ABSTRACT

As solid state drives (SSDs) technology matures, the performance it delivers is constrained by old disk controller technologies. The NVMe protocol on PCIe interface allows to bypass the controller by attaching directly a disk drive to the bus. In this report we present the results of using Microsoft Storage Spaces and Intel RSTe with SSDs attached to SATA and NVMe. The tests compare RAID0, RAID5, and RAID 10 bandwidth performances on up to four drives.

The tests show that it is possible to obtain up to 10GB/s of throughput with 4 NVMe drives, about four times the performance achieved through SATA by SSDs drives based on the same technology.

Antonio Cisternino, Maurizio Davini
TR-02-16
Executive Summary

NVMe SSD drives provide more performance in terms of bandwidth and lower latency than SAS and SATA based counterparts (look for instance to [5]). However, traditional disk controllers feature functionalities such as RAID, which are unavailable to disks attached directly to the PCIe® bus. In this paper we present the results of our experiments to test the performance of Microsoft Storage Spaces in aggregating NVMe PCIe® and SATA SSD drives (20nm MLC). Tests aimed at verifying the throughput that can be achieved using the open source diskspd tool [6] on software defined RAID modules to understand the potential impact of not using a hardware controller.

The results have shown that it is possible to scale up linearly on RAID0 aggregation, though with PCIe drives the significant bandwidth they offer seems to saturate the PCI Express bus when adding the fourth drive. However, we achieved the significant bandwidth of 10GB/s with this configuration, a 4x improvement with respect to SATA drives. We also compared, using only NVMe drives, the RAID performances of Microsoft Storage Spaces and Intel RSTe (even though they are designed for different purposes). The two software perform similarly though Intel RSTe seems to offer slight better performance at the cost of an increased CPU usage.

Introduction

This report is the second in a series dedicated to the evaluating solid state drives (SSDs) performances. In the first report [5] we compared the performance of SATA and NVMe PCIe SSDs connection using Hammer DB, showing that the NVMe protocol used for attaching directly drives to the PCIe bus offers significant improvements with respect to SATA controller in terms of bandwidth and latency under a database-like workload. This initial work focused on single drive use in order to avoid the differences between controller-based and software-based drive aggregation.

We now look into drive aggregation performance using only software defined storage with the goal of understanding how the superior performance NVMe scales up when multiple drives are combined on a latest generation server. We used the open source diskspd tool [6] on Windows Server 2012R2 to perform the tests and get the bandwidth measurements. Latency has not been measured because it is expected to keep a behavior similar to that reported in the previous work. Moreover, we decided to focus on bandwidth because we were able to measure almost 3GB/s when reading from a NVMe drive, which means that with a linear scale up we would achieve 12GB/s throughput, significantly close to the theoretical limit of x16 lanes transfer of PCIe 3.0 of 15.754GB/s.

Tests have been organized along two directions: (a) compare how Microsoft Storage Spaces software defined RAID0 disk aggregation behaves with 1 to 4 drives using SATA and NVMe PCIe interfaces; and (b) compare Microsoft Storage Spaces RAID with Intel® Rapid Storage Technology Enterprise (Intel RSTe) for NVMe PCIe SSDs.

In this technical report, we present experiment results comparing Intel® SSD Data Center S3610 Series (“S3610”, 1.6TB, 2.5in SATA 6Gbps, 20nm, MLC) [2] attached to a HBA/bus controller on the SATA bus and Intel® SSD Data Center P3600 Series (“P3600”, 1.6TB, 2.5in PCIe 3.0 (8Gbps), 20nm, MLC) [3]—using the NVMe protocol—directly connected to the PCIe bus (no separate HBA/adapter required). Tests were conducted on a NVMe-capable Dell® PowerEdge® R630 rack server with 2 Intel® Xeon® E5-2683 v3 processors and 128GB of RAM. Intel Rapid Storage Technology software version 4.5.0.2072. The workload used is diskspd running Microsoft® Windows Server® 2012 R2 Standard edition operating system. The specific details about the numbers obtained, system configuration, and testing procedures are discussed in the appendix.

