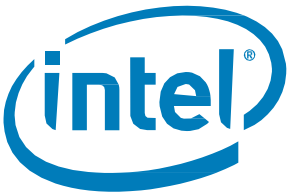
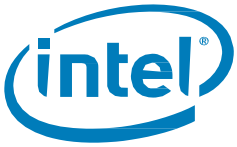
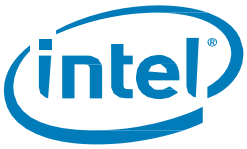


Intel® Solid-State Drives in Server Storage Applications

White Paper
February 2014

Order Number: 330037-001US





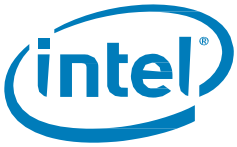
Revision History

Document Number	Revision Number	Description	Revision Date
330037	001	• Initial release.	February 2014

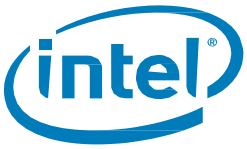
Related Documentation

Document	Document No.
Monitoring Media Wearout Levels of Intel® Solid-State Drives	325551
Partition Alignment of Intel SSDs for Archieving Maximum Performance and Endurance	330105

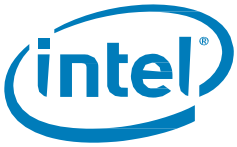
§



Contents



Figures



1 Introduction

This guide provides an overview of the performance characteristics of Intel® Solid-State Drives (SSDs) DC S3500 and DC S3700 Series and provides recommendations for optimizing performance in server storage applications.

This document is intended for designers and administrators of server systems using Intel® SSDs.

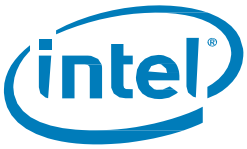
1.1 Overview

NAND flash memory-based SSDs are a revolutionary development in server storage technologies. Benefits of using Intel SATA SSDs over a rotational hard disk drives (HDDs) include:

- Up to 375X higher random Input/Output Operations per Second (IOPS)
- Up to 100X lower latencies
- Up to 400X higher vibration and shock tolerance
- Lower failure rates due to no mechanical components
- Lower power consumption
- Less heat generated

As a result, Intel SSDs accelerate existing applications, enable a more efficient storage tier definition, replace expensive Storage Area Networks (SANs) with SSD direct-attached storage, and create the possibility of new applications not feasible with HDD based systems.

In many applications, substantial benefits are gained by simply replacing HDDs with SSDs; additional changes to hardware or software are not always required. However, optimizing software and hardware settings can ensure optimal performance of SSDs in enterprise applications. This document explains the benefits of using SSDs, the important differences in using SSDs instead of HDDs, and useful tools and utilities available. This document provides a basic understanding of Enterprise class SSDs and explains where to find more information on more complex uses.



2 Understanding Intel® SSD Performance Characteristics

To take full advantage of the benefits of the Intel SSD, it is important to understand SSD performance characteristics and how they differ from hard disk drive (HDD) performance characteristics.

This section describes these performance characteristics of the Intel SSD:

- Importance of Aligning Partitions
- Random Write Bandwidth of an Empty SSD
- Write Performance Depends on Existing Fragmentation
- Random Performance Depends on IO Transfer Size
- Importance of Queuing for Read Performance
- End-to-End Data Protection
- Power Loss Detection
- Using Trim in Server Applications

2.1 Importance of Aligning Partitions

In a traditional HDD, a 512 Byte sector size has been the standard layout and operating systems align their partitions with these sectors. However, most new HDDs and Intel SSDs perform better with a 4096 Byte (4KB) alignment. Many newer operating systems will align properly when installed on a clean SSD, but some do not. Also, some software RAID does not align properly and most systems cloned from an HDD will not align properly on an SSD. When partitions are misaligned, a write from the host can cause two writes to the drive. This requires twice the amount of time to write, resulting in poor performance and double the wear on the SSD. See Figure 2–1 and Figure 2–2 for visual representation.

Figure 2-1: Partition Alignment

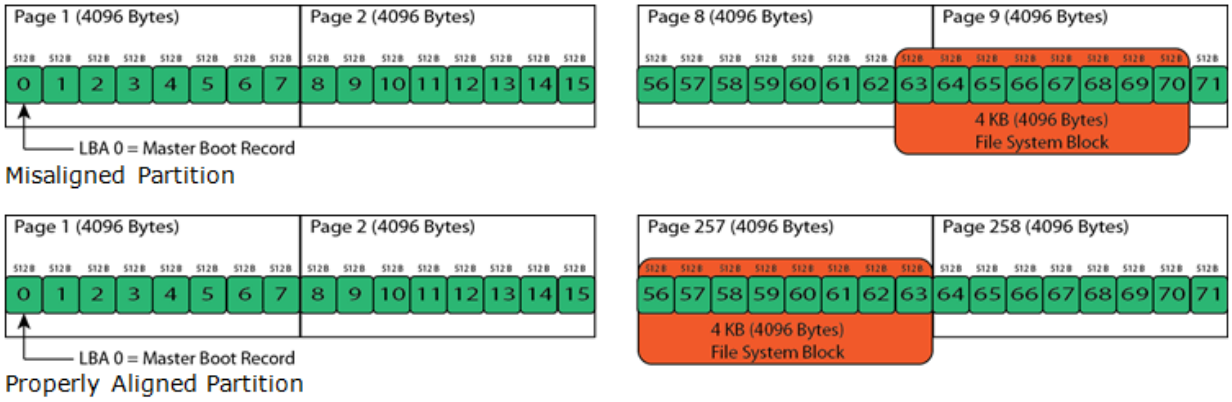
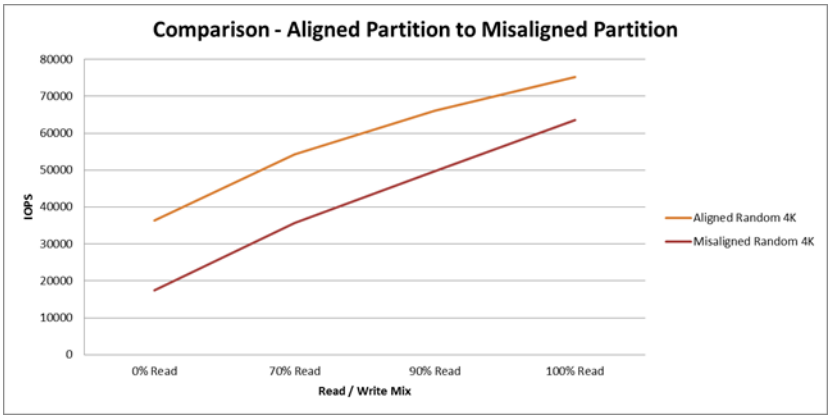


