Intel® Solid-State Drive 320 Series in Server Storage Applications

Reference Guide
INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to:  http://www.intel.com/design/literature.htm

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2011, Intel Corporation. All Rights Reserved.
Contents

1.0 About This Guide .................................................................................................. 5
2.0 Overview .............................................................................................................. 5
3.0 Understanding Intel® SSD 320 Series Performance Characteristics ...................... 6
   3.1 Random Write Bandwidth is Faster with an Empty SSD ..................................... 6
   3.2 Write Performance Depends on Existing Fragmentation ................................. 7
   3.3 Random Performance Depends on I/O Transfer Size ...................................... 8
   3.4 Importance of Aligning Write I/Os, not Partitions ......................................... 9
   3.5 Importance of Queuing for Read Performance ............................................. 9
   3.6 Reads Have Higher Priority Than Writes .................................................... 10
   3.7 Write Cache Is Always Enabled and Protected .......................................... 10
   3.8 Performance with Compressible and Non-Compressible Data ....................... 11
   3.9 Using Trim in Server Applications .............................................................. 11
4.0 Optimizing Intel® SSD 320 Series Performance and Endurance .......................... 12
   4.1 Adjusting Usable Capacity ............................................................................ 12
      4.1.1 Recommended Intel SSD 320 Series Usable Capacity Settings ............ 13
      4.1.2 How to Adjust Usable Capacity ............................................................ 14
   4.2 Using Direct I/O in Linux ............................................................................. 15
   4.3 Selecting I/O Scheduler in Linux .................................................................. 15
   4.4 Optimizing RAID Configurations .................................................................. 16
      4.4.1 Enabling SSD-Specific Features ........................................................... 16
      4.4.2 Selecting RAID Level ........................................................................... 16
      4.4.3 Setting RAID Stripe Unit Size ............................................................... 16
      4.4.4 Using RAID Write-Back Caching ......................................................... 16
      4.4.5 Disabling RAID Read-Ahead ............................................................... 17
      4.4.6 Disabling RAID Read Caching ............................................................. 18
      4.4.7 I/O Queuing in Striped Configurations ............................................. 18
5.0 Measuring Intel® SSD 320 Series Performance ................................................... 19
   5.1 Defining Workloads and Performance Metrics .............................................. 19
   5.2 Conditioning the SSD Before Measuring Performance .................................. 19
   5.3 Selecting Performance Measurement Tools ............................................... 20
      5.3.1 Example Performance Measurement Using FIO* ............................ 21
6.0 Estimating Lifetime and Monitoring Wear .......................................................... 22
   6.1 Intel® SSD 320 Series Endurance Characteristics ......................................... 22
   6.2 Monitoring Wear Using SMART ................................................................ 23
      6.2.1 Timed Workload Media Wear Attribute (E2h/226) ............................ 23
      6.2.2 Media Wearout Indicator Attribute (E9h/233) .................................... 24
      6.2.3 Available Reserves Attribute (E8h/232) ............................................ 24
   6.3 Estimating Drive Lifetime .............................................................................. 24
7.0 Additional Information ....................................................................................... 26
8.0 Revision History ................................................................................................. 26
1.0 About This Guide

This guide provides an overview of the performance characteristics of the Intel® Solid-State Drive 320 Series (Intel® SSD 320 Series) and provides recommendations for optimizing and measuring performance in server storage applications.

This document is intended for designers and administrators of server systems using the Intel® SSD 320 Series.

2.0 Overview

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

- Up to 100X higher random Input/Output Operations Per Second (IOPS)
- Up to 100X lower latencies
- Lower failure rates
- Higher vibration tolerance
- Lower power consumption
- Less heat generated

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

In most applications, substantial benefits can be realized by simply replacing HDDs with SSDs without making any other changes to hardware or software. However, optimizing software and hardware settings can help you realize the full potential of SSDs.
3.0 **Understanding Intel® SSD 320 Series Performance Characteristics**

To take full advantage of the benefits of the Intel SSD 320 Series, it is important to understand SSD performance characteristics and how they differ from hard disk drive (HDD) performance characteristics.

