SSD & HDD Performance Testing with TKperf

Georg Schönberger <gschoenberger@thomas-krell.com>
LinuxTag 2013
The Who
Agenda

1) The Why
2) The How
3) IOPS
4) Throughput
5) Write Saturation
6) Latency
7) Résumé
• Solid State Storage Performance Test Specification\(^1\)

1 http://www.snia.org/tech_activities/standards/curr_standards/pts
The Why
1) The Why

- Which devices are suited for enterprise
  - What is the difference to a consumer device?
- SNIA
  - SSDs are different
- Fio
  - Flexible, actively developed
- Python
  - Matplotlib
- Open Source
  - Get Feedback, improve tests
1) The Why

- Automatism
  - Fio calls are logged
  - Write results to XML
  - Generate rst report

```
sudo tkperf ssd intelDCS3700 /dev/sdb -nj 2 -iod 16 -rfb
```
Handy Tools
2) The How

- Information gathering and Secure Erase
  - hdparm, lsblk, lsscsi

- Direct IO

- Number of jobs, IO depth
  - Reach the maximum of a device

# fio --rw=write --name=test --size=20M
[...]
Run status group 0 (all jobs):
  WRITE: io=20480KB, aggrb=930909KB/s

# fio --rw=write --name=test --size=20M --direct=1
[...]
Run status group 0 (all jobs):
  WRITE: io=20480KB, aggrb=28563KB/s

# hdparm -I /dev/sdb|grep -e NCQ -e depth
Queue depth: 32
  * Native Command Queueing (NCQ)
• Asynchronous IO
  • Outstanding IOs in flight
  • Requires direct IO
• Difference application ↔ device levels\(^1\)
  • Block sizes are split
  • IO scheduler

\(^1\) http://www.spinics.net/lists/fio/msg01526.html
IO Depth

- It matters!
SNIA
Tests

• Synthetic
  • Test a specific scenario
  • Testing an individual component
  • Corner case behavior
• Get a first impression, compareable
• Not an application based test
  • blkreplay
New Words...

- Secure Erase
- Workload Independent Preconditioning
  - 2x device 128KB sequential write
- Workloads (IO access)
  - Random, sequential
  - Mixed (95/5 R/W)
- Steady State
  - Dependent Variable
    - States when device is stable
  - Performance values of last 5 rounds
Device: Intel DC S3700 Series SSDs
## Test Overview

<table>
<thead>
<tr>
<th>IOPS</th>
<th>TP</th>
<th>Write Sat.</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Access</td>
<td>Sequential Access</td>
<td>Random Access</td>
<td>Random Access</td>
</tr>
<tr>
<td>R/W 100/0, 95/5, 65/35, 50/50, 36/65, 5/95, 0/10</td>
<td>R/W 100/0, 0/100</td>
<td>R/W 100% writes</td>
<td>R/W 100/0, 65/35, 0/100</td>
</tr>
<tr>
<td>BS 1M, 128K, 64K, 32K, 16K, 8K, 4K, 0.5K</td>
<td>BS 1M, 64K, 8K, 4K, 0.5K</td>
<td>BS 4K</td>
<td>BS 8K, 4K, 0.5K</td>
</tr>
<tr>
<td>DV 4K random writes</td>
<td>DV 1M seq. writes</td>
<td>-</td>
<td>DV 4K random write mean latency</td>
</tr>
</tbody>
</table>
IOPS

→ A random Workload
3) **IOPS**

- IO operations per second
- THE number for SSDs
- Dependent variable
  - 4KB random writes

---

Make Secure Erase
Workload Ind. Preconditioning
While not Steady State
  For workloads [100, 95, 65, 50, 35, 5, 0]
  For block sizes [1024k, 128k, 64k, 32k, 16k, 8k, 4k, 512]
  Random Workload for 1 Minute
IOPS Measurement Plot

