Reference Guide

Technical Reference Guide

Using JetStream Accelerate Software in AppDynamics Environments

Version 1.2a, Updated April 2018

JetStream SOFTWARE

Table of Contents

1. Introduction

This document is a basic technical introduction to the JetStream Accelerate software for I/O acceleration. This document is intended to serve as a complement to the product documentation and any training provided by the JetStream Software engineering team.

This document describes the capabilities of the JetStream Accelerate software version 3.9.1 for Linux. This particular document describes the standard "out of the box" functionality and operation of the software, but particular focus is paid to the use of the software to optimize the performance of AppDynamics controllers. This focus introduces some assumptions about typical deployments:

- The primary server operating system is Linux-based (the AppD Controller runs on Linux).
- The primary operating mode is write-back (the typical AppD I/O workload is 99% of writes).
- Flash devices are deployed redundantly in the servers.

1.1. Objectives and Overall Design

The JetStream Accelerate software is designed to increase application performance and server workload capability by optimizing storage I/O. It does this by enabling solid-state memory in the server to provide a low-latency target for frequently accessed data ("hot data"). Because data in the solid-state device is presented as a part of the native storage volume, storage volumes appear exactly as they would if all data resided in a traditional shared storage environment. However, because most I/O operations actually run on the solid-state device(s) in the server, the result is much faster performance and reduced overhead on storage systems.

Just as the operation of the JetStream Accelerate software is transparent upward to file systems, databases and applications, the software is compatible with any standard block-based storage. Any SAN or DAS storage is supported. Additionally, there are no special requirements for the solid-state devices, which may be attached to the server via PCIe interfaces (including NVMe), SAS, or SATA.

The three core objectives of the software are:

- Improve application performance by reducing storage I/O latency
- Reduce I/O contention on the server by optimizing read and write operation execution
- Minimize overhead on storage infrastructure by reducing I/O traffic on storage systems

1.2. Supported Environments

JetStream Accelerate works with virtually any x86 server, and has minimal CPU and memory requirements. The software supports block-based storage, either SAN or DAS. The software is available for Linux, Windows Server, and VMware vSphere environments.

1.3. SSD Support and SSD RAID

JetStream Accelerate will work with any solid-state device from any vendor. The software employs a "Logical SSD" (LSSD) as a data accelerator. Any resource that the server operating system can see as a logical volume may be used as an accelerator. This would include a single solid-state device, a partition on a single physical device, or multiple devices configured via software or hardware RAID. For software RAID, mdadm is recommended.

Note: In this document, and in most JetStream Software documentation, the LSSD used by the software is often simply referred to as the "SSD" regardless of the underlying physical device(s) being used.

When the data accelerator comprises multiple solid-state devices in a RAID configuration, it is recommended that they be attached via the same type of interface, be of the same size, and provide approximately equivalent performance. The recommended RAID types are RAID 0, RAID 1 or RAID 10. When the accelerator is used in write-back mode, RAID 1 or RAID 10 is recommended, to protect against loss of data in the event of a physical device failure. This same RAID configuration is recommended for AppDynamics clusters with two redundant Controllers.

When solid-state devices are mirrored with mdadm, the health of the RAID configuration can be monitored by the software to detect RAID degradation. If it appears that a physical device has failed or is in danger of failing, the software can automatically switch the accelerator to bypass mode, and flush dirty data to storage. This would effectively "turn off" acceleration, but it would eliminate the risk of a single point of failure, in which data not yet written to storage resides on just one physical SSD within the server. If this degree of data protection is not required (e.g., because data is replicated between two servers in a cluster), the automatic switch to bypass mode may be replaced by a simple alert.

Note: In AppDynamics cluster configurations, this automatic switch to bypass mode usually is not enabled, because the failure of a single solid-state device does not expose a single point of failure in the cluster, because a Controller can fail over to the other server in the cluster.

1.4. Operating System Kernels Supported (Linux)

The JetStream Accelerate software supports CentOS/RHEL 6.x versions and 7.0 through 7.4, from a single installer. Additionally (if needed), SLES 11 SP3, SP4 and SLES 12 are supported with an installer specifically for SLES. Ubuntu 14.04 is also supported, with its own unique installer.