Figure 2–2 shows the performance difference between a misaligned partition and a properly aligned partition.

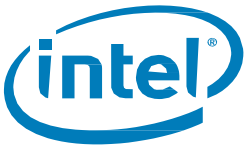
Figure 2-2: Aligned and Misaligned Partition Comparison



It is important to verify alignment of partitions in a system before testing performance or building a production system.

For Windows* systems:

- Hit the START button and type *msinfo32.exe*
- Navigate to Components > Storage > Disks
- Identify the correct drive
- Locate Partition Starting Offset
- Divide the offset number by 4096 (4KB)
- If the result is an whole number (no decimal), the partition is aligned correctly
- Repeat for each partition



For Linux systems:

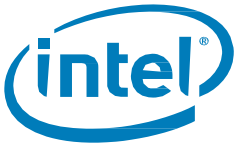
- Run *parted*
- Enter *unit s*
- Enter *print all*
- Each disk will be listed with its partitions, showing start point and end point in sectors
- The header for each drive should show Sector size (logical/physical): 512B/512B
- Multiply the Start number by the Units number. For example, 63 * 512 = 32256
- Divide the result by 4096, such as 32256 / 4096 = 7.875
- If the result is NOT a whole number (as shown above), the partition is NOT aligned on a 4KB boundary
- Repeat this for each partition listed

There is a great deal of information on the internet explaining how to correct misaligned SSD partitions. The recommendation however, unless it is absolutely necessary to repair a system, is to reload the operating system and verify the alignment of partitions as the system is building. For information on repairing alignment problems see the Intel document "*Alignment of Intel® SSDs for Achieving Maximum Performance and Endurance*".

2.2 Random Write Bandwidth of an Empty SSD

Random write bandwidth will be artificially high (above specification) when an SSD is *empty. Empty* means fresh out-of-the box or immediately following a Secure Erase operation. With an Intel SSD in the *empty* state, there is no background clean-up (garbage collection) to be performed before data can be written to it. As data is written to the drive, it will reach a level referred to as steady-state, where writes and garbage collection are appropriately balanced to measure the Intel SSD's performance. All Intel Enterprise class SSDs are benchmarked after steady-state is reached, therefore giving the most accurate and consistent results.

The way in which the drives are written to is a primary difference between HDDs and SSDs. Data can be over-written to an HDD at any time by just changing the magnetic information on the platter. With an SSD, information cannot be overwritten because SSDs are made up of NAND flash memory. NAND memory is arranged into pages; the pages are arranged into blocks. Data can only be written to a page that is empty (or newly erased). When the drive is new, all the pages are empty and therefore can be written quickly. As most or all of the pages are written to the drive becomes full, therefore a block must be erased to make space for new data to be written. Erases can only occur in blocks, not individual pages. To make the erasing and moving of data possible, SSDs have extra NAND that is not calculated into the advertised capacity of the drive. This amount of extra NAND varies by drive model. The extra NAND, or spare area, is necessary so the drive can perform writes, even when the drive is full of data. For illustration purposes, imagine the extra NAND as an empty square in a sliding block puzzle; without the extra space, none of the puzzle pieces would be able to move.



2.3 Write Performance Depends on Existing Fragmentation

As an enterprise class SSD the Intel® SSD DC S3700 Series contains higher levels of spare NAND to improve consistency and performance—two areas of high importance to the enterprise user.

Different write workloads create different levels of fragmentation of the physical storage media, whether HDD or SSD. In HDDs, fragmentation affects the performance of reads, but in SSDs, fragmentation will affect write operations. It is important to note that once a typical datacenter workload reaches stabilization, the Intel SSD will show very consistent performance until that workload changes significantly.

If a random write creates a major size discrepancy—a small amount of data being written to a larger Logical Block Address (LBA) target—higher levels of fragmentation will occur. However, a sequential write workload creates very little fragmentation. The negative consequences of fragmentation can be minimized by using a drive with more spare area, such as the Intel SSD DC S3700 Series, which is optimized for write-heavy environments.