This section describes performance characteristics of the Intel SSD 320 Series:

- Random Write Bandwidth is Faster with an Empty SSD
- Write Performance Depends on Existing Fragmentation
- Random Performance Depends on I/O Transfer Size
- Importance of Aligning I/Os, not Partitions
- Importance of Queuing for Read Performance
- Reads Have Higher Priority Than Writes
- Write Cache Is Always Enabled and Protected
- Performance with Compressible and Non-Compressible Data
- Using Trim in Server Applications

3.1 **Random Write Bandwidth is Faster with an Empty SSD**

Random write bandwidth is at its highest when an SSD is empty (empty means new, out-of-the-box or immediately following a Security Erase operation).

When empty, an SSD does not have to move and re-write data previously written to the NAND. As a result, random write bandwidth – in terms of megabytes per second (MB/s) – can be almost as high as sequential write bandwidth. With 4KB blocks, this translates into 10K to 30K IOPS. However, after an amount of data is written to the SSD, random write performance may drop significantly. (Note, however, the resulting random write performance of an SSD is still many times higher than the random write performance of an HDD.)

Additionally, if you measure random write performance after filling an SSD sequentially, you can observe very high initial random write performance (IOPS) followed by a large drop in performance (see Figure 1). Note, however, after the drop, the SSD is still very fast compared to an HDD.
3.2 Write Performance Depends on Existing Fragmentation

With the Intel SSD 320 Series, different write workloads create different amounts of fragmentation of the physical NAND space. Fragmentation requires additional defragmentation work and results in slower write performance.

The smaller the random write transfer size and the larger the target Logical Block Address (LBA) range, the more fragments are created. A 100% sequential write workload creates no fragmentation.

If a random write workload is applied to an SSD that was filled sequentially, there is no defragmentation work to be done in the beginning. This results in higher initial random write performance that degrades over time with fragmentation. Conversely, if the SSD is subjected to a random write workload before running a sequential write workload, the sequential performance will be initially slower.

Switching from a larger random write transfer size to a smaller transfer size may result in the smaller transfer size workload initially being faster and then losing some speed as the amount of fragmentation increases.

For accurate performance metrics with a new workload, write an amount of data equal to the capacity of the SSD to alleviate anomalies due to defragmentation.
3.3 Random Performance Depends on I/O Transfer Size

Many applications use configurable transfer sizes for accessing storage. For example, some database applications allow changing their block sizes. This provides an opportunity for tuning performance of SSDs.

Unlike an HDD, an SSD’s random read/write performance is generally better if smaller transfer sizes are used. This is especially true for random read performance. However, using larger block sizes may be beneficial for applications that can benefit from higher bandwidth in terms of MB/s, especially for writes.

Figure 2 and Figure 3 illustrate how random read and random write performance depends on transfer size.

Note: Matching I/O transfer size and NAND page size provides no direct benefit.

**Figure 2.** Example of random read performance by transfer size with 300 GB Intel SSD 320 Series and queue depth set to 32

![Random Read Performance Graph](image)

**Figure 3.** Example of random write performance by transfer size with 300 GB Intel SSD 320 Series configured with 250 GB usable capacity

![Random Write Performance Graph](image)
3.4 Importance of Aligning Write I/Os, not Partitions

Most applications that use small (less than 64KB) write I/Os do not have those write I/Os overlapping each other. For example, typical write I/Os generated by an Online Transaction Processing (OLTP) database are aligned on block boundaries. However, in some synthetic test tools, I/Os may have no alignment by default. This can result in overlapping I/Os, extra NAND space fragmentation, and lower performance. For example, Iometer* has I/Os aligned to ‘Sector Size’ by default. In this case, the I/O alignment should be changed to the I/O transfer size.

