ODX with FlashArray: An Engineer's Perspective

As an engineer, it is not often that you get to work on a feature that will produce up to an 80X performance improvement. I got that chance when we developed the ODX feature for FlashArray.

What is ODX?

ODX stands for *Offloaded Data Transfer*. Microsoft introduced ODX in Windows Server 2012 to offload copy operations from servers to storage arrays. Without ODX, a host copying a large block of data would read the data from the source location on an array, and write it to a different location on the same array.

The VMWare VAAI facility and other applications use the SCSI EXTENDED COPY (XCOPY) command to speed up bulk data copying; ODX performs the same function for the Windows Server operating system. Both mechanisms are used extensively for virtual machine deployment and migration.

Although the SCSI commands and protocols for XCOPY and ODX differ, the end result is the same: copy operations are offloaded to a storage array. Offloading copy operations provides several important benefits:

- lower latency
- lower host CPU utilization
- reduced network utilization

Beyond these obvious benefits, with FlashArray's unique data organization, most copy operations are

simple metadata updates. This results in bulk copy operations completing almost instantaneously, regardless of the amount of data copied. But more on that later…

How Does the ODX Protocol Work?

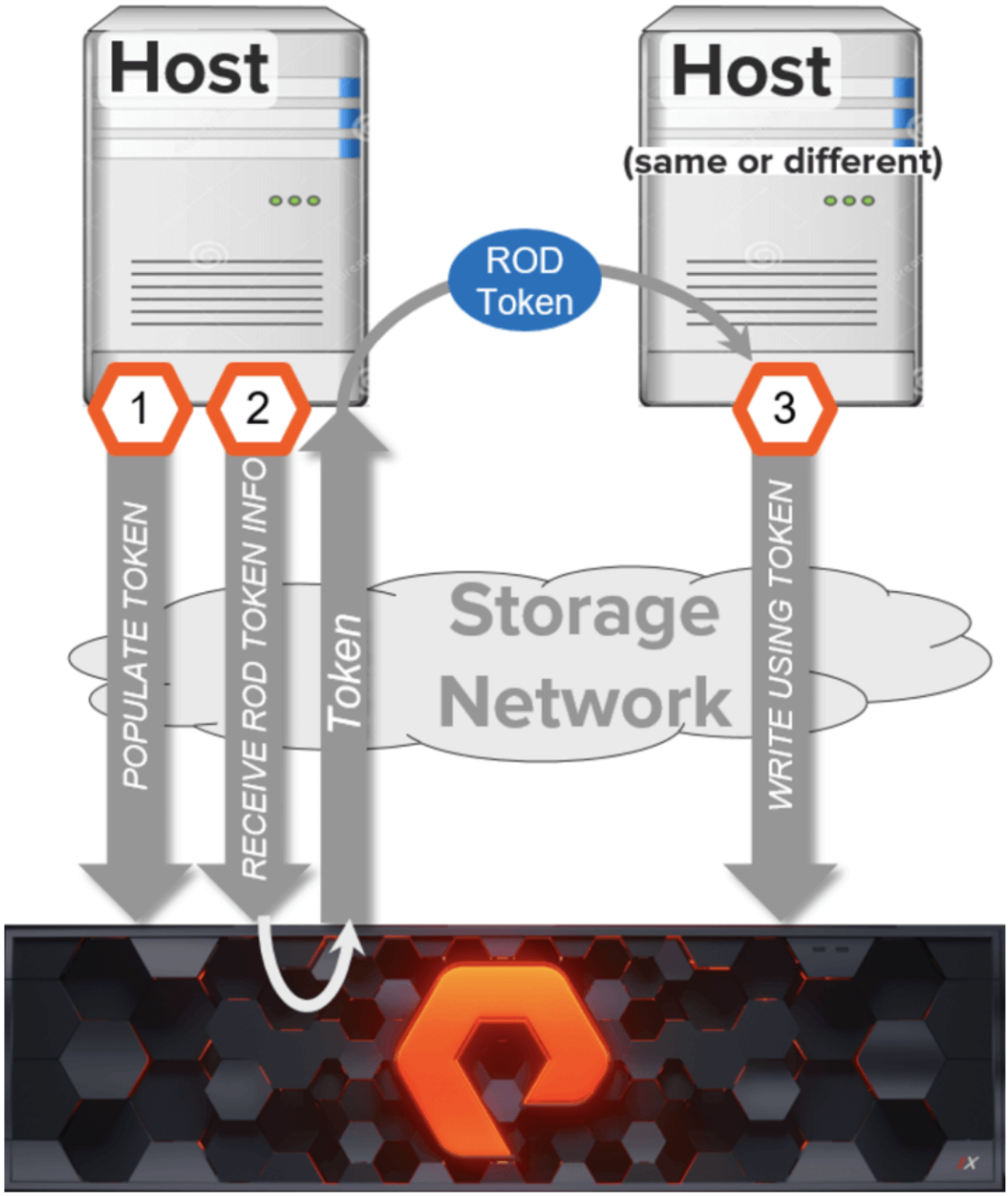
ODX uses a token based mechanism for making point-in-time copies. Rather than routing the data to be copied through a host, ODX commands direct an array to copy the data by specifying a token that represents the source data as of a single point in time. The tokens are known as *Representation Of Data* (ROD) tokens. They are 512-byte opaque structures that a storage array uses internally to identify a point-in-time copy of source data.

