destination.

To summarize, inline data compaction doesn't alter logical data at all; we just figure out how to pack the data more efficiently. Unless you're in an environment with all small files, the best way to think of compaction is as a multiplier on top of inline adaptive compression. If you have partially filled I/Os, that's where compaction can help you. With low CPU overhead and more free space to store data, inline data compaction is a no-brainer for All Flash FAS systems.

Skip Shapiro is a NetApp Technical Marketing Engineer responsible for NetApp All Flash FAS, Flash Pool, and Flash Cache™ technology. Skip will deliver two technical sessions during the NetApp Insight® conference in Las Vegas and Berlin. For more information, see the following links:

NetApp Insight Las Vegas (September 26â€“29):
NetApp ONTAP 9: Predictable Performance in SLA-Driven Business Environments
All Flash FAS Technical Deep Dive

NetApp Insight Berlin (November 14â€“17):
NetApp ONTAP 9: Predictable Performance in SLA-Driven Business Environments
All Flash FAS Technical Deep Dive

Quick Links Tech ONTAP Community > Archive > PDF >