2. Core Architecture

This section describes the core features and operations of JetStream Accelerate.

2.1. How the Software Fits in the I/O Stack

The JetStream Accelerate software for Linux runs as a block device in the Linux kernel I/O stack, installed as a loadable kernel module. Additional shadow block devices are created for data access. For example, when data volume /dev/sda is accelerated, the software creates a /dev/fio/sda shadow block device, which is then used for all application access.

The raw SSD and data volume are used exclusively by the software. To prevent data corruption, the driver rejects any attempts to directly access caching device and accelerated data volumes (/dev/sda in the above example).

2.2. Volume Formatting

In order to use a solid-state device (or more precisely, the LSSD), JetStream Accelerate formats it in a unique way for its own use. This means that when a SSD is identified for use as an accelerator, the device is reformatted and any data on the device will be lost. Looking at the device with operating system tools will show a volume as “not formatted.”

The software prevents direct access to any SSD assigned as an acceleration device. However, the devices could be deleted, for example using a disk manager, which would result in a loss of all data on the device.

Understanding how the software places data on the SSD and integrates the SSD with the underlying storage architecture will help prevent the most common mistakes, including:

- Attempting to place data onto the SSD manually prior to installing the software, rather than allowing the software to automatically populate the accelerator with data
- Thinking the SSD volume is faulty when it is seen as “not formatted” in a disk manager
- Deleting an SSD volume still being used in write-back mode, corrupting the data volumes that are using the SSD for acceleration
- Attempting to accelerate the server boot volume

2.3. Write-Back Mode

The software can operate in either write-back mode or write-through mode. For AppDynamics environments, the software is used in write-back mode, due to the need to improve the performance of write operations on the Controller.

When the accelerator runs in write-back mode, the write operation is acknowledged as soon as data is written to the LSSD. For a period of time, the data resides on the LSSD only (this is called “dirty data”) and the data is written to back-end storage later (this is called “flushing”). The result is that write-back mode significantly reduces latency for write operations as well as reads.

2.4. Write-Back Mode: Important Considerations

Accelerating I/O in write-back mode is essential for many types of workloads. Transactional databases and business intelligence applications often make large numbers of small, random writes, sometimes overwriting the same data blocks many times in a brief timespan. This kind of I/O profile is ideal for write-back mode. However, some precautions should be taken to ensure that data is protected when write back acceleration is enabled.

The physical SSDs comprising the accelerator’s Logical SSD (LSSD) should be mirrored. Because the LSSD holds data that has not yet been flushed to the underlying storage, the failure of a lone SSD device could cause data to be lost, and possibly corrupt the entire storage volume. Even when AppDynamics Controllers are deployed redundantly in clusters, it is strongly recommended that each server’s LSSD comprise two physical SSD devices in RAID1 configuration.

In general, off-host snapshots should not be relied on for data recovery while the software is operating in write-back mode. If a snapshot is made by the back-end storage system, some data on the SSD will not have been synchronized (flushed) to the backed-end or may have been written to the underlying storage in a different order from the application writes to the volume. The back-end storage is not aware of this unflushed or out-of-sequence data and the off-host snapshot will be inconsistent with the volume.

2.5. Data & Metadata Persistence

Because the accelerator is based on a solid-state storage device, the data remains persistent even in the event of an unexpected server restart. In write-back mode, data is preserved during system restarts because the accelerator retains “dirty data” (data not yet flushed to storage). When the server is restarted, the accelerator is immediately available and does not require data to be repopulated. This is true even if the server crashes and restarts unexpectedly. The accelerator’s essential metadata is also persisted to the SSD, so the software can recover the necessary metadata from the non-volatile media and resume operation in a fraction of a second.

2.6. Circular Buffer Design, Flushing Control

JetStream software generally writes data to the solid-state device resource sequentially[1], in blocks of variable size. This minimizes garbage collection and also allows multiple write operations to be coalesced into a single write operation on the accelerator, reducing metadata overhead, and consequently the memory footprint of the application. The data write point progresses through the SSD sequentially, until it is fully populated and then begins writing again from the start, populating the data in a circular manner. This is sometimes summarized as "write like a log, view like a cache."