Average IOPS vs. Block Size and R/W Mix %

<table>
<thead>
<tr>
<th>Block Size</th>
<th>Wld. → 100/0</th>
<th>95/5</th>
<th>65/35</th>
<th>50/50</th>
<th>35/65</th>
<th>5/95</th>
<th>0/100</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>99793.0</td>
<td>80136.0</td>
<td>47181.6</td>
<td>42332.6</td>
<td>39282.0</td>
<td>36557.6</td>
<td>37873.8</td>
</tr>
<tr>
<td>4k</td>
<td>36192.0</td>
<td>75921.6</td>
<td>48053.0</td>
<td>42298.8</td>
<td>38834.0</td>
<td>36022.8</td>
<td>36788.2</td>
</tr>
<tr>
<td>8k</td>
<td>85564.0</td>
<td>72106.0</td>
<td>49605.0</td>
<td>44574.2</td>
<td>42669.2</td>
<td>33858.0</td>
<td>34035.0</td>
</tr>
<tr>
<td>16k</td>
<td>58048.8</td>
<td>47198.8</td>
<td>30972.6</td>
<td>27568.8</td>
<td>26603.8</td>
<td>18538.0</td>
<td>17653.4</td>
</tr>
<tr>
<td>32k</td>
<td>34408.0</td>
<td>28315.4</td>
<td>18163.4</td>
<td>15687.8</td>
<td>14661.6</td>
<td>10583.0</td>
<td>10239.2</td>
</tr>
<tr>
<td>64k</td>
<td>16424.4</td>
<td>16467.8</td>
<td>10337.2</td>
<td>8401.0</td>
<td>7772.2</td>
<td>6131.2</td>
<td>5879.4</td>
</tr>
<tr>
<td>128k</td>
<td>10392.4</td>
<td>9185.6</td>
<td>5322.8</td>
<td>4302.8</td>
<td>4191.6</td>
<td>3575.0</td>
<td>2936.2</td>
</tr>
<tr>
<td>1024k</td>
<td>1623.2</td>
<td>1420.8</td>
<td>668.8</td>
<td>518.2</td>
<td>480.2</td>
<td>401.8</td>
<td>489.2</td>
</tr>
</tbody>
</table>

Device: STEC s1120 PCIe Accelerator
Excursus: PCIe Cards

- Secure Erase must be done individually
  - Patch src/perfTest/DeviceTest.py
    ```
    # sdmcmd64 sanitize target=gen4pcie:Drive0 sanitizetype=erase
    # isdct -device 0 -drive 0 -erase -force
    ```

- `hdparm` doesn't work
  - Use a manually created description file
    ```
    # sdmcmd64 GetInfo target=gen4pcie:Drive0 > S1120.dsc
    # tkperf ssd S1120 /dev/skd1 -nj 2 -iod 16 -rfb -dsc S1120.dsc
    ```
Device: Intel DC S3700 Series SSDs
Device: Intel DC S3700 Series SSDs
MBIs

→ A sequential Workload
4) Throughput

- MB per second
- Throughput, Streaming IO
- Interesting block sizes
  - 1MB, 512KB, 256KB
  - Smaller block sizes are more common for IOPS

For block sizes ['1024k', '64k', '8k', '4k', '512']
Make Secure Erase
While not Steady State
  - Sequential read for 1 Minute
  - Sequential write for 1 Minute
Device: Intel DC S3700 Series SSDs
TP Measurement Plot

- **read bs=1024k**
- **read bs=4k**
- **write bs=1024k**
- **write bs=4k**

Bandwidth (MB/s)

Area of Device (in rounds)

Device: HDD 4TB Western Digital WDC WD4000FYYZ-01UL1B0
Saturation
5) Write Saturation

- Continuous writes to the device
- Write enough data
- Show how stable the device is
Devices
Left - Intel DC S3700 Series SSDs,
Right - Intel SSD 520
6) Latency

- Jobs and threads
  - Set to 1!
- DV
  - Average Latency 4KB writes
Devices
Left – Intel DC S3700 Series SSDs,
Right – Intel SSD 320
Device: STEC s1120 PCIe Accelerator
... because as we know, there are known knowns; ....