If a random write workload is applied to an SSD that was filled sequentially, there will be a period of time where the SSD is out of steady-state and will show anomalous performance, possibly above specification. Conversely, if the SSD is subjected to a random write workload before running a sequential write workload, the sequential performance will be skewed lower for a period of time.

This same effect will occur when changing from larger to smaller transfer sizes in random workloads.

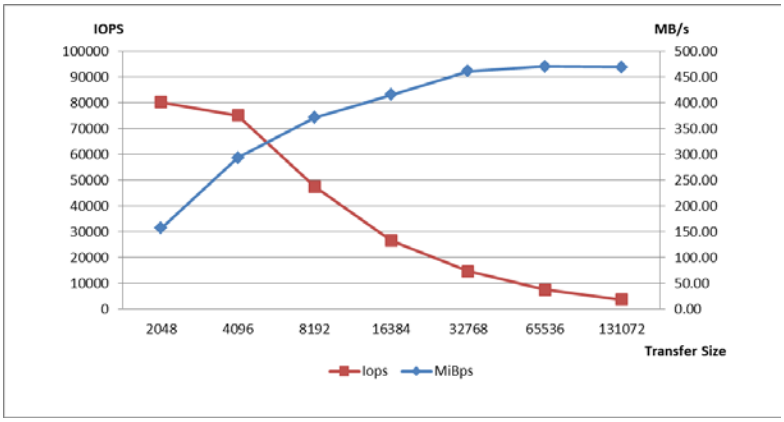
When benchmarking performance metrics with a new workload, write an amount of data equal to 1.5 to 2 times the capacity of the SSD to alleviate these anomalies.

2.4 Random Performance Depends on IO Transfer Size

Many applications use configurable transfer sizes for accessing storage. For example, some database applications allow block size changes, which enables performance tuning of the SSDs.

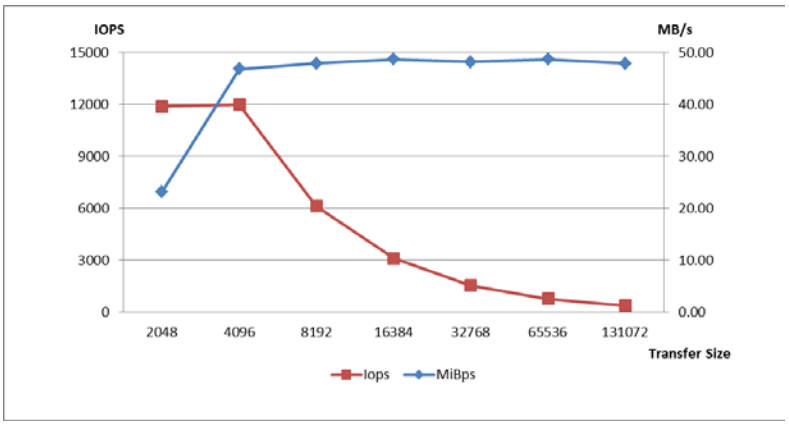
In general, an SSD's random read/write IOPS performance is better if smaller transfer sizes are used (recommended minimum size of 4KB). Throughput performance (MB/s) is better with larger transfer sizes (above 32KB). Both of these statements are supported by Figures 2–3 and 2–4. The performance of the Intel SSD DC S3500, with regard to throughput, saturates on random reads when using 32KB transfer sizes and larger. The performance on random writes saturates using 4KB transfer sizes and larger.

Figure 2-3: Random Read Performance: Intel SSD DC S3500 Series



*Transfer size with 800GB at queue depth of 32

Figure 2-4: Random Write Performance: Intel® SSD DC3500 Series



*Transfer size with 800GB at queue depth of 32

2.5 Importance of Queuing for Read Performance

Intel® SSDs implement an architecture that dispatches work to multiple NAND die across multiple channels simultaneously. Issuing queued IO in an application will allow the system to take advantage of this architecture, maximizing storage bandwidth, especially with smaller read block sizes.

Latency is directly proportional to queue depth; the higher the queue depth, the higher the latency. Setting a greater queue depth during performance testing will result in better IOPS and throughput performance, but may result in unacceptable latency in IO completion. A balance must be achieved between increased performance and increased latency.

The Intel SATA SSD supports a maximum queue depth of 32 per drive. Depending on the transfer size, increasing IO queue depth above a certain number diminishes the increase of performance and causes a sharp increase in latency.

Queuing *write* operations has less impact on performance because the Intel® SSD always has power-protected write cache enabled. Power-protected write caching enables the host to immediately acknowledge the write IO completion, and queues the write IOs internally.