In write-back mode, data is held in the SSD ("dirty data") and must be flushed regularly. Once the accelerator is warmed (populated with data based on storage I/O behavior) it is primarily populated by a mixture of clean data (data that has been flushed to storage) and dirty data (data that has not yet been flushed to storage). Having a large amount of dirty data reduces overhead on storage in two ways:

- *Write cancellation:* If the same data block is written multiple times before it is flushed to storage, multiple application-level write operations result in just one storage write.
- *Write coalescing:* when data is flushed to the back-end storage, the data from multiple write operations to the accelerator can be coalesced into a single, larger write operation to storage.

In order to provide space that is readily available for new data writes, the flush point progresses through the SSD space ahead of the write point, flushing dirty data to create a section of the SSD that contains clean data only – data that has been flushed to storage, and whose space can be instantaneously reclaimed for writing new data.[2]

The size of this space can be increased by designating a larger amount of space between the write point and the flush point. Doing this will increase the amount of "readily claimable" space containing clean data only, and reduce the amount of space containing clean and dirty data. Increasing this setting will also cause the accelerator to do more aggressive flushing, reducing the benefits of write cancellation and write coalescing, potentially increasing contention for storage bandwidth.

2.7. Dynamic Space Allocation

When the administrator configures an accelerator, its size is fixed cannot be changed without purging and rebuilding. However, within the accelerator, space is allocated dynamically among the accelerated volumes. With a dynamically allocated space, turning acceleration on or off for volumes doesn't

---

[1] Mostly. There are exceptions, such as when data is overwritten in place on the SSD.

[2] The parameter called FlusherFreeAndCleanGoalPercent controls the percentage of the capacity that is dedicated to clean data only. The default setting for this parameter is 10%.

require any “rebalancing” of static partitions. Additionally, as I/O activity shifts from one volume to another, the software responds automatically, providing space for the data volumes that need it most.

In the event that an administrator wishes to guarantee that a fixed amount of space is always allocated to a specific volume (or volumes), it is possible to create a separate accelerator specifically for that purpose, and provide it for the exclusive use of the volume(s).

2.8. Read Ahead and Pre-Fetching Algorithms

The algorithms used by the software enable “read-ahead” (or pre-fetching) of data. For example, when a read request indicates that an 8KB block of data should be moved to the accelerator for future read requests, a 32KB block containing that 8KB will be read into the space, so if adjacent data are required for a subsequent read operation, they will be already available.

2.9. Write Stream Recognition

The software recognizes write-streams – large amounts of contiguous data that are written to storage. Because a write stream could take up a large amount of SSD space, and the speed at which contiguous data can be written to a solid-state storage is not usually much faster than writing to disk, the software recognizes the write stream and allows it to bypass the accelerator, writing directly to the back-end storage. The administrator can change the default size threshold at which a write stream will bypass.

3. Features & Operation

3.1. Maximum 16TB Size, Up to Four Accelerators Per Server

The maximum amount of data that can be handled by a single accelerator is 16TB. JetStream Accelerate allows up to four independent accelerators to be installed on a single server. These different accelerators could use solid-state devices of different sizes or performance levels. Furthermore, different accelerators can operate in different modes (write-back or write-through). However, a volume accelerated by one accelerator cannot use a different accelerator at the same time. The maximum number of storage volumes that can be accelerated per server is 2048.

3.2. Designating Volumes for Acceleration

During configuration, the administrator is asked to select which volumes he wishes to accelerate. Please note:

- The boot volume cannot be selected for acceleration.
- Volumes can be selected or unselected for acceleration at any time.

3.3. Aggressive Warming

When an accelerator is first started, certain default parameters are automatically set to capture data more aggressively than they would during ordinary operation. This enables the accelerator to be populated faster, and more I/O operations can be accelerated sooner.

3.4. Maintenance

If the operating system needs to be re-installed, the administrator should switch to bypass mode, stopping acceleration for all volumes, then wait for the dirty data to be completely flushed, and then reinstall the OS. In the event that the system is not bootable and cannot be flushed to the back-end storage, the OS should be reinstalled. The newly re-installed OS will see the accelerated volumes as "unformatted" until the software is also re-installed. At this point, the accelerated volumes should now be recognized by the software, and the data for the volumes will once again become available.

The "rule of thumb" for handling SSDs used for acceleration should be essentially the same procedure used for any SSD utilized as a primary storage volume. In general, SSD hot swap operations are not supported for SSDs used for acceleration. Some specific requirements for various maintenance activities are described below.

Requirements for Specific Maintenance Activities

Maintenance Activity	Software Requirements
Operating System Security Patch	Nothing needs to be done. Patches can be applied while the software is running.
Update kernel version	Stop acceleration for all volumes, verify the kernel, then re-run the installer for the most recent version of the software.
Replace a failed/degraded SSD in a RAID 1 LSSD that is still running in WB mode, ("unmonitored").	Simply replace the device. An unmonitored RAID 1 configuration will not shut down when one of the SSDs is degraded. When the new SSD is installed, the OS will transparently add it to the RAID 1, and the accelerator will continue to access the LSSD as it had before.
Replace a failed/degraded SSD in a RAID 1 LSSD when the software has switched from WB mode to bypass mode ("monitored").	If the software switched to bypass mode and flushed data due to the degradation of one SSD in RAID 1, first replace the failed SSD. Once the new SSD has been installed and RAID 1 degradation is no longer detected, the accelerator resumes operation automatically, though the admin may have to manually restart acceleration for the volumes.

Maintenance Activity	Software Requirements
Replace SSD or add SSD to existing LSSD (e.g. to RAID0 config.)	Changing the accelerator's SSD requires the data be flushed, and acceleration for all volumes stopped. A new accelerator should be configured, based on the LSSD containing the new SSD.
Upgrade SSD Firmware	When making any significant change to the SSD, it is recommended the operation be treated in the same manner as replacement of the SSD. The data should be flushed and the software switched to bypass mode. A new accelerator should be configured, based on the LSSD containing the upgraded SSD.
Upgrade the JetStream Accelerate software	In a small number of cases, stopping acceleration and flushing dirty data may be required before performing an upgrade; however, in most cases it is not required and an upgrade can be performed on a live system. The requirements vary depending on which version is currently installed, and the functionality of the new version. In all cases a system reboot is required after a new version is installed.

3.5. Saving Configuration Information

When an uninstall/install process is required, the administrator can save the accelerator's configuration settings (what volumes are accelerated, by what accelerators, and in what mode) and conveniently re-apply the settings to the newly installed software. The settings can also be exported as a file for use at a later time or to apply the configuration information to other servers.

When multiple servers are deployed with identical configurations (their available storage volumes are identically identified), the configuration information of one server may be applied to configure the software on the other servers.

3.6. Support for 512n, 512e and 4Kn Formatted SSDs and HDDs

JetStream Accelerate works with SSDs with 512-byte formatting, as well as 4K formatted devices. Regardless of the formatting of the SSDs, the back-end storage may be made up of 512n, 512e or 4Kn HDDs, where applicable.

3.7. Linux CLI

The JetStream Accelerate software for Linux is administered entirely through the command line interface. No GUI is needed.

Appendix: Primary Configuration Parameters

JetStream Accelerate has many tunable parameters, but these are not generally accessible to end users. Only a few parameters will be of interest to support personnel, and only in exceptional cases.

To understand the context in which these parameters might be used, it is extremely helpful to understand the behavior of the accelerator, specifically how it employs the SSD resource as a logical circular buffer and manages flushing of dirty data to storage. In general, these parameters are rarely, if ever, changed, and they should only be changed under the direction of a member of the JetStream engineering team after a careful analysis of the customer's workload and environment.

In the definitions below, the default value is shown. The variable CHANGE=YES|NO indicates whether a parameter can be changed on the fly, or if a system reboot is required for the new parameter value to take effect.