With the Intel SSD 320 Series, aligning partitions or RAID volumes is not required and provides no performance benefit.

3.5 Importance of Queuing for Read Performance

The Intel SSD 320 Series has a parallel architecture with 10 NAND channels (five channels on the 40 GB SSD). Additionally, the 160 GB, 300 GB, and 600 GB SSDs have more than one NAND die per channel. As a result, the SSD can execute multiple operations simultaneously. Read performance can benefit most from queuing multiple operations, especially with smaller block sizes.

The Intel SSD 320 Series supports a maximum queue depth of 32 per drive. Depending on the transfer size, increasing I/O queue depth above a certain number results in diminishing increase of performance.

If multiple SSDs are used behind a RAID controller with striping, the total I/O queue depth used by an application typically equals the optimal queue depth per SSD multiplied by the number of SSDs in the array (excluding parity drives in RAID 5/50/6/60).

Queueing write operations has small impact on performance because the Intel SSD 320 Series always has power-protected write cache enabled, which allows acknowledging the write I/O completion to the host immediately and queuing the write I/Os internally. However, if using striped RAID configuration with no write-back caching at the controller level, using write queue depth not smaller than the number of drives in the stripe is recommended. See Section 4.4, Optimizing RAID Configurations for more information.

At the software level, I/O queuing is accomplished either by using asynchronous I/O or by using multiple threads.

If using a native SATA controller (for example 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 usually support NCQ.

SAS and SATA host bus adapters (HBAs) and RAID controllers usually have NCQ enabled by default. If there is an option for enabling or disabling NCQ, keep it enabled.
Figure 4 illustrates the read performance curves (IOPS) generated for a 300 GB Intel SSD 320 Series with varying data transfer size and queue depth.

**Figure 4. Example of random read performance by queue depth and transfer size of 300 GB Intel SSD 320 Series**

3.6 **Reads Have Higher Priority Than Writes**

In most applications, read latencies are more critical than write latencies. The Intel SSD 320 Series services read I/Os with higher priority than write I/Os.

If both read and write I/Os are in the SSD’s queue, the SSD will service the read I/Os first. However, to avoid potential write stalls, write I/Os that are older than 200 ms are serviced with the same priority as reads.

3.7 **Write Cache Is Always Enabled and Protected**

Enabling write caching allows substantial improvement of write latencies and write bandwidth. However, many applications are sensitive to potential loss of cached data in case of power loss. Traditionally, drive write cache is disabled in such applications resulting in slower performance.

The Intel SSD 320 Series has 256KB of shared (read and write) SRAM cache protected with capacitors that provide energy to flush cached data to the NAND in case of unexpected power loss. This allows the write cache to be enabled always, providing maximum write performance.
3.8 Performance with Compressible and Non-Compressible Data

The Intel SSD 320 Series does not use data compression at the drive-level. Drive performance does not depend on whether the data is compressible or not.

When comparing performance of the Intel SSD 320 Series with SSDs that use data compression, it is important to remember that many server applications will not benefit from data compression. For example, media content (usually compressed before storing) or encrypted data is not compressible. However, many existing performance testing utilities may use simple data patterns such as all 0s or all 1s that have the highest compression rate. In summary, performance results measured with such utilities on SSDs that use data compression may be misleading.

3.9 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 defragmentation activity. This approach improves write performance after large amounts of data are discarded.

However, if old data is discarded at the same rate as new data is written, simply overwriting the old data with new data is more efficient than using Trim.

In most server applications, reserving larger spare capacity is likely to be sufficient and the benefits of using Trim are reduced. Larger spare capacity provides performance benefits similar to Trim, but does so in a ‘static’ way that ensures better performance predictability. Allocating larger spare capacity is recommended on the Intel SSD 320 Series for server workloads that have substantial amounts of random writes. See Section 4.1, "Adjusting Usable Capacity", for more information on allocating spare capacity on the Intel SSD 320 Series.