To direct an array to perform a point-in-time copy, a host sends the following sequence of commands:

SCSI command	Command Description	How Purity handles the command
1. POPULATE TOKEN	Directs an array to make an internal point-in-time copy of source data.	Stages the source data in a special volume so that it may be copied at a later time.
2. RECEIVE ROD TOKEN INFO	Directs an array to send a ROD token to the host. The ROD token represents the data staged during the execution of the POPULATE TOKEN command.	Generates a ROD token and returns it to the host. The ROD token contains information that enables the array to uniquely identify the point-in-time copy of the source data previously staged.
3. WRITE USING TOKEN	Directs an array to copy the source data represented by a provided ROD token to a destination specified in the command.	Copies the staged source data to the specified destination.

How does an array know which ROD token to return when the host sends a RECEIVE ROD TOKEN INFO command?

The ODX protocol requires the host to specify a unique identifier to link corresponding POPULATE TOKEN and RECEIVE ROD TOKEN INFO commands. An array must return the ROD token that represents the data staged by the POPULATE TOKEN command that came from the same host and contains a matching identifier as the one the host provides in the RECEIVE ROD TOKEN INFO command.



A host that initiates a point-in-time copy may use the token, or it may pass it to another host, to copy the data to a different location. A WRITE USING TOKEN command may be issued by any host in possession of the ROD token, but it must be sent to the same array that generated the token, otherwise it will be rejected.