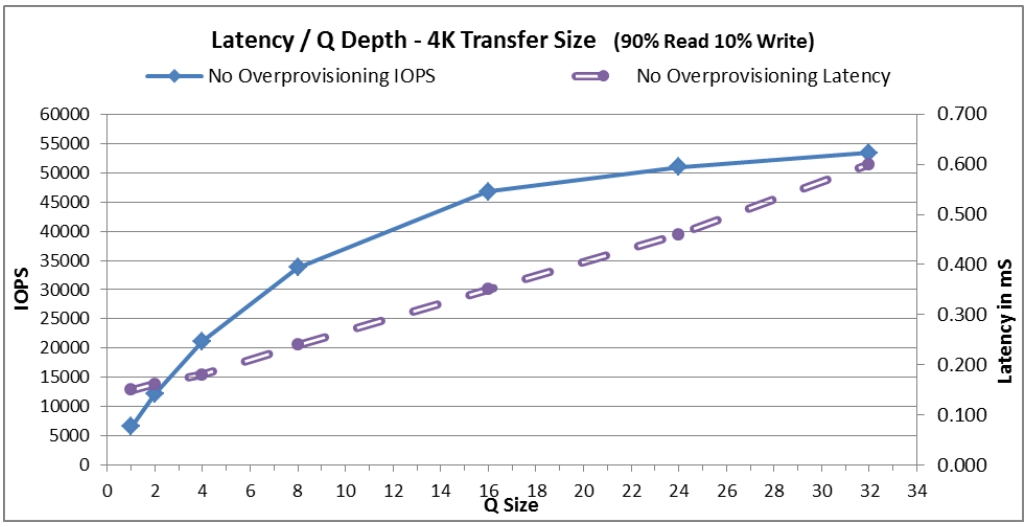
At the software level, IO queuing is accomplished using asynchronous IO or multiple threads.

If using a native SATA controller, such as Intel ICH10, the controller must be configured to use AHCI mode, which enables Native Command Queuing (NCQ) support. Usually this can be done in the system BIOS. Embedded software RAID modes generally support NCQ.

SAS and SATA host bus adapters (HBAs) and RAID controllers generally have NCQ enabled by default. If there is an option for enabling or disabling NCQ, it should be enabled.

Figure 2–5 illustrates the performance curves (IOPS and Latency) generated by an Intel SSD DC S3500 800GB drive with varying queue depth in a mixed workload.

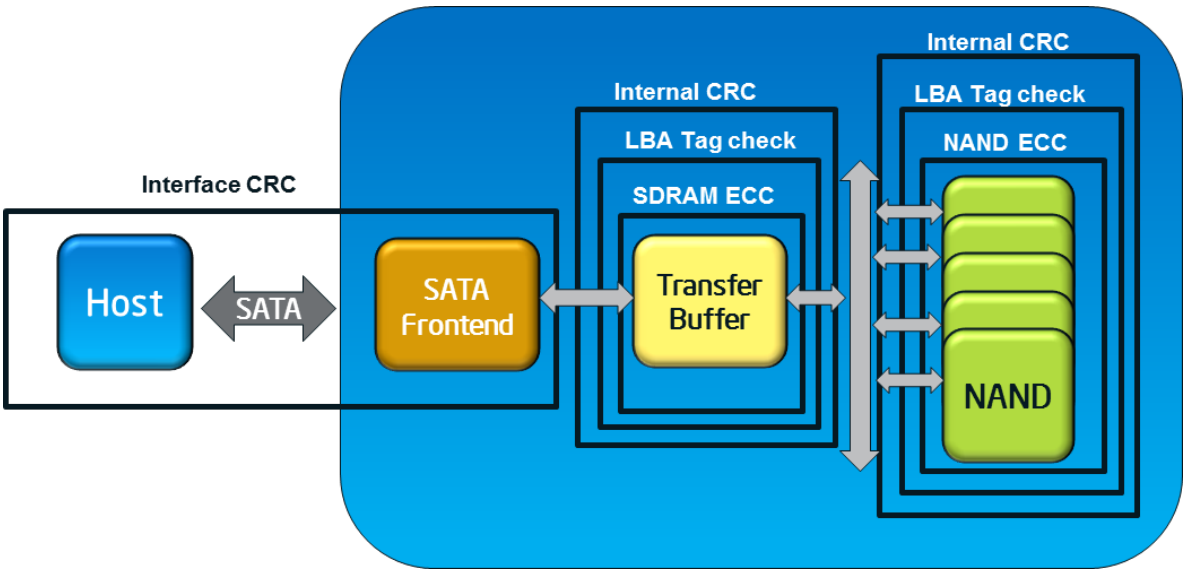
Figure 2-5: Intel® DC S3500 800 GB SSD Showing Latency versus Queue Depth - 4K Transfers



2.6 End to End Date Protection

The Intel® SSD DC S3700 and S3500 Series' end-to-end data protection scheme delivers robust data integrity by protecting against possible corruption of in-transit and stored data through every element of the SSD—including NAND, SRAM and DRAM memory—starting when the data enters the drive until the point it leaves. End-to-end data protection uses several techniques such as parity checks, Cyclic Redundancy Checks (CRC) and Logical Block Address (LBA) tag checks along the data path. Once an error is detected, the SSD immediately attempts to correct it; any uncorrectable error is reported to the host. Figure 2–6 shows how each stage of the data path and non-data path are protected using the end-to-end data protection scheme.

Figure 2-6: End-to-End Data Protection



2.7 Power Loss Detection

Intel® Enterprise class SSDs contain a power loss detection circuit called PLI (Power Loss Imminent). This PLI circuit monitors the power supply from the host. When the voltage drops, the PLI circuit switches to pull power from large backup capacitors contained within the SSD. These capacitors are able to power the drive long enough to allow any data in transition in the SSD to be written to the NAND correctly, thereby preventing information loss.

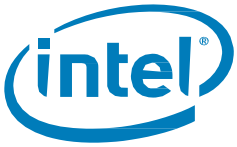
2.8 Using Trim in Server Applications

The ATA Trim command allows proactively marking NAND blocks that contain user data—but which are no longer used—as invalid. This allows the SSD to be more efficient by eliminating the need for moving obsolete data during internal garbage collection activity. This approach improves write performance after large amounts of data is discarded.

