

# Utilizing VDBench to Perform IDC AFA Testing

Michael Ault, IBM  
Oracle FlashSystem Consulting Manager

- ◆ The material contained in this tutorial is copyrighted by the SNIA unless otherwise noted.
- ◆ Member companies and individual members may use this material in presentations and literature under the following conditions:
  - ◆ Any slide or slides used must be reproduced in their entirety without modification
  - ◆ The SNIA must be acknowledged as the source of any material used in the body of any document containing material from these presentations.
- ◆ This presentation is a project of the SNIA Education Committee.
- ◆ Neither the author nor the presenter is an attorney and nothing in this presentation is intended to be, or should be construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion please contact your attorney.
- ◆ The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved. The author, the presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.

**NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.**

## ➤ Utilizing VDBench to Perform IDC AFA Testing

- ◆ This session will review what an all-flash-array consists of, what IDC is and what their testing involves and how to use VDBench to perform AFA testing and comparisons.

# Agenda

- What is an AFA?
- What is VDBench?
- What is the IDC and why do we care?
- What are the suggested AFA tests?
- Implementing the IDC AFA tests with VDBench

# What is an AFA?

- All-Flash-Array
- Not SSD (form-factor flash drives)
- Built from Function-designed flash modules
- Integrated control, configuration and monitoring
- S4-TWG defines AFA as a solid state storage array connected by redundant network protocols

# What is VDBench?

- Command line system
- Simulates various load scenarios including multiple hosts
- Doesn't require an Oracle install
- Scriptable
- Provided free from Oracle
- Vdbench has been tested on Solaris Sparc and x86, Windows NT, 2000, 2003, 2008, 2012, XP and Windows 7+8, HP/UX, AIX, Linux, Mac OS X, zLinux, and native VmWare
- Download from:

<http://www.oracle.com/technetwork/server-storage/vdbench-downloads-1901681.html>

# VDBench Download Page

**ORACLE**

Sign In/Register Help Country ▾ Communities ▾ I am a... ▾ I want to... ▾ Search

Products Solutions Downloads Store Support Training Partners About **OTN**

Oracle Technology Network > Server and Storage Systems

- Engineered Systems
- Ksplice
- Oracle Linux
- Oracle Optimized Solutions
- Oracle VM
- Oracle VM VirtualBox
- SAN Storage
- Secure Global Desktop
- Server Management Tools
- Solaris 10
- Solaris 11
- Solaris Cluster
- Solaris Studio
- SPARC Servers
- StorageTek Tape Storage
- Sun Blade 6000 Modular Systems
- Sun Desktops & Peripherals
- Sun Flash Storage
- Sun NAS Storage
- Sun Netra Carrier-Grade Systems
- Networking and Data Center Fabric Products

## Vdbench Downloads

Vdbench is a command line utility specifically created to help engineers and customers generate disk I/O workloads to be used for validating storage performance and storage data integrity. Vdbench execution parameters may also specified via an input text file.

You must accept the [OTN License Agreement](#) to download this software.

Accept License Agreement |  Decline License Agreement

---

### Vdbench (5.04.02)

- [vdbench50402.zip](#)
- User Guide
- Vdbench Discussion Forum

### Vdbench (5.04)

- [vdbench50401.zip](#)
- User Guide
- Vdbench Discussion Forum

### Related Downloads

- Vdbench source code

- IDC provides guidelines on standard tests to use in comparing computer technologies
- IDC has provided a detailed test guidelines for AFA
- Using the IDC guidelines you can easily test and compare AFAs
- The public link for the IDC testing guidelines for AFAs is:  
<http://idcdocserv.com/251951>



# Differences Between AFA and HDD

- ❖ Preconditioning. Flash-based array performance will differ significantly from FOB and after first write on every flash cell. HDDs don't exhibit the FOB effect.
- ❖ Read/write asymmetries. Traditional HDD overwrites data in place and does not lock data on writes. Flash needs to erase before rewriting and cell locks on writes causing writes to be slower
- ❖ Endurance. After a defined number of P/E cycles, flash will become read only and ultimately inoperable.
- ❖ Data stream/data set definition. The mixed virtual workloads in 3rd Platform computing are very different than traditional client/server workloads. For example: A relevant workload will exhibit a variety of read/write ratios, use a wide distribution of block sizes, is heavily skewed toward random I/O, has a high percentage of data that is reducible.
- ❖ Load generation. Flash-based arrays generally deliver at least 10 times better performance. Load generators have to take this into account

# Some Flash Terminology

- ❖ P/E cycles. Each time an individual flash cell is erased and programmed (i.e., written again), the flash cell incurs a slight amount of damage known as wear. A single iteration of erasing and writing to a flash cell is known as a "program/erase cycle" (P/E cycle).
- ❖ Asymmetric behavior. Although on average flash storage delivers latencies that are at least an order of magnitude lower than HDDs, read/write I/O can exhibit different latencies depending on the state of flash media at the time an I/O is attempted. If the page is available to be written but has data in it that must be overwritten, a P/E cycle must occur resulting in increased latency.