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

The diskspd tool is an open source tool developed by Microsoft and used by its engineering team to evaluate storage performance. It is designed to generate a wide variety of disk patterns given a huge number of command line switches. We restrained ourselves to just four tests (read and write) executed for 10 seconds each:

- Seq128K56Q: Sequential R/W, 128KB block size, 56 queues, single thread
- Rand4K56Q: Random R/W, 4KB block size, 56 queues, single thread
- Seq1M1Q: Sequential R/W, 1MB block size, 1 queue, single thread
- Rand4K1Q: Random R/W, 4KB block size, 1 queue, single thread

The number of queues has been empirically determined by testing the bandwidth improvement for the Seq128K for improvement on a particular volume; on the server we tested 56 seemed to be a good value.

Solid state drives usually exhibit different performance when they are empty or full, due to the way firmware allocates cells to ensure optimal usage of the gates. In particular, when a disk is empty the firmware can use the available space to optimize access. If, however, the drive is preconditioned (i.e. it’s full) the performance may degrade, even significantly. For this reason, we performed the tests with both the disk full and empty to verify how the disk state may affect the performance.

The test has been conducted automatically using a powershell script, which is available in the appendix of this report, according to the following steps:

- Allocate the volume (only for Microsoft Storage Spaces)
- Format it using NTFS
- For each of the four tests:
  o If precondition is required create a file using diskspd large enough to fill disk leaving the space required for the test
  o The read test is performed
  o The created file is removed to avoid cache effects
  o The write test is performed
  o The created file is removed

We created manually the volumes using Intel RSTe, and then performed the tests using the same script functions. For the test results we used the Microsoft terminology for the volume resiliency: Simple, Parity, and Mirror, corresponding to RAID-0, RAID-5, and RAID-10.

Microsoft Storage Spaces

Since Storage Spaces technology allows to pool drives of any kind, it has been possible to perform the same tests on both kind of drives.

SSD Bandwidth

The bandwidth for SATA drives is about 2GB/s using four drives, showing how the SSD is constrained by the HBA/bus controller. The constraint is evident if we look at the performance of four SATA drives when parity is used: the bandwidth in this case is 6GB/s, three times the maximum bandwidth otherwise obtained. The total CPU used by the test increases more than three times since, in addition to a larger size of data processed during the test, there is the parity processing. From these numbers we conclude that when the information is spread...
across four drives the Storage Spaces implementation is capable of using multiple drives simultaneously for serving the 56 requests in queue.

<table>
<thead>
<tr>
<th>#Drives</th>
<th>Resiliency</th>
<th>CPU Usage</th>
<th>MB/s</th>
<th>IOPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Simple</td>
<td>0,12%</td>
<td>518,23</td>
<td>4145,85</td>
</tr>
<tr>
<td>2</td>
<td>Simple</td>
<td>0,23%</td>
<td>1045,14</td>
<td>8361,13</td>
</tr>
<tr>
<td>3</td>
<td>Simple</td>
<td>0,33%</td>
<td>1574,75</td>
<td>12597,98</td>
</tr>
<tr>
<td>4</td>
<td>Simple</td>
<td>0,42%</td>
<td>2097,11</td>
<td>16776,89</td>
</tr>
<tr>
<td>4</td>
<td>Mirror</td>
<td>0,42%</td>
<td>2093,89</td>
<td>16751,1</td>
</tr>
<tr>
<td>4</td>
<td>Parity</td>
<td>1,91%</td>
<td>6071,75</td>
<td>48573,98</td>
</tr>
</tbody>
</table>

Table 1: SATA bandwidth and IOPS performance for Seq128K56Q read.

NVMe drives show greater performance, with 9.6GB/s peak using four drives. In this case, however, the parity calculation slows down the transfer process since without controller the drives are not capped. Moreover, the total CPU is higher due to the larger amount of data processed.

<table>
<thead>
<tr>
<th>#Drives</th>
<th>Resiliency</th>
<th>CPU Usage</th>
<th>MB/s</th>
<th>IOPS</th>
<th>NVMe v. SATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Simple</td>
<td>0,68%</td>
<td>2852,4</td>
<td>22819,21</td>
<td>5.50x faster</td>
</tr>
<tr>
<td>2</td>
<td>Simple</td>
<td>1,34%</td>
<td>5699,84</td>
<td>45598,69</td>
<td>5.45x faster</td>
</tr>
<tr>
<td>3</td>
<td>Simple</td>
<td>1,77%</td>
<td>8555,4</td>
<td>68443,16</td>
<td>5.43x faster</td>
</tr>
<tr>
<td>4</td>
<td>Simple</td>
<td>1,96%</td>
<td>9584,18</td>
<td>76673,46</td>
<td>4.57x faster</td>
</tr>
<tr>
<td>4</td>
<td>Mirror</td>
<td>1,94%</td>
<td>9352,74</td>
<td>74821,96</td>
<td>4.46x faster</td>
</tr>
<tr>
<td>4</td>
<td>Parity</td>
<td>1,97%</td>
<td>6236,02</td>
<td>49888,15</td>
<td>1.02x faster</td>
</tr>
</tbody>
</table>

Table 2: NVMe SSD bandwidth and IOPS performance for Seq128K56Q read.