TRIM is supported and enabled in Windows* 7 and Server 2008* R2 (and newer). TRIM support was also added to the Linux kernel as of version 3.7, but only in the ext4, btrfs, xfs and jfs file systems. TRIM can be enabled in the *fstab* file using the “discard” tag on each partition of the SSD.

```
/dev/sda1  /       ext4   defaults,noatime,discard   0  1
/dev/sda2  /home   ext4   defaults,noatime,discard   0  2
```

It is best to test the use of TRIM in your environment before wide adoption. TRIM does not always improve performance in enterprise applications due to the possibility of increasing latency. Intel SSD DC S3700 Series drives contain more spare area than many other SSDs. This increased spare area gives the SSD more room to move blocks of data as it performs its tasks, thereby diminishing the benefit of using TRIM.



2.9 Using Secure Erase

For erasing all data from a drive, you can use ATA Security Erase Unit command. The Security Erase Unit command returns the SSD to its cleanest out-of-the-box state by wiping out all fragmentation and physically erasing all NAND blocks. Security Erase also resets the AES Encryption key on the media, therefore making any access to previous data impossible. It is recommended to perform a Security Erase when redeploying drives or before discarding drives. It is also a good idea to perform an erase when a drive will be used for a new function or when the workload on that drive will change significantly.

To perform a Security Erase, you can download the Intel SSD Toolbox from www.intel.com/go/ssd, listed under Tools. It is also possible to perform a Security Erase from Linux using *hdparm*, as follows:

```
hdparm --user-master master --security-set-pass password /dev/sdX
hdparm --user-master master --security-erase password /dev/sdX
hdparm -Np468862128 /dev/sdX
<Power cycle the drive>
```

3 Optimizing Intel® SSD Performance and Endurance

This section describes how usable capacity impacts random write performance and endurance, and provides recommendations for optimizing Intel® SSD usable capacity for server storage environments.

- Adjusting Usable Capacity
- Using Direct IO in Linux
- Selecting IO Scheduler in Linux
- Optimizing RAID Configurations

3.1 Adjusting Usable Capacity – Over-Provisioning

A small increase in an SSD's spare area can provide a large increase in random write performance and endurance.

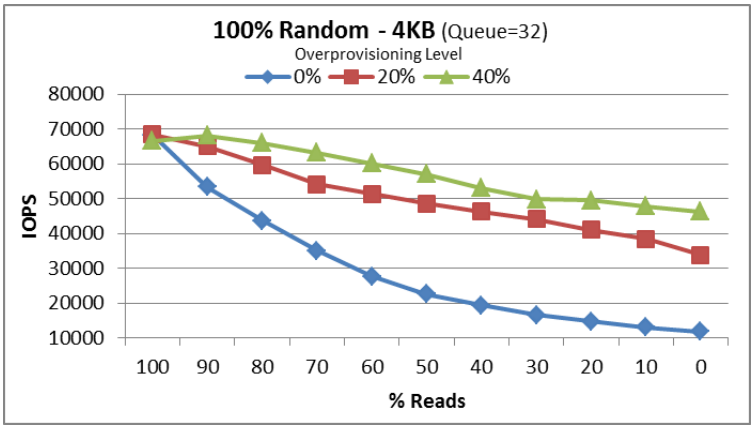
All Intel SSDs have more NAND capacity than what is available for user data. This additional NAND capacity is reserved for internal operations. The larger the spare area, the more efficiently the SSD can perform random write operations and the better the random write performance.

For better random write performance and endurance, the spare area can be increased by reducing the usable capacity of the drive; this process is called *over-provisioning*.

Figure 3–1 shows an illustration of how random write performance increases with over-provisioning.

Note: Over-provisioning has no impact on pure read or sequential write performance.

Figure 3-1: Example of random performance with different levels of over-provisioning on the Intel® SSD DC S3500 Series – 800GB Drive



3.1.1 How to Adjust Usable Capacity

You can reduce the usable capacity of, or over-provision, a drive several ways:

Important: The SSD must be new, fresh out-of-the-box or must be erased using the ATA SECURITY ERASE UNIT command immediately before adjusting the usable capacity.

- (Recommended) Create partition(s) that occupy only the desired usable capacity and leave the remaining capacity unused.
- If using a RAID controller or Software RAID, create *virtual drive(s)* that occupy only the desired usable capacity and leave the remaining capacity unused.
- (Advanced Users) Use the ATA standard SET MAX ADDRESS command to limit the user accessible capacity of an SSD. This feature, also referred to as Host Protected Area (HPA) in Linux*, can also be modified using the hdparm* utility with -Np command, or the HDAT2* utility under DOS. Note that this method will change the drive capacity reported to the user through various utilities.

3.2 Using Direct IO in Linux

With SSDs in Linux* using direct IO instead of buffered IO is recommended, when possible. The Linux IO subsystem provides read and write buffering at the block device level. In most cases, buffering is undesirable with SSDs for the following reasons:

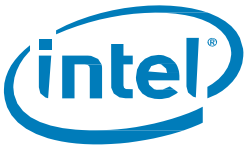
- SSDs have lower latencies than HDDs, therefore the benefits of buffering are reduced.
- Buffering read IOs consumes extra CPU cycles for memory copy operations. At IO rates typical for SSDs, this extra CPU consumption may be high and create a read performance bottleneck.