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.
4.0 Optimizing Intel® SSD 320 Series Performance and Endurance

This section describes how usable capacity impacts random write performance and endurance, and provides recommendations for adjusting Intel SSD 320 Series usable capacity for server storage environments.

- Adjusting Usable Capacity
- Using Direct I/O in Linux
- Selecting I/O Scheduler in Linux
- Optimizing RAID Configurations

4.1 Adjusting Usable Capacity

A small reduction in an SSD’s usable capacity 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. The unused capacity is called *spare capacity*. This area is reserved for internal operations. The larger the spare capacity, the more efficiently the SSD can perform random write operations and the higher the random write performance.

On the Intel SSD 320 Series, the spare capacity reserved at the factory is 7% to 11% (depending on the SKU) of the full NAND capacity. For better random write performance and endurance, the spare capacity can be increased by reducing the usable capacity of the drive; this process is called *over-provisioning*.

Figure 5 shows an illustration of how random read/write performance increases as usable capacity decreases.

Note: Reducing usable capacity has no impact on pure read or sequential write performance.

**Figure 5. Example of random performance by usable capacity of the 300 GB Intel SSD 320 Series**
Figure 6 shows an illustration of how endurance increases as usable capacity decreases.

**Figure 6. Example of endurance by usable capacity of the 300 GB Intel SSD 320 Series**

<table>
<thead>
<tr>
<th><strong>Intel® SSD 320 Series SKU</strong></th>
<th><strong>Default factory configuration</strong></th>
<th><strong>Usable capacity = 80% of factory default</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity, GB (IDEMA)</td>
<td>Capacity, GiB (GiB=2^32 Bytes)</td>
<td>Max LBA</td>
</tr>
<tr>
<td>40GB</td>
<td>40</td>
<td>37</td>
</tr>
<tr>
<td>80GB</td>
<td>80</td>
<td>75</td>
</tr>
<tr>
<td>120GB</td>
<td>120</td>
<td>112</td>
</tr>
<tr>
<td>160GB</td>
<td>160</td>
<td>149</td>
</tr>
<tr>
<td>300GB</td>
<td>300</td>
<td>279</td>
</tr>
<tr>
<td>600GB</td>
<td>600</td>
<td>559</td>
</tr>
</tbody>
</table>
4.1.2 How to Adjust Usable Capacity

You can reduce usable capacity, and thus increase spare capacity, several ways:

- (Recommended) 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.

- 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.

- Create partition(s) that occupy only the desired usable capacity and leave the remaining capacity unused.

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

The following example shows how to use the hdparm utility to erase an SSD and set the usable capacity (Max LBA 468862128 represents 240 GB usable capacity).

Note: The first two commands (erasing the drive) are not required if the drive 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
hdparm -Np468862128 /dev/sdX
```

<Power cycle the drive>

Note: Some Linux* kernel versions (for example, Ubuntu* 10.10) will automatically unlock Host Protected Area (HPA) on the next drive power-on or reboot and will use full drive capacity. If this occurs with your kernel version, pass `libata.ignore_hpa=0` kernel parameter during boot (for example, by editing GRUB configuration). To check whether HPA is honored or gets unlocked, power-cycle the SSD or the system, execute the `hdparm -I /dev/sdX` command, and then check whether LBA48 user addressable sectors value matches the value set.
4.2 Using Direct I/O in Linux

With SSDs in Linux it is recommended to use direct I/O instead of buffered I/O when possible. The Linux I/O 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, and the benefits of buffering are reduced.
- Buffering read I/Os consumes extra CPU cycles for memory copy operations. At I/O rates typical for SSDs, this extra CPU consumption may be high and create a read performance bottleneck.

To use direct I/O 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 I/O, 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’

If your software does not use the O_DIRECT flag and cannot be modified – but at the same time does not have to use a file system – a ‘raw device’ can be created and used instead of the standard block device name. The raw device I/O will bypass the buffer similar to using the O_DIRECT flag.