**SSD Scale-up**

If we consider how the aggregated bandwidth scales up between 1 and 4 drives (i.e. the ratio between the bandwidth of a disk aggregation and the one of a single disk) we obtain the following graphs.

*Other names and brands may be claimed as the property of others.*
As you can notice SATA drives scale up almost linearly while NVMe drives suffer a slow down when the fourth drive is added. This is not unexpected: four times the bandwidth of an NVMe drive would mean 12GB/s for reading; considering that the PCIe 3.0 maximum nominal bandwidth is of 15.8GB/s we believe that we start seeing contentions on the PCI express bus, slightly degrading the performance.

Another salient aspect is the comparison between the bandwidth and the CPU usage change. In all cases the bandwidth increment implies a lesser CPU usage increment, due to the DMA nature of the transfer; this is what we expected. This relation between bandwidth increment and lesser CPU Usage increment has been observed for all the other tests using Microsoft Storage Spaces.

The sequential read lead to a similar result (though reaching 3.5x rather than 4x), while it is interesting to see how the bandwidth graph looks for the 4KB random access test:

![Bandwidth speedup: Rand4K56Q Read](image)

In this case the SATA drives show the scale up while the NVMe drives seem to not benefit from the aggregation. We believe that this behavior shows the effect of the caching subsystem of the SATA controller, NVMe drives do not get this benefit without the HBA/bus controller, and the random access of small blocks does not benefit from the aggregated bandwidth of the drives.

Notice that the absolute bandwidth in both cases is comparable (NVMe: 441MB/s, SATA: 552MB/s), and it is hardly a bandwidth test. It is also interesting to see the difference of preconditioning SSD drives: with a full SATA drive the performance is only 116MB/s.

**SATA vs. NVMe PCIe**

When we plot the ratio between the bandwidth obtained using NVMe PCIe drives and the one observed using SATA drives we easily notice the superiority of NVMe-based SSDs. The typical speedup of NVMe over SATA for sequential read is 5x as shown in the following graphs.

*Other names and brands may be claimed as the property of others.*
Random access shows significant differences between preconditioned and empty disk tests. NVMe drives perform much better than SATA when not full. Otherwise the performance is comparable as already discussed.

For parity and mirror resiliency (RAID-5 and RAID-10 respectively), the sequential read is break-even for parity as already discussed. As expected mirror is in general higher performing for sequential access.

Random access reflects the same trends we already found for simple resiliency.

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

Preconditioning affects the performance of drives as clearly noticed in the graphs. It is natural to wonder how much this phenomenon affects a single drive performance. In the following table we included the ratio between bandwidth tests of preconditioned and empty NVMe PCIe drives. As it can be noticed, the performance of the drive degrades at most by 10%, but in most cases it even speeds up.

<table>
<thead>
<tr>
<th>Test</th>
<th>RW</th>
<th>DriveFull/DriveEmpty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rand4K1Q</td>
<td>Read</td>
<td>1,57</td>
</tr>
<tr>
<td>Rand4K1Q</td>
<td>Write</td>
<td>0,89</td>
</tr>
<tr>
<td>Rand4K56Q</td>
<td>Read</td>
<td>1,50</td>
</tr>
<tr>
<td>Rand4K56Q</td>
<td>Write</td>
<td>0,94</td>
</tr>
<tr>
<td>Seq128K56Q</td>
<td>Read</td>
<td>1,00</td>
</tr>
<tr>
<td>Seq128K56Q</td>
<td>Write</td>
<td>1,06</td>
</tr>
<tr>
<td>Seq1M1Q</td>
<td>Read</td>
<td>1,52</td>
</tr>
<tr>
<td>Seq1M1Q</td>
<td>Write</td>
<td>1,16</td>
</tr>
</tbody>
</table>

Table 3: Ratio of test bandwidth between preconditioned and empty single NVMe PCIe drive.