To use direct IO and bypass the buffer, software can set O_DIRECT flag when opening a file. Many applications and test tools have configurable options that allow selecting direct or buffered IO, for example:

- FIO* utility: use ‘--direct=1’ option
- MySQL InnoDB*: use ‘--innodb_flush_method=O_DIRECT’ option
- Oracle* 10g/11g: use ‘filesystemio_options = SETALL’ or ‘filesystemio_options = DIRECTIO’

3.3 Selecting IO Scheduler in Linux*

IO schedulers in Linux* determine how competing IOs from different processes are prioritized and balanced. With SSDs, it is common to recommend the use of the simplest *noop* (no operation) scheduler, however, *deadline* can also work well. Many Linux* distributions use the CFQ scheduler as the default, which does have SSD support built in.



Test different schedulers in your application to determine which is most appropriate.

The “scheduler” command displays available and active schedulers as follows:

```
cat /sys/block/sdX/queue/scheduler
```

The “iosched” command displays available parameters for the currently active scheduler as follows:

```
cat /sys/block/sdX/queue/iosched
```

The “echo noop” command sets *noop* scheduler for the current session as follows:

```
echo noop > /sys/block/sdX/queue/scheduler
```

To permanently change your scheduler, please refer to your Linux Distribution’s documentation.

3.4 Optimizing RAID Configurations

3.4.1 Enabling SSD-Specific Features

Some RAID controllers may have SSD-specific features that can be enabled for best performance with SSDs. For example, Intel® RAID Controller has the optional FastPath* IO feature that boosts the random IO processing capability of the controller.

3.4.2 Selecting RAID Level

With SSDs, RAID level selection criteria are similar to HDDs. However, because of shorter rebuild times under load and lower performance cost of reads during Read-Modify-Write operations, RAID 5/6/50/60 may be more effective in some applications that traditionally would require the use of RAID 10 with HDDs.

3.4.3 Setting RAID Stripe Unit Size

To optimize the speed of random IOPS, stripe unit size should be at least 2X of the typical transfer size used by the application. This minimizes the frequency of transfers crossing the stripe boundaries and writing to more than one drive at a time. If the files and transfers are aligned on the stripe boundaries, the stripe size can be set to be equal to the typical transfer size.

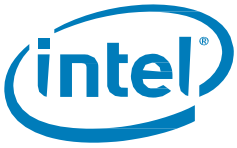
In large RAID sets, you must verify that your stripe size is large enough so that the stripe per drive is not less than 4KB. To calculate the smallest stripe:

4096 X Number_of_Striped_Drives = Smallest_Stripe_Size

For higher sequential bandwidth with applications that cannot use IO queuing, the following stripe unit size or smaller should be used:

0.5 x Transfer_size / Number_of_striped_drives

This allows each transfer to be split between all drives in the stripe to maximize total bandwidth.



3.4.4 Using RAID Write-Back Caching

Write-back caching at the controller level can reduce write latencies. This is especially true with RAID 5/50/6/60 as RAID parity calculations introduce additional latency.

Also, write-back caching helps with merging smaller sequential write transfers into larger write transfers. Size of the resulting larger transfers equals the write-back cache element size, which typically equals stripe unit size. Merging write transfers is important when there are concurrent sequential write threads with small transfer sizes (for example, database transaction logs).

Using a combination of the following techniques can help obtain optimal performance:

- Enable write-back caching on virtual drives used for latency-sensitive writes or small sequential transfer writes, such as database transaction logs.
- Disable write-back caching on virtual drives used for large amounts of writes that are not latency-sensitive.
- For different types of workloads, create separate virtual drives across the same set of SSDs with the write-back caching enabled or disabled (at the virtual drive level) according to the particular workload requirements.
- For different types of workloads, use separate SSDs.
- For virtual drives with write-back caching enabled, limit write rates from non-critical processes to avoid filling the cache.
- Use larger count of read threads or higher read queue depths for counter-balancing writes. The maximum amount of outstanding read IOs should scale in proportion to the number of drives in the RAID array.

Enabling write-back caching on the RAID controller usually requires either a battery backup unit (BBU) or super-capacitor and onboard flash memory for protecting the cache during power loss.

Also, it is important to remember that RAID BBU or super-capacitors by themselves do not protect the write cache on the drives; therefore, the general practice is to disable the drive cache. However, the Intel SSDs for datacenter users have embedded capacitors that protect the entire data path.

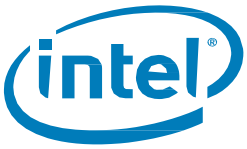
3.4.5 Disabling RAID Read Ahead

Typically with SSDs, read-ahead should be disabled. Read-ahead may increase random read latencies and may create a read bandwidth bottleneck. Read-ahead also consumes RAID memory that could improve efficiency if left available for use by write-back caching.

However, when measuring performance, remember that disabling read-ahead may reduce sequential read bandwidth with single threaded applications (for example, with Linux* DD command or when copying files) and may create the impression of limited scalability. If sequential read bandwidth with a single threaded application is important, either enable read-ahead or set the read transfer size to *2 x Stripe_unit_size x Number_of_striped_drives*.

3.4.6 Disabling RAID Read Caching

Typically with SSDs, read caching should be disabled. Enabling read caching may create a read bandwidth bottleneck due to extra processing and RAID memory access overhead. Also, read caching consumes RAID memory that could be used by write-back caching.



4 Measuring Intel® SSD Performance

This section describes how to measure performance of an Intel® SSD:

- Defining Workloads and Performance Metrics
- Conditioning the SSD Before Measuring Performance
- Selecting Performance Measurement Tools

4.1 Defining Workloads and Performance Metrics

Whenever possible, Intel recommends using the actual target application for evaluating and tuning performance of Intel SSDs.