# Some Flash Terminology

- Overprovisioned capacity. Each SSD, flash module, or flash-based array has a set amount of capacity reserved for administrative activities associated with flash performance, endurance, or reliability.
- Wear leveling. Flash cells have a specified wear rating. If certain cells are used frequently, they may wear out significantly sooner than other cells in the same array.

# Some Flash Terminology

- Free space management. To write even a single byte, an entire erase block must be available. "garbage collection," works periodically to ensure that systems always have a ready supply of available blocks. As data gets invalidated on the flash media, it is not immediately overwritten but rather marked as "available" to be written.
- Write amplification. Relative to write I/O requested by an application, there are two types of processes in most flash-based arrays that amplify the number of writes that actually occur within the array: garbage collection and data protection. The ratio of FTL writes to Server requests

# Some Flash Terminology

- Flash translation layer (FTL). Flash storage is generally combined with an FTL (FPGA, ASIC, or software based) that provides a logical block interface for operating systems to access the flash media as if it were a traditional HDD.

In the universe of performance testing, there are five basic types of tests that could potentially be performed, and these are ranked in order of relevance to understanding how an array will perform on your actual production workload:

- **Class A:** Workload testing using your actual data streams, data sets, and workflows
- **Class B:** Workload testing using data streams, data sets, and workflows that are generated using application-specific testing tools like SLOB (Oracle), Jetstress (Exchange), Login VSI (VDI), and others
- **Class C:** Workload testing that uses data streams, data sets, and workflows generated by general-purpose workload generation tools like Load DynamiX, FIO, or vdbench that are capable of very accurate modeling
- **Class D:** An intelligent performance corners test using generic data sets, data streams, and workflows designed to closely model 3rd Platform computing workloads
- **Class E:** Hero testing that encompasses most classic performance corners testing

In AFA testing, IDC recommends Class D type tests

# IDC Test Plan

## AFA Test Plan Summary

Phase	Description	Tasks	Estimated Elapsed Time
1	Setup and preconditioning	2 x 85% overwrite	24 hours
	Data set preparation	2 x 5% overwrite 5 x 1% overwrite	6 hours
2	Baseline workload testing	Develop scripts	6 hours
		Thread/queue depth optimization	6 hours
		IOPS ramp testing (4 runs at 3 hours per run)	12 hours
3	Functional testing	Document workflows	6 hours
		Test workflows (4 runs at 4 hours per run)	16 hours
4	Fault injection testing	Inject failures (4 runs at 2 hours per run)	8 hours
5	Soak test	Run 4 baseline workloads consecutively(12 hours each)	48 hours
			132 hours

Source: IDC, 2014

# Testing the a 40 TiB System

## ➤ Preconditioning

- ◆ Actual amount of flash is 65.9 TB
- ◆ System 40 usable TiB (37.5 TB after RAID5)
- ◆ We already have extracted the “overhead” so we will configure the 37 into 37 -1 TiB LUNS
- ◆ Preconditioning volume of writes would be  $65.9 * 0.85 * 2 = 112$  TB
- ◆ At 128K blocks the throughput for the 840 is 1.7 GB/s for a single Power8 server with 16 threads per LUN (592 total)
- ◆ 67,500 seconds to do writes (~19 hours)
- ◆ Monitor for write cliff, if not reached at end of first run, repeat



# Testing the a 40 TiB System

## ➤ Data Set Preparation and Aging

- ◆ 2 additional runs with small random writes (4-16kb) each at 5% of 65.9 TB so 3.3 TB each run randomly spread across system (3 runs of around 1,100 GiB one at 4k, 8k and 16k)
- ◆ 5 additional runs with small random writes (4-16kb) each at 1% of 65.9 TiB (660 GB) each run randomly spread across system (3 streams 4, 8 and 16k each doing 220 GB each) each run addressing different sets of luns.

# VDBench Scripting

- Utilizes tags for each script section
- SD for drive/storage designations
- WD for workload definitions
- RD for run definitions

# Preconditioning Script

Create several luns, do 37-1.0 TiB size that used most of the 37.5 TiB.

Contents of precondition.cfg:

```
sd=default, openflags=o_direct
```

```
sd=sd1,lun=\\.\PhysicalDrive1
```

```
sd=sd2,lun=\\.\PhysicalDrive2
```

```
sd=sd3,lun=\\.\PhysicalDrive3
```

```
...
```

```
sd=sd37,lun=\\.\PhysicalDrive37
```

```
wd=fill,sd=sd*,rdpct=0,xfersize=128k,seekpct=-1
```

```
rd=precondition,wd=fill,iorate=max,elapsed=67500,interval=30,
```

```
threads=16
```

# Running the Script

- Use the command line
- Specify output location if you want a specific location for the output HTML files

```
\vdbench\vdbench -f precondition.cfg -m 37 -o  
\vdbench\output\precon
```

# Example Output Files

The screenshot shows a Windows File Explorer window with the address bar displaying the path: Computer > Local Disk (C:) > IBM > IDC Testing > IDC\_testing > IDC\_testing > Day2\_output. The left sidebar shows the 'Favorites' section with Desktop, Recent Places, Downloads, OneDrive, and RealPlayer Cloud. The main pane displays a list of files with columns for Name, Date modified, Type, and Size.