The same ratio applied to a single SATA drive leads to similar results even though it seems that empty SSD drives for random read in this case perform significantly worse. It is likely to be an effect of the interaction between the drive firmware and the HBA/bus controller policies.

<table>
<thead>
<tr>
<th>Test</th>
<th>RW</th>
<th>DriveFull/DriveEmpty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rand4K1Q</td>
<td>Read</td>
<td>246,95</td>
</tr>
<tr>
<td>Rand4K1Q</td>
<td>Write</td>
<td>252,19</td>
</tr>
<tr>
<td>Rand4K56Q</td>
<td>Read</td>
<td>10,32</td>
</tr>
<tr>
<td>Rand4K56Q</td>
<td>Write</td>
<td>9,01</td>
</tr>
<tr>
<td>Seq128K56Q</td>
<td>Read</td>
<td>0,98</td>
</tr>
<tr>
<td>Seq128K56Q</td>
<td>Write</td>
<td>0,96</td>
</tr>
<tr>
<td>Seq1M1Q</td>
<td>Read</td>
<td>5,88</td>
</tr>
<tr>
<td>Seq1M1Q</td>
<td>Write</td>
<td>3,04</td>
</tr>
</tbody>
</table>

Table 4: Ratio of test bandwidth between preconditioned and empty single SATA drive.

Microsoft Storage Spaces vs. Intel RSTe

Intel offers a software RAID solution for NVMe PCIe drives called Intel Rapid Storage Technology enterprise (or Intel RSTe). The goal of this software is to create resilient volumes by mimicking typical RAID configurations of hardware controllers. Microsoft Storage Spaces features resiliency settings to storage pools among different and possibly heterogeneous drives. It is difficult to compare the two solutions even though the latter can be used to implement RAID for NVMe drives.

We compared the performance we recorded for volumes created using Microsoft Storage Spaces with volumes created using Intel RSTe, under the assumption that Simple resiliency means RAID-0, Parity resiliency means RAID-5, and Mirror resiliency means RAID-10.

*Other names and brands may be claimed as the property of others.
As shown by the graphs above the sequential performance of Intel RSTe is better when it comes to parity, though it is considerably faster than Storage Spaces. At the same time the increment requires a larger increment in the CPU usage shown by Intel RSTe. Random access tests show a similar behavior as shown in the graphs below.

### Conclusions

In this technical report we have presented the results of testing NVMe PCIe and SATA solid-state drives aggregated using Microsoft Storage Spaces, and Intel RSTe. The `diskspd` tool was used for performing the benchmark.

Experiments show that software RAID is capable of aggregating SSD drives very efficiently, showing a linear speedup as the number of aggregated drives increase. NVMe drives are so efficient that we recorded a non-optimal speedup when adding the fourth drive, due to the approaching of the maximum bandwidth allowed by PCIe bus. It is possible to overcome some of these limitations by dedicating more PCIe lanes to drives or using a PCIe expander (both of these options are beyond the scope of this technical report).

Random disk access patterns with small blocks have shown to benefit from cache offered by the SATA disk controller, though not enough to outperform NVMe drives.

The bandwidth recorded has been of 9.6GB/s, an impressive number for disk access, and surely one to be considered by software developers when they assume that the “disk is order of magnitude slower than memory”.

Intel RSTe has proven to be overall more efficient than Microsoft Storage Spaces including a considerable performance gain when it comes to parity (though the two software have a little subset of shared functionalities); however, the improvement comes at a cost of an additional use of the CPU larger than the speedup obtained.

*Other names and brands may be claimed as the property of others.*
Finally, we have been able to appreciate the difference between testing empty and preconditioned drives. In all our tests the performance of preconditioned drives never degraded by more than about 10%, which ensure a stable performance from the drive.

