StorScore system for SSD qualification

Dr. Laura Caulfield
Mark Santaniello
Outline

Introduction
Who we are
Unique needs & opportunities
Motivation

Background
SSD testing “gotchas”

How it works
Recipes: define the test suite
Scores: compare devices and summarize performance
Endurance: measurement of write amplification

Demo
Introduction
Who we are

Cloud server infrastructure engineering

Designers of Open Cloud Server
Part of the Open Compute Project
http://opencompute.org
Unique needs & opportunities

- SSD vendors
- System development (Web search, hosted cloud, etc.)

Microsoft’s platform
Variety / quantity of workloads
Flexibility to modify stack
Co-design SSD & apps
Wide variety of expertise
Additional metrics
Motivation

Need for a component-level storage test (AVL/QCL)

Problems with 3rd-party testing services

- Want test results under Windows, not Linux
- Need results in our server, not a "reference platform"
- Desire to share results with component vendors

Existing tools are insufficient (Iometer, etc.)

- GUI = difficult to automate properly, therefore error-prone
- Microsoft-specific tests (SMART injection)
- Cloud metrics (QoS via 5-nines percentile latency)

Methodology is critical when testing SSDs
Background
SSD gotcha – initial performance

NAND must be erased before it can be programmed

When fresh-out-of-box, all NAND is already erased

Drive contains more NAND than the user-visible space
Overprovisioned (OP) space, typically 7% or 28%

This is an unnatural and ephemeral state

Reads short-circuit if block state is erased (never go to media)
Sustained writes will eventually require garbage collection (GC)

StorScore initializes by writing the entire SSD 2x

Every user-exposed LBA is written twice
Virtually guaranteed to cover any OP space
SSD gotcha – history effect

Previous workload affects *current* workload
Transition phase can take hours

Main causes

- FTL map fragmentation (ex: random → sequential)
- Concurrent garbage collection activity (ex: 100% write → 0% write)
SSD gotcha – history effect, continued

Consistent workload will eventually reach steady-state

1st Pass, 3.5 hours
30 – 100 MB/sec

2nd Pass, 30 minutes
~220 MB/sec
SSD gotcha – detecting steady-state

StorScore includes precondition.exe
Always drives to steady-state *before* measuring performance
Method: rolling linear regression, detect near-zero slope

StorScore performs all sequential tests before any random tests
Minimizes fragmentation and therefore time required to achieve steady-state
SSD gotcha – entropy of written data

Some controllers can compress on-the-fly
Customer data is often compressible
Entropy has a big impact on performance and endurance

StorScore supports variable compressibility
Uses incompressible data by default
Allows use of compressible data in 10% increments

Random Write Throughput: 0%, 20%, 50%
How it works
Recipes – a single test

```c
test(
    name_string => 'foo',
    write_percentage => 0,
    access_pattern => 'random',
    block_size => '8K',
    queue_depth => 32,
    warmup_time => 60,
    run_time => 3600
);
```

The entire contents of single.rcp

Reference the file from the cmd line:
C:\>StorScore --recipe=single.rcp

Reads like English
Recipes – a matrix of tests

```perl
# vim: set filetype=perl:
require 'matrix.rpm';

do_matrix(
    access_patterns => [qw( sequential random )],
    write_percentages => [qw( 100 30 0 )],
    block_sizes => [qw( 2M 1M 512K 64K 16K 8K 4K 1K )],
    queue_depths => [qw( 256 64 16 4 1 )],
    warmup_time => 60,
    run_time => 3600
);

include 'targeted_tests.rcp';
```

Mimics test designer’s whiteboard sketch

*Include* statements combine test files

Full functionality of Perl

```perl
do_workload( "Targeted Test Read Baseline" );

bg_exec( "smart_loop.cmd $gc{'target_physicaldrive'}" );
do_workload( "Targeted Test SMART Read Data " );
bg_killall();
```
Results parser

Raw output files ➔ one Excel file

Dectes, highlights outliers

Easy pivot tables & graphs

Still too much data

Example policy:
Bandwidth matters a lot, latency matters a little

Device A scores 72/100
Device B scores 65/100

24 SSDs
× 218 workloads
5,232 files

Table:

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Write Mix</th>
<th>Access Size (kB)</th>
<th>Access Type</th>
<th>Queue Depth</th>
<th>Bandwidth (MB/s)</th>
<th>Average Latency (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device A</td>
<td>100%</td>
<td>16</td>
<td>random</td>
<td>1</td>
<td>54.32</td>
<td>1.04</td>
</tr>
<tr>
<td>Device B</td>
<td>100%</td>
<td>16</td>
<td>random</td>
<td>1</td>
<td>15.05</td>
<td>0.29</td>
</tr>
<tr>
<td>Device A</td>
<td>30%</td>
<td>16</td>
<td>random</td>
<td>1</td>
<td>20.01</td>
<td>1.39</td>
</tr>
</tbody>
</table>
## Putting the “score” in StorScore

### Goal
Enable data-driven decisions throughout the company

### Reduce data to 1 score / drive

### Method
A weighted average of all the metrics for each workload

---

#### Step 1:
Convert each value to z-score

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Write Mix</th>
<th>Access Size (kB)</th>
<th>Access Type</th>
<th>Queue Depth</th>
<th>Bandwidth (MB/s)</th>
<th>Average Latency (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device A</td>
<td>100%</td>
<td>16</td>
<td>random</td>
<td>1</td>
<td>Z_AX0</td>
<td>Z_AX1</td>
</tr>
<tr>
<td>Device B</td>
<td>100%</td>
<td>16</td>
<td>random</td>
<td>1</td>
<td>Z_BX0</td>
<td>Z_BX1</td>
</tr>
<tr>
<td>Device A</td>
<td>30%</td>
<td>16</td>
<td>random</td>
<td>1</td>
<td>Z_AY0</td>
<td>Z_AY1</td>
</tr>
</tbody>
</table>
Calculating z-scores

A z-score (or standard score) is the number of standard deviations from the mean.

Drive: A
Workload: X (4k, rand, QD1, 100% writes)
Metric: 0 (Read Latency)

One z-score per data point
Positive = better than average
Negative = worse than average

Based on cohort of drives
Applying scoring policies

**General Policy:**

Throughput Metrics

\[ 50\% \times Z_{A(n+m)i} \]  

Latency Metrics

\[ 50\% \times Z_{A(n+m)j} \]  

\[ + \]  

\[ Z_{A(n+m)i} \]  

\[ + \]  

\[ Z_{A(n+m)j} \]  

\[ = \]  

\[ 70 / 100 \]  

- Drive A  
  - Wkld range 0 to (n+m)  
  - Metric range 0 to i  

- Score for Drive A

**Policy to Favor Mixed Workloads:**

70/30 Read/Write Mix Workloads  

\[ 5 \times Z_{An(i+j)} \]  

100% Read & 100% Write Workloads  

\[ 1 \times Z_{Am(i+j)} \]  

\[ = \]  

\[ 65 / 100 \]  

- Can apply multiple policies at once  
- Can use any kind of weight system (stay consistent within policy)
H is so close, and they’ve got a great price. How do we tweak the drive or application to make it work?

If you’ve been using L, A-I will be comparable or better.

A is best-in-class.

E & N are stragglers.

H is so close, and they’ve got a great price. How do we tweak the drive or application to make it work?
H is so close, and they’ve got a great price. How do we tweak the drive or application to make it work?

Answer:
Drive should improve random mix (not seq. mix), or App should favor sequential mix (not random mix)
SSD failure mechanism: writes

Drive Writes Per Day (DWPD)
Total Bytes Writes (TBW)
Drive Writes (DW)

Host Writes

Program / Erase Cycles (P/E Cycles, or PEC)
Write / Erase Cycles (W/E Cycles)

SSD

Controller

Controller Writes

NAND

Total Drive Writes = \times \text{Write Amplification Factor} \times \text{P/E Cycles}
Workload Dependent,
Vendor Reported, Implementation Specific

Previously Available

<table>
<thead>
<tr>
<th>SMART “Media Wear Indicator”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reported in units of 1%</td>
</tr>
<tr>
<td>(300 TB for 30k, 1TB drive)</td>
</tr>
<tr>
<td>4.7 months for 1 workload</td>
</tr>
</tbody>
</table>

New Telemetry

<table>
<thead>
<tr>
<th>SMART “Controller Writes”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reported in units of sectors or GB</td>
</tr>
<tr>
<td>1,700 workloads in 4.7 months</td>
</tr>
</tbody>
</table>
Endurance Results

Drive Writes Per Day (3 years)

Reported Range

Queue Depth

Write Size (kB)

Access Pattern

random

sequential

Device D
Device E
Device F
Device H
Device J
Endurance Results

Not all sequential workloads achieve High Endurance

Identify problem workloads

Reported Range
Demo
Thanks!

- Download StorScore
  - http://aka.ms/storscore

- Questions?