We also know there are known unknowns; ...

But there are also unknown unknowns -- the ones we don't know we don't know.

(Donald H. Rumsfeld)

http://de.wikipedia.org/wiki/There_are_known_knowns
The Where

- http://git.thomas-krenn.com/TKperf_v1.git
- http://git.kernel.dk/?p=fio.git;a=summary
Pitfalls

- Compression
  - Sandforce Controller!
  - Fio: refill_buffers
6) Resumé

1. Stick tricky to find a correct setup
2. No perfect device
3. RAID/Caches
• Presentations
  • The Why and How of SSD Performance Benchmarking

• Pictures
  • https://commons.wikimedia.org/wiki/File:04KJER0243.jpg?uselang=en-gb
  • https://commons.wikimedia.org/wiki/File:Soca.jpg?uselang=en-gb
  • Augustinushaus Würfel
  • https://commons.wikimedia.org/wiki/File:Cgs_fat.JPG?uselang=en-gb
  • https://commons.wikimedia.org/wiki/File:Rusty_tools.JPG
Backup Slides
• Synchronous IO-Engine \(^1\)
  • Sync Engine → iodepth = 1
  • Submission = Completion
  • Application level: IO is done when system call returns
    – Read: IO is done by device
    – Write: Page Cache
  • Device level
    – Means not O_SYNC → can reside in drive cache

Page Cache

VFS

Block I/O Layer
optional stackable devices on top of "normal" block devices - work on bios

BIOS (Block I/O)
Steady State

```python
stdyState = True
maxY = max(ys)
minY = min(ys)
avg = sum(ys)/len(ys)#calc average of values
#allow max excursion of 20% of average
avgRange = avg * 0.20
if (maxY - minY) > avgRange:
    stdyState = False

#do linear regression to calculate slope of linear best fit
y = np.array(ys)
x = np.array(xs)
A = np.vstack([x, np.ones(len(x))]).T
#calculate k*x+d
k, d = np.linalg.lstsq(A, y)[0]

#as we have a measurement window of 4, we calculate
#the slope excursion in the window
slopeExc = k * self.testMesWindow
if slopeExc < 0:
    slopeExc *= -1
maxSlopeExc = avg * 0.10 #allowed are 10% of avg
if slopeExc > maxSlopeExc:
    stdyState = False
```
<iops>
  <fioversion>"fio 2.0.7\n"</fioversion>
  <numjobs>2</numjobs>
  <iodepth>16</iodepth>
  <roundmat>[[[126, 1087, 2157, 4235, 8052, 12196, 19534, 34917], [131, 976, 1916, 3717, 7101, 10179, 15669, 28266], [88, 600, 1153, 2230, 3870, 7975, 8766, 17704], [74, 544, 1013, 2137, 4117, 6277, 7410, 10754], [26, 261, 531, 999, 2524, 2999, 5101, 10291], [41, 415, 436, 865, 1390, 2626, 4758, 10799], [70, 635, 560, 666, 2185, 3231, 4625, 14559]], [...]
  [[[127, 1094, 2162, 4207, 7325, 10454, 13402, 20640], [130, 1032, 1920, 3130, 4813, 5161, 7352, 16104], [45, 565, 1265, 2485, 4833, 5933, 8703, 18716], [92, 521, 1050, 2017, 3446, 4774, 7641, 16926], [75, 505, 984, 2289, 3992, 6880, 8240, 17015], [94, 776, 1457, 3026, 5400, 9200, 11315, 15993], [102, 936, 1846, 3576, 6533, 10590, 13927, 17903]]]
</roundmat>
  <stdyrounds>[7, 8, 9, 10, 11]</stdyrounds>
  <stdyvalues>[13705, 14639, 14747, 14678, 13927]</stdyvalues>
  <stdyslope>[48.30000000000001042,13904.499999999989]</stdyslope>
  <stdyavg>14339.2</stdyavg>
  <reachstdystate>true</reachstdystate>
  <rndnr>11</rndnr>
</iops>