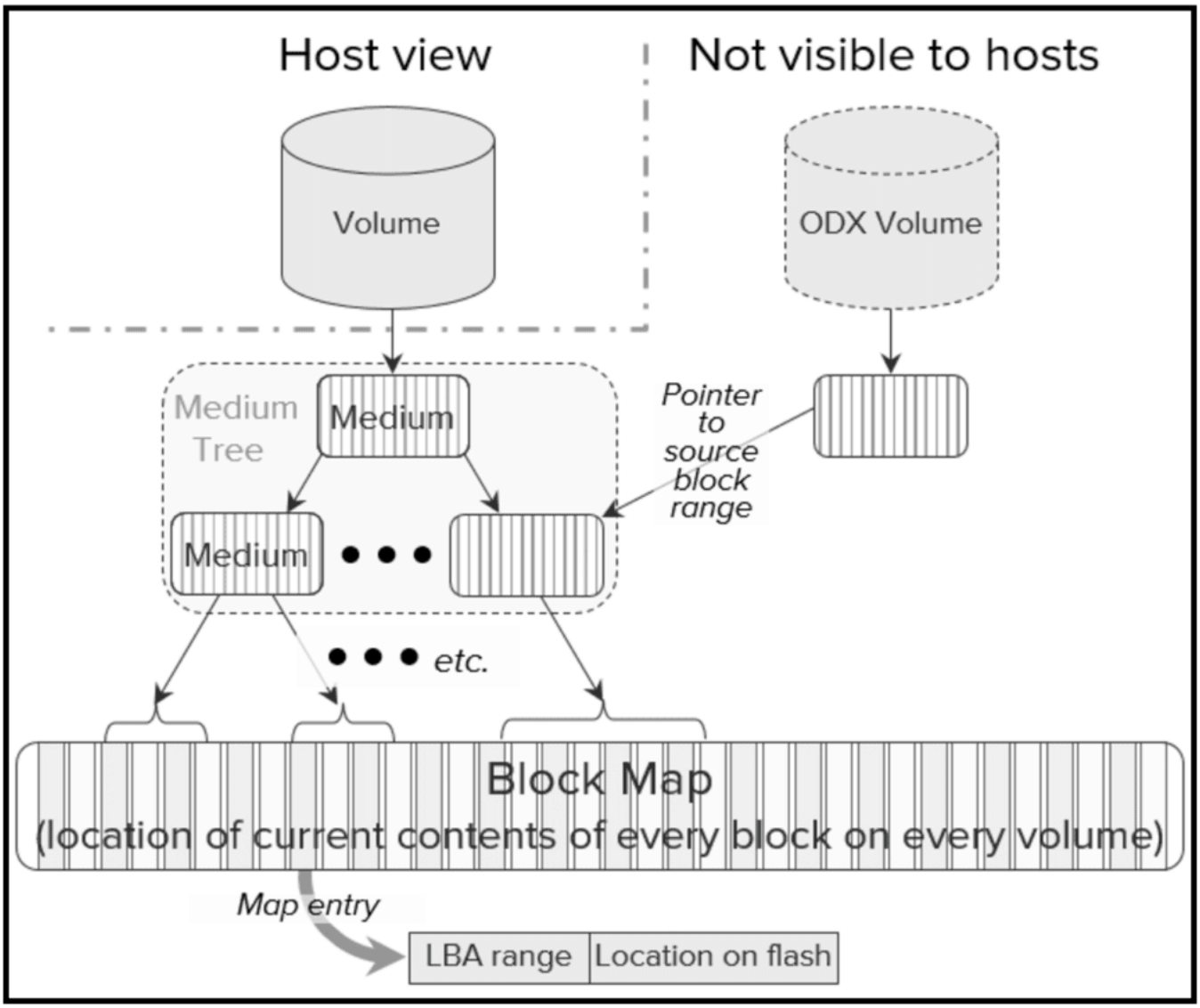
In addition to point-in-time-persistent ROD tokens, FlashArray also supports *Zero ROD* tokens. Rather than sending a write with a lot of zeroes, a host can offload the operation to the array by using Zero ROD tokens to zero out a range of blocks. The process to zero a block range is different than the point-in-time copy because there is no source data to copy, and the Zero ROD token is generated by the host rather than the array. To direct an array to zero out a block range, a host sends the following command:

SCSI command	Command Description	How Purity handles the command
1. WRITE USING TOKEN	When this command is sent with a Zero ROD token, it directs an array to write zeroes to a destination specified in the command.	Writes zeroes to the specified destination.

How Does FlashArray Copy Data So Quickly?

An array that receives a WRITE USING TOKEN command may copy the associated data to the specified destination as a background operation. ODX includes a polling mechanism that hosts can use to determine the progress of a copy, and for an array to indicate when it is finished.

With FlashArray, however, copying data between volumes or block ranges within a volume is nearly instantaneous because it updates the metadata representation of the address space to point at the locations of the source blocks. Updating the metadata representation is so fast that a FlashArray always responds to WRITE USING TOKEN commands immediately with a status of copy complete.