Bibliography

6. https://github.com/Microsoft/diskspd

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

Hardware Configuration
Tests have been performed on a Dell® R630 with the following configuration:

- Dell R630 with support for up to four NVMe drives
- Perc® H730 Mini controller for Boot drive
- 2 Intel® Xeon® E5-2683 v3 2GHz CPU
- 2 Intel® SSD DC S3710 Series (SATA boot drive)
- 4 Intel® SSD SC P3600 Series (NVMe PCIe)
- 4 Intel® SSD DC S3610 Series (SATA)
- 128Gb (8x16Gb) DDR4 RAM

System Setup
The system has been installed with Windows Server 2012 R2 Standard edition with the drivers provided by Dell for the R630 and the Intel® Solid State Drive Data Center Family for NVMe Drivers. We then installed all the latest updates from Microsoft. The SATA RAID hardware control for Intel SSD DC S3610 Series has been configured in pass-through mode. The system has joined the AD of our network. The versions of software used for testing are diskspd 2.0.15, and Intel RSTe 4.5.0.2072.

Test script
The

```powershell
$nvme0 = Get-PhysicalDisk -FriendlyName PhysicalDisk5
$nvme1 = Get-PhysicalDisk -FriendlyName PhysicalDisk6
$nvme2 = Get-PhysicalDisk -FriendlyName PhysicalDisk7
$nvme3 = Get-PhysicalDisk -FriendlyName PhysicalDisk8

$sata0 = Get-PhysicalDisk -FriendlyName PhysicalDisk0
$sata1 = Get-PhysicalDisk -FriendlyName PhysicalDisk1
$sata2 = Get-PhysicalDisk -FriendlyName PhysicalDisk2
$sata3 = Get-PhysicalDisk -FriendlyName PhysicalDisk3

    $name = $poolName + "\_vd"
    $drive = New-VirtualDisk -StoragePoolFriendlyName $poolName -ResiliencySettingName $resiliency -FriendlyName $name
    $vol = New-Partition -DiskId $drive.UniqueId -UseMaximumSize -DriveLetter $letter
    $vol | Format-Volume -Confirm:$false -FileSystem $filesystem -NewFileSystemLabel $poolName
    return $drive
}

    $fn = $letter + "\disktest.dat"
    $bsz = "$b + $bsz
    $q = "$q + $q
    $sq = "$square
    if ($false) { $sq = "-r" }
    C:\Users\cisterni\Desktop\Diskspd-v2.0.15\amd64fre\diskspd.exe -c500G $bsz -d10 $q -t1 $sq -w -h -w0 $fn
    rm $fn
    C:\Users\cisterni\Desktop\Diskspd-v2.0.15\amd64fre\diskspd.exe -c500G $bsz -d10 $q -t1 $sq -w -h -w100 $fn
    rm $fn
}
```

*Other names and brands may be claimed as the property of others.*
function testVolume ([char] $letter, [bool] $reserve = $true)
{
    if ($reserve) {
        $vol = Get-Volume -DriveLetter D
        $gig = 1024 * 1024
        $sz = "-c" + ((math::Truncate($vol.SizeRemaining - 500 * $gig) / $gig)) + "G"
        C:\Users\cisterni\Desktop\Diskspd-v2.0.15\amd64fre\diskspd.exe $sz -056 -t1 -b256k d:\reserve.dat
    }
    echo "Test sequential 128K block queues 56..."
    runTest -letter $letter -seq $true -block 128k -queues 56
    echo "Test random 4K block queues 56..."
    runTest -letter $letter -seq $false -block 4k -queues 56
    echo "Test sequential 1M block queues 1..."
    runTest -letter $letter -seq $true -block 1m -queues 1
    echo "Test random 4K block queues 1..."
    runTest -letter $letter -seq $false -block 4k -queues 1
}

function testDrive ([string] $pool, [string] $resiliency, [bool] $reserve = $true)
{
    $n = "$pool = " + "wd"
    echo "PhysicalDisk -storagePool (Get-StoragePool -FriendlyName $pool) | echo"
    echo "Creating volume..."
    $drive = createVolume -poolName $pool -resiliency $resiliency -letter D
    testVolume -letter D -reserve $reserve
    Remove-VirtualDisk -Confirm:$false -FriendlyName $n
}