```
raw /dev/raw/raw<N> /dev/sdX
```

4.3 Selecting I/O Scheduler in Linux

I/O schedulers in Linux determine how competing I/Os from different processes are prioritized and balanced. With SSDs, it is recommended to use the simplest noop (no operation) scheduler, unless there are processes in the system that can produce excessive amounts of I/Os and starve other processes.

The following command displays available and active schedulers:

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

The following command displays available parameters for the currently active scheduler:

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

The following command sets noop scheduler:

```
echo noop > /sys/block/sdX/queue/scheduler
```
4.4 Optimizing RAID Configurations

4.4.1 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* I/O feature that boosts random I/O processing capability of the controller.

4.4.2 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 use RAID 10.

4.4.3 Setting RAID Stripe Unit Size

For higher random IOPS, stripe unit size should not be smaller than 2X of the typical transfer size used by the application. This minimizes the frequency of transfers crossing the stripe boundaries and hitting 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.

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

$$0.5 \times \frac{\text{Transfer size}}{\text{Number of striped drives}}$$

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

4.4.4 Using RAID Write-Back Caching

Write-back caching at the controller level reduces 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).

However, with write-back caching enabled, high write rates may starve read performance by occupying the SSD’s I/O queue slots. The total amount of queue slots is 32 per SSD. When the write-back cache gets full and read queue depth is, for example, only 2, the remaining 30 queue slots will be occupied by writes. In this case the resulting read/write performance ratio will be roughly 2:30.
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 I/Os should scale in proportion to the amount 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 SSD 320 Series has embedded capacitors that protect the drive write cache, and the drive write cache is always enabled. This fault tolerance at the drive level reduces the need for write back caching at the RAID controller level, especially in RAID 0/1/10 configurations.

### 4.4.5 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. It also consumes RAID memory that could be used 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 $2 \times \text{Stripe\_unit\_size} \times \text{Number\_of\_striped\_drives}$. 
4.4.6 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.

4.4.7 I/O Queuing in Striped Configurations

For small read I/Os (that is, I/O size < RAID stripe size), optimal queue depth should be calculated by multiplying the optimal queue depth per drive by the number of drives in the array (excluding parity drives in RAID 5/50/6/60).
5.0 Measuring Intel® SSD 320 Series Performance

This section describes how to measure performance of an Intel SSD 320 Series:

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

5.1 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 I/O
- I/O transfer size
- Read, write, or mixed I/O operations
- I/O queue depth or the number of parallel threads

5.2 Conditioning the SSD Before Measuring Performance

Before taking write or mixed workload performance measurements of an Intel SSD 320 Series, make sure the SSD is in a steady state. This process to prepare the SSD for performance measurements is known as conditioning.

To condition the Intel SSD 320 Series:

1. Write the full capacity of the SSD sequentially.
   - This step ensures all LBAs have valid data and removes all fragmentation.
   - For example, in Linux* use the following command:
     \[ \text{dd if=/dev/zero of=/dev/sdX bs=256k oflag=direct} \]

2. Write the amount equal to 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.

3. Measure performance using the target workload.
   - In this step you can use different I/O queue depths and read/write ratios.

Refer to Figure 1 on page 7 to review how performance typically stabilizes during Step 2.
5.3 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 I/O
- I/O transfer size
- Read, write, or mixed I/O operations
- I/O queue depth or the number of parallel threads

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

- I/O queuing is configured correctly.
  - In Iometer*, set the desired queue depth (on the Disk Targets tab, set the 
    # of Outstanding I/Os setting). By default, this setting is set to 1, which means 
    no queuing used.
  - In Linux* using Asynchronous I/O 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 I/O 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 I/O when possible. See Section 4.2, "Using Direct I/O in Linux", 
  for details.
5.3.1 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 300 GB Intel SSD 320 Series with two different read/write ratios and queue depths.