FlashArray's Purity software maintains metadata representations, called *mediums,* that point indirectly to the current physical locations of data in volumes. Mediums are organized as trees whose leaf nodes point to areas in an array-wide *block map* that indicate where data is stored.

Purity creates snapshots of an entire volume, or copies a range of blocks within a volume, by simply allocating a new medium and pointing it to the part of a source volume's medium tree that represents the block range to be copied.

This is really useful for ODX because making a point-in-time copy of a range of blocks uses the same mechanism.

The POPULATE TOKEN command doesn't specify a destination, it just directs the array to stage a range of blocks to be copied later when a WRITE USING TOKEN command is received.

So where does a FlashArray put the data it stages in response to a POPULATE TOKEN command?

Purity maintains a set of internal *ODX* volumes. These are otherwise normal volumes, however they are not visible outside an array. When a POPULATE TOKEN command directs a FlashArray to stage source data, Purity simply copies it to an ODX volume.

Doesn't that waste physical storage?

No, because Purity doesn't actually copy data to an ODX volume; it simply updates pointers in the ODX volume's medium tree to point to the source volume medium nodes that represent the block range to be copied. Only if a host overwrites source volume blocks in the copied range does Purity copy data to maintain the point-in-time state of the staged data.

Likewise, Purity executes a WRITE USING TOKEN command by copying the data staged in an ODX volume to the destination volume specified in the command. Again, it doesn't actually copy the blocks; it simply updates the destination volume's medium tree to point to the source data's location and indicates to the host that the copy is complete.

In the case of a WRITE USING TOKEN command that contains a Zero ROD token, Purity doesn't actually write zeroes to the destination block range. Rather it updates the destination's medium tree to point to a special zero medium.

So where does the Array store all those tokens?

Here's the cool part, it doesn't… Purity doesn't generate a token when it executes a POPULATE TOKEN command; it stores only enough information to indicate where to find the staged data. The team developed a lockless data structure that holds the information required to generate the token. As it executes a POPULATE TOKEN command, Purity stores the information in an entry in the data structure. To execute a subsequent RECEIVE ROD TOKEN INFO command, the software locates the entry with a matching identifier and generates the ROD token to send to the host.

That's all very interesting, But how well does it work?

The table below is a sample of test results observed during the development of FlashArray ODX support.

Use Case	Without ODX	With ODX	Improvement
Hyper-V Live Migration (100 GBytes)	131 sec	20 sec	6.5X
Hyper-V Fixed VHD(X) Provisioning (50 GBytes)	36 sec	1 sec	36X
Hyper-V File Copy (50 GBytes)	83 sec	1 sec	83X

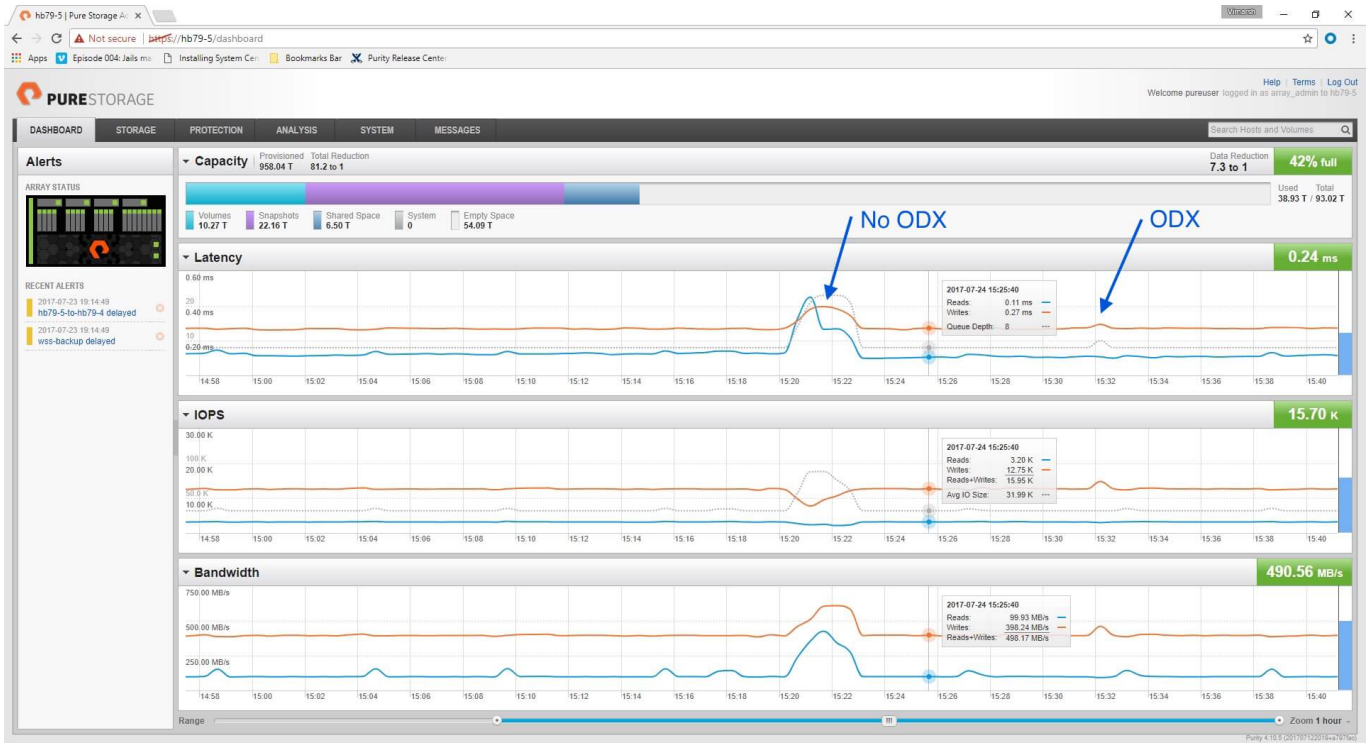
The first row of the table shows the time to perform a live migration of a 100 GByte virtual machine through Hyper-V while I/O is going to the VM. Without ODX, migration took more than six times as long as it did with ODX.

The second row shows the time to provision a 50 GByte virtual hard drive (VHD). With ODX, the host uses a Zero ROD token to direct the array to zero the blocks required for the new virtual drive.

The third row shows the time to copy a 50 GByte file. In this case, the benefit was huge: about an 80X reduction in copy time!

Keep in mind this is just a small sample of use cases. Many factors will affect the resulting performance gain. But what remains the same is that ODX will consistently provide significant performance improvements.

The screenshot below of the FlashArray GUI dashboard shows the effect of ODX in the live migration example. The blue and orange lines represent reads and writes to the source and destination volumes. When ODX was not being used, latency and bandwidth spike just after 15:20 and remain high for the duration of the copy (until approximately 15:23). When ODX was used (at around 15:32), there was barely a blip. The graph reinforces what the table above indicates: migration is much faster when ODX is used. This clearly indicates the benefits of ODX: reduced latency, reduced bandwidth, and most importantly, much shorter migration time.



Pure Simplicity

As with every feature that Pure supports, engineering made an extreme effort to ensure ODX would be both simple to use and efficient. Once you update to a release that supports ODX, there is nothing more you need to do to benefit from the performance gains. There is no special license needed, there are no special configuration requirements, and no knobs to tune. ODX will be on by default, and any Windows Server that supports ODX will automatically begin offloading copies to the FlashArray.

Microsoft has approved our implementation and FlashArray is certified and complies with the Windows

Offloaded Data Transfers Hardware Certification Kit.

FlashArray's unique internal data representation makes copying large block ranges, either within a volume or between volumes, nearly instantaneous. With the addition of ODX to its command repertoire, the benefits of offloading copy operations from hosts by performing them within the array are now available in Windows host environments.

I can't wait to see how customers react when they see just how fast this is. I'll leave you with a quote from one of our Beta customers: **WOW! Copying 8GB is faster than the window needs to appear.**