function testPool ([string] $name, [bool] $reserve=$true, $drives)
{
    $reserveName = "$resno"
    if ($reserve -eq $true) { $reserveName = "resyes" }
    echo "Creating pool $name..."
    $pool = New-StoragePool -FriendlyName $name -PhysicalDisks $drives[0] -StorageSubSystemFriendlyName "Storage Spaces on intelssd"
    echo "done.
    echo "Testing one drive..."
    testDrive -pool $name -resiliency Simple -reserve $reserve > "$name-sp-ntfs-1-drive-$reserveName.txt"
    echo "done.
    echo "Adding a new drive..."
    Add-PhysicalDisk -storagePoolFriendlyName $name -PhysicalDisks $drives[1]
    echo "done.
    echo "Testing two drives..."
    testDrive -pool $name -resiliency Simple -reserve $reserve > "$name-sp-ntfs-2-drive-$reserveName.txt"
    echo "done.
    echo "Adding a new drive..."
    Add-PhysicalDisk -storagePoolFriendlyName $name -PhysicalDisks $drives[2]
    echo "done.
    echo "Testing three drives..."
    testDrive -pool $name -resiliency Simple -reserve $reserve > "$name-sp-ntfs-3-drive-$reserveName.txt"
    echo "done.
    echo "Adding a new drive..."
    Add-PhysicalDisk -storagePoolFriendlyName $name -PhysicalDisks $drives[3]
    echo "done.
    echo "Testing four drives..."
    testDrive -pool $name -resiliency Simple -reserve $reserve > "$name-sp-ntfs-4-drive-$reserveName.txt"
    echo "done.
    echo "Removing pool $name"
    Remove-StoragePool -Confirm:$false -FriendlyName $name
    echo "done."}

function testParityPool ([string] $name, [bool] $reserve=$true, $drives)
{
    $reserveName = "$resno"
    if ($reserve -eq $true) { $reserveName = "resyes" }
    echo "Creating pool $name..."
    $pool = New-StoragePool -FriendlyName $name -PhysicalDisks $drives -StorageSubSystemFriendlyName "Storage Spaces on intelssd"
    echo "done.
    echo "Testing with parity 4 drives..."
testDrive -pool $name -resiliency Parity -reserve $reserve > "$name-sp-ntfs-4-drive-parity-$reserveName.txt"

  echo "done."
  echo "Testing with mirror 4 drives..."

  testDrive -pool $name -resiliency Mirror -reserve $reserve > "$name-sp-ntfs-4-drive-mirror-$reserveName.txt"

  echo "done."
  echo "Removing pool $name"

Remove-StoragePool -Confirm:$false -FriendlyName $name
  echo "done."

#

testPool -name NVMe -reserve $false -drives @($nvme0, $nvme1, $nvme2, $nvme3)
testPool -name NVMe -reserve $true -drives @($nvme0, $nvme1, $nvme2, $nvme3)
testPool -name SATA -reserve $false -drives @($sata0, $sata1, $sata2, $sata3)
testPool -name SATA -reserve $true -drives @($sata0, $sata1, $sata2, $sata3)
testParityPool -name NVMe -reserve $false -drives @($nvme0, $nvme1, $nvme2, $nvme3)
testParityPool -name NVMe -reserve $true -drives @($nvme0, $nvme1, $nvme2, $nvme3)
testParityPool -name SATA -reserve $false -drives @($sata0, $sata1, $sata2, $sata3)
testParityPool -name SATA -reserve $true -drives @($sata0, $sata1, $sata2, $sata3)

function getCSV (s$filename) {
  $t = 1
  $out = ""
  cat $filename |
  where { $_ -like "avg.*" -or $_ -like "  0 |" -or $_ -like "Test *" ]
    ForEach-Object { $_ -replace "\"D:\\\disktest\data\\"(500GB\\)\", "" ]
      ForEach-Object { $_ -replace "\"(d)\\.\(d)\\"", "$1,$2" ]
        ForEach-Object { $_ -replace "\"avg\\)\\"", "" ]
          ForEach-Object -Process {
            if ($._ -like "Test ") { $t = 0; $out = $out + "r'n" + $._; }
            elseif ($._ -like "Avg") { $t = 1; $out = $out + "r'n"; + $._; }
            else ($t -eq 1) { $t = -1; $out = $out + "r'n"; + $._; }
          } -End { $out; }
        } -End { $out; }
      } -End { $out; }
    } -End { $out; }
  }
}

del .\out.csv
getSuite *.txt > .\out.csv

Intel technologies’ features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. **No computer system can be absolutely secure.** Check with your system manufacturer or retailer or learn more at intel.com.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit [http://www.intel.com/performance](http://www.intel.com/performance).

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Intel, the Intel logo, Intel® Xeon®, Intel® SSD DC S3610, Intel® SSD DC S3710, and Intel® SSD DC P3600 are trademarks of Intel Corporation in the U.S. and/or other countries.

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