When using synthetic tests, the following workload parameters should be considered:

- Sequential or random IO
- IO transfer size
- Read, write, or mixed IO operations
- IO queue depth and the number of parallel threads (worker processes)

Refer to "*Accelerating Data Center Workloads with Solid-State Drives"* on Intel.com for a description of the methodology used to replicate a workload and validate that replication.

4.2 Conditioning the SSD Before Measuring Performance

Before taking write or mixed workload performance measurements of an Intel SSD, make sure the SSD is in a *steady state*. This process to prepare the SSD for performance measurements is known as *conditioning*.

Please note: These instructions are for use with Intel Enterprise class SSDs. Other SSDs may not show correct results using these steps.

To condition the Intel SSD:

1. Perform a Security Erase Unit on the drive.
2. Write the full capacity of the SSD sequentially.

   This step ensures all LBAs have valid data and removes fragmentations.

For example, in Linux* use the following command:

```
dd if=/dev/zero of=/dev/sdX bs=256k oflag=direct
```

3. Write the amount equal to 1.5 – 2 times the capacity of the SSD using the target workload.

   This step creates fragmentation typical to the target workload. It may take several hours to complete. Reads can be removed from this step to speed up the process, as reads have no impact on fragmentation.

4. Measure performance using the target workload.

   In this step you can use different IO queue depths and read/write ratios.

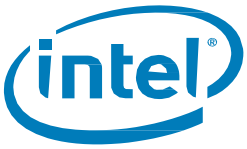
4.3 Selecting Performance Measurement Tools

Examples of common measurement tools include Iometer* in Windows* and FIO* in Linux*. Intel recommends using a performance measurement tool that allows customization of all workload parameters:

- Sequential or random IO
- IO transfer size
- Read, write, or mixed IO operations
- IO queue depth or the number of parallel threads (worker processes)

Additionally, once you select a performance measurement tool, make sure:

- IO queuing is configured correctly
  - In Iometer*, set the desired queue depth (on the Disk Targets tab, set the *# of Outstanding IOs* setting). By default, this is set to 1, which means no queuing used.
  - In Linux* using Asynchronous IO requires installation of the LIBAIO library. Some software (for example, FIO*) may also require LIBAIODEV rpm file to be installed before the software is compiled.
  - Native SATA controllers (for example, Intel ICH10) must be configured to use AHCI mode. Usually this is accomplished in the system BIOS. Additionally, various software RAID modes usually support queuing.
- In striped configurations, the IO queue depth may need to be multiplied by the number of drives in the array.
- Transfers are aligned on block boundaries equal to the transfer size.
- In Linux* use direct IO when possible. See Section 3.2, “Using Direct IO in Linux", for details.



4.3.1 Example Performance Measurement Using FIO*

The script below provides an example of using the FIO* utility for measuring random mixed (read+write) 16KB workload on a 800GB Intel® SSD DC S3500 Series with two different read/write ratios and queue depths.

```
# First filling the drive sequentially
dd if=/dev/zero of=/dev/sdX bs=256k oflag=direct

# Conditioning the drive with 800GB of writes before taking measurements
fio --name=myjob --filename=/dev/sdX \
    --ioengine=libaio --direct=1 --norandommap --randrepeat=0 \
    --blocksize=16k --rw=randrw --rwmixwrite=100 --iodepth=32 \
    --size=800G

# Measuring for 20 minutes with read/write = 70/30 and queue depth = 4
fio --output=results.30w.qd4.txt --name=myjob --filename=/dev/sdX \
    --ioengine=libaio --direct=1 --norandommap --randrepeat=0 \
    --blocksize=16k --rw=randrw --rwmixwrite=30 --iodepth=4 \
    --time_based --run_time=1200

# Measuring for 20 minutes with read/write = 30/70 and queue depth = 16
fio --output=results.70w.qd16.txt --name=myjob --filename=/dev/sdX \
    --ioengine=libaio --direct=1 --norandommap --randrepeat=0 \
    --blocksize=16k --rw=randrw --rwmixwrite=70 --iodepth=16 \
    --time_based --run_time=1200
```

5 Estimating Lifetime and Monitoring Wear

This section describes the Intel® SSD endurance characteristics and explains how to monitor wear and estimate drive lifetime.

- Intel SSD Endurance Characteristics
- Monitoring Wear Using SMART
- Estimating Drive Lifetime

5.1 Intel® SSD Endurance Characteristics

The Intel SSD is built with MLC Intel® NAND Flash Memory and has approximately 450TB Written for standard endurance and 14PB Written for high-endurance rated in accordance with JEDEC standard JESD218.

SSD lifetime requirements in server applications usually range between 2 to 10 years. The Intel SSD can meet or exceed these lifetime requirements in many applications, while providing cost benefits of MLC NAND. However, due to limited MLC NAND cycling capability, the target workload assessment and lifetime estimation is required to confirm whether the SSD is suitable for the application. Also, monitoring Self-Monitoring, Analysis, and Reporting Technology (SMART) attributes that report the state of the NAND wear is recommended in production environments.

The Intel SSD has a write-leveling mechanism that ensures that NAND wear is distributed evenly across the entire NAND capacity, even for workloads that have regions of high write activity mixed with regions of static data. As a result of write leveling, the difference between maximum and average program/erase cycles for different blocks of NAND is not significant for purposes of monitoring wear and estimating SSD lifetime.