```bash
# First filling the drive sequentially
dd if=/dev/zero of=/dev/sdX bs=256k oflag=direct
# Conditioning the drive with 300GB 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=300G
# 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
```
6.0 Estimating Lifetime and Monitoring Wear

This section describes the Intel SSD 320 Series endurance characteristics and explains how to monitor wear and estimate drive lifetime.

- Intel® SSD 320 Series Endurance Characteristics
- Monitoring Wear Using SMART
- Estimating Drive Lifetime

6.1 Intel® SSD 320 Series Endurance Characteristics

The Intel SSD 320 Series is built with MLC Intel® NAND Flash Memory and has 5,000 program/erase cycles resource (Max_PE_Cycles) rated in accordance with JEDEC standard JESD47G.

SSD lifetime requirements in server applications usually range between 2 to 10 years. The Intel SSD 320 Series 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 320 Series 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:

\[
Write\_Amplification\_Factor = \frac{NAND\_Writes\_GB}{Host\_Writes\_GB}
\]

\[
Consumed\_PE\_Cycles = \frac{(Host\_Writes\_GB \times Write\_Amplification\_Factor)}{Raw\_NAND\_Capacity\_GB}
\]

\[
Host\_Write\_Endurance\_GB = \frac{(Raw\_NAND\_Capacity\_GB \times Max\_PE\_Cycles)}{Write\_Amplification\_Factor}
\]

Write Amplification Factor (WAF) depends on workload and drive configuration. For 100% sequential write workloads, the Write Amplification Factor is 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 – less random writes have lower WAF
- Locality – localized writes (for example, actively writing to a 20 GB range only) have lower WAF
- Write transfer size – writes with larger transfer sizes have lower WAF
- Usable capacity – smaller usable capacity 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 for reducing the Write Amplification Factor and improving drive endurance. See Section 4.1 Adjusting Usable Capacity for details.

6.2 Monitoring Wear Using SMART

The Intel SSD 320 Series 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.

6.2.1 Timed 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: \( E2h\textunderscore Raw\textunderscore Value / 1024 = 100 - E9h\textunderscore Normalized\textunderscore Value. \)
6.2.2 Media Wearout Indicator Attribute (E9h/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.

6.2.3 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 value decreases when NAND blocks retire. The pre-fail threshold value for this attribute is 10.

6.3 Estimating Drive Lifetime

This section outlines the steps that Intel recommends for estimating drive lifetime of the Intel SSD 320 Series. 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
   ```

3. Power cycle the drive

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

5. 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
   ```

6. Read SMART attribute E2h/226 (Timed Workload Media Wear). This is the start value.
   
   ```
   > smartctl -A /dev/sdX
   ```
7. 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.

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

   > smartctl -A /dev/sdX

9. Calculate estimated drive lifetime with this workload.

   \[
   \text{Added\_Wear\_Percent} = \frac{(E2h\_Raw\_Value\_End - E2h\_Raw\_Value\_Start)}{1024}
   \]

   \[
   \text{Estimated\_Lifetime} = 100 \times 1024 \times \text{Test\_Time} \div (E2h\_Raw\_Value\_End - E2h\_Raw\_Value\_Start)
   \]
7.0 Additional Information

For more information on Intel SSDs, see the corresponding documentation.

Table 2. Related Documentation

<table>
<thead>
<tr>
<th>Document</th>
<th>Document No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel® Solid-State Drive 320 Series Product Specification</td>
<td>325152</td>
</tr>
<tr>
<td>Intel® Solid-State Drive 320 Series Enterprise Server/Storage Application Product Specification Addendum</td>
<td>325170</td>
</tr>
<tr>
<td>Intel® Solid-State Drive 320 Series Enhanced Power-Loss Data Protection Technology Brief</td>
<td>325207</td>
</tr>
<tr>
<td>Monitoring Media Wearout Levels of Intel® Solid-State Drives</td>
<td>325551</td>
</tr>
</tbody>
</table>

8.0 Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>June 2011</td>
<td>001</td>
<td>Initial release</td>
</tr>
</tbody>
</table>