Summary List of Parameters

`FlusherCmdsFlushOut=32`
When a user sends a "stop acceleration" command for a volume accelerated in write-back mode, the accelerator starts flushing dirty data from the SSD to the target data volume. This parameter limits the number of writes that can be flushed concurrently to the target volume. Increasing the default may reduce the time required to flush the data completely, but it may not help if the back-end storage cannot support the increased activity.

`FlusherFreeAndCleanGoalPercent=10`
As the flush point progresses through the SSD ahead of the data write point, it flushes dirty data to create a section of the SSD that contains clean data only. Increasing this parameter setting increases the amount of "readily claimable" space containing clean data only (calculated as a percentage of the total SSD capacity), and correspondingly reduces the amount of space containing clean and dirty data. Increasing this setting will also force the accelerator to do more aggressive flushing, reducing the benefits of write cancellation and write coalescing.

`IoSpaceThreshholdIncrementMB=100`
If application write intensity is too high relative to storage performance, it is possible for the flusher to move more slowly than the write point, reducing the SSD space available for new writes. In this case, the accelerator can slow down application writes to the SSD to enable the flusher to progress relatively faster. If the available space falls 100MB short of the target size, then some throttling is applied to processing application writes; if the available space further decreases to 200MB short of the target size, then even more throttling is applied to processing application writes, and so on. This parameter defines the degree of granularity by which this "write throttling" is invoked.

BypassLengthKB=256
Relative to HDD, the accelerator provides the greatest benefit for write requests comprising small, random I/O patterns. If a write operation is for a large amount of data, it is written directly to HDD; preventing over-utilization of SSD space for data that should be written directly to disk. This parameter defines the size of a data block that is "not too large" to be written to the SSD. Changing the default value of this parameter may change application performance, depending on the workload and amount of SSD resources available.

Note: In AppDynamics deployments, this parameter is recommended to be set to 64.

WriteToDirtyData=1
If a write request can be applied to data that is already resident in the SSD (dirty data) the accelerator can handle the write in one of two ways: overwriting the dirty data (write in place) or writing the data in the free space at the write point and invalidating the earlier dirty data. When this parameter value is set to 1, "write in place" is enabled, resulting in better SSD space utilization, but potentially executing more writes to the SSD per write request. When the parameter value is set to 0, the accelerator will always write new data to the clean space, which utilizes SSD space less efficiently, but enables better coalescing of write requests into fewer SSD writes.

SlotsMax=120000
The accelerator maintains a metadata map of all current disk regions. The map is stored on the SSD, and the most active portion of the map is swapped into RAM. This parameter sets the maximum size of the RAM-resident portion of the map, to ensure optimal utilization of system memory. This parameter setting can be increased if the runtime statistics indicate a high percentage of metadata swap-in/swap-out operations.

AggressiveCachePopulation=66
The accelerator applies various rules when deciding which I/Os should be accelerated and which I/Os should bypass the accelerator and go directly to back-end storage. These rules are controlled by the previously described parameters RcRepeatCnt, RcMaxLengthKB, BypassWriteLengthKB, and SeqWriteEnabled. The parameter values are selected to optimize performance for on-going use in most common production environments. However, when the accelerator is initially configured and the SSD is still empty, it is often desirable to speed up data population even before information about the most active (hot) data has been collected. The accelerator applies 'aggressive' data population (by temporarily relaxing the rules) until a certain percentage of the SSD is filled with data. The AggressiveCachePopulation parameter defines this percentage.

Appendix: Recommended Parameter Settings in AppDynamics Deployments

In an AppDynamics environment, the primary purpose of the accelerator is to optimize write performance by directing virtually all write activity to the accelerator, with the exception of the occasional large write that should go to the underlying storage. In addition to the **`BypassLengthKB`** parameter mentioned earlier, four additional parameters can be adjusted from their default values to focus the accelerator's operation on these write operations. The parameters and their recommended settings are shown below:

```
SsdWriteGateMax=256
SsdWriteMax=256
SsdReadGateMax=128
SsdReadMax=64
```

A basic setup script may be used to set these five parameters immediately upon installation of the software.

JS-0418-GRD-028-1