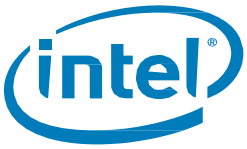
The following formulas describe how the amount of NAND wear relates to the amount of host writes:

Write_Amplification_Factor = NAND_Writes_GB / Host_Writes_GB

Consumed_PE_Cycles = (Host_Writes_GB x Write_Amplification_Factor) / Raw_NAND_Capacity_GB

Host_Write_Endurance_GB = (Raw_NAND_Capacity_GB x Max_PE_Cycles) / Write_Amplification_Factor

Write Amplification Factor (WAF) depends on workload and drive configuration. For 100% sequential write workloads, the Write Amplification Factor is roughly 1. However, for workloads that have substantial amounts of random writes, the Write Amplification Factor is higher than 1 because of defragmentation overhead.



The following factors affect defragmentation and WAF:

- Randomness – fewer random writes have lower WAF
- Locality – localized writes (for example, actively writing to a 20GB range only) have lower WAF
- Write transfer size – writes with larger transfer sizes have lower WAF
- Over-Provisioning – smaller usable capacity (more spare area) results in lower WAF

Other mechanisms, such as write leveling, read-disturb relocation and block retirement, also create additional writes to NAND. However, their contribution to NAND wear is insignificant for most workloads.

Reducing usable capacity of the SSD provides an effective way to reduce the Write Amplification Factor and improve drive endurance and performance. See *Section 3.1 Adjusting Usable Capacity* for details. However, selecting a drive with a higher endurance rating is preferred for applications that require it.

5.2 Monitoring Wear Using SMART

The Intel® SSD provides three SMART attributes that reflect the current state of NAND wear on the SSD:

- Timed Workload Media Wear (E2h/266)
- Media Wearout Indicator (E9h/233)
- Available Reserves (E8h/232)

To read SMART attributes in Linux* or Windows*, use the Smartmontool* utility. The 'smartctl –A' command displays all SMART attributes supported by the drive.

5.2.1 Time Workload Media Wear Attribute (E2h/226)

This attribute tracks the drive wear seen by the device during the last wear timer loop, as a percentage of the maximum rated cycles. The raw value tracks the percentage up to 3 decimal points. This value should be divided by 1024 to get the percentage. For example: if the raw value is 4450, the percentage is 4450/1024 = 4.345%. The raw value is held at FFFFh until the wear timer (attribute E4h) reaches 60 (minutes). The normalized value is always set to 100 and should be ignored.

The attribute is reset when a SMART EXECUTE OFFLINE IMMEDIATE (D4h) subcommand

40h is issued to the drive. However, if E2h is never reset, E2h and E9h attributes indicate the same wear-out: *E2h_Raw_Value / 1024 = 100 - E9h_Normalized_Value.*

5.2.2 Media Wearout Indicator Attribute (E9/233)

The Media Wearout Indicator attribute reports the number of program/erase cycles the NAND media has undergone. The normalized value declines linearly from 100 to 1 as the average program/erase cycle count increases from 0 to the maximum rated cycles. Once the normalized value reaches 1, the number will not decrease.

A value of 1 indicates that the rated maximum NAND program/erase cycles have been consumed. The SSD is likely to continue working after the value reaches 1; however, data retention time and uncorrectable bit error rate may be outside of the product specification.

5.2.3 Available Reserves Attribute (E8h/232)

The Available Reserves attribute reports the number of reserve blocks remaining. The normalized value begins at 100, which corresponds to 100% availability of the reserved space. The pre-fail threshold value for this attribute is 10.

5.3 Estimating Drive Lifetime

This section outlines the steps that Intel recommends for estimating drive lifetime of the Intel SSD. The commands shown are for a Linux* operating system; however, the same general procedure applies to other operating systems.

1. Run Security Erase on the SSD (not required if the SSD is new, out-of-the-box).

   ```
   > hdparm --user-master master --security-set-pass password /dev/sdX
   > hdparm --user-master master --security-erase password /dev/sdX
   ```

2. Configure desired usable capacity (not required if already configured).

   ```
   > hdparm –Np<MAXLBA> /dev/sdX
   <Power cycle the drive>
   ```

3. Create partitions, RAID logical drives, file system, etc. per your target application requirements.

4. Run the target workload long enough for the amount of writes per drive to reach twice the drive capacity.

   - Read SMART attribute E1h/225 (Host Write Sectors) before and after to confirm.
   - *GB_Written = (E1h_Raw_Value_End - E1h_Raw_Value_Start) / 32*

5. Read SMART attribute E2h/226 (Timed Workload Media Wear). This is the start value.

   ```
   > smartctl -A /dev/sdX
   ```

6. Run the target workload long enough for the amount of writes per drive to be at least twice the drive capacity.

   The test period should represent average expected activity level; otherwise, the final results will need to be adjusted accordingly.

7. Read SMART attribute E2h/226 (Timed Workload Media Wear). This is the end value.

   ```
   > smartctl -A /dev/sdX
   ```

8. Calculate estimated drive lifetime with this workload.

   - *Added_Wear_Percent = (E2h_Raw_Value_End - E2h_Raw_Value_Start) / 1024*
   - *Estimated_Lifetime = 100 x 1024 x Test_Time / (E2h_Raw_Value_End - E2h_Raw_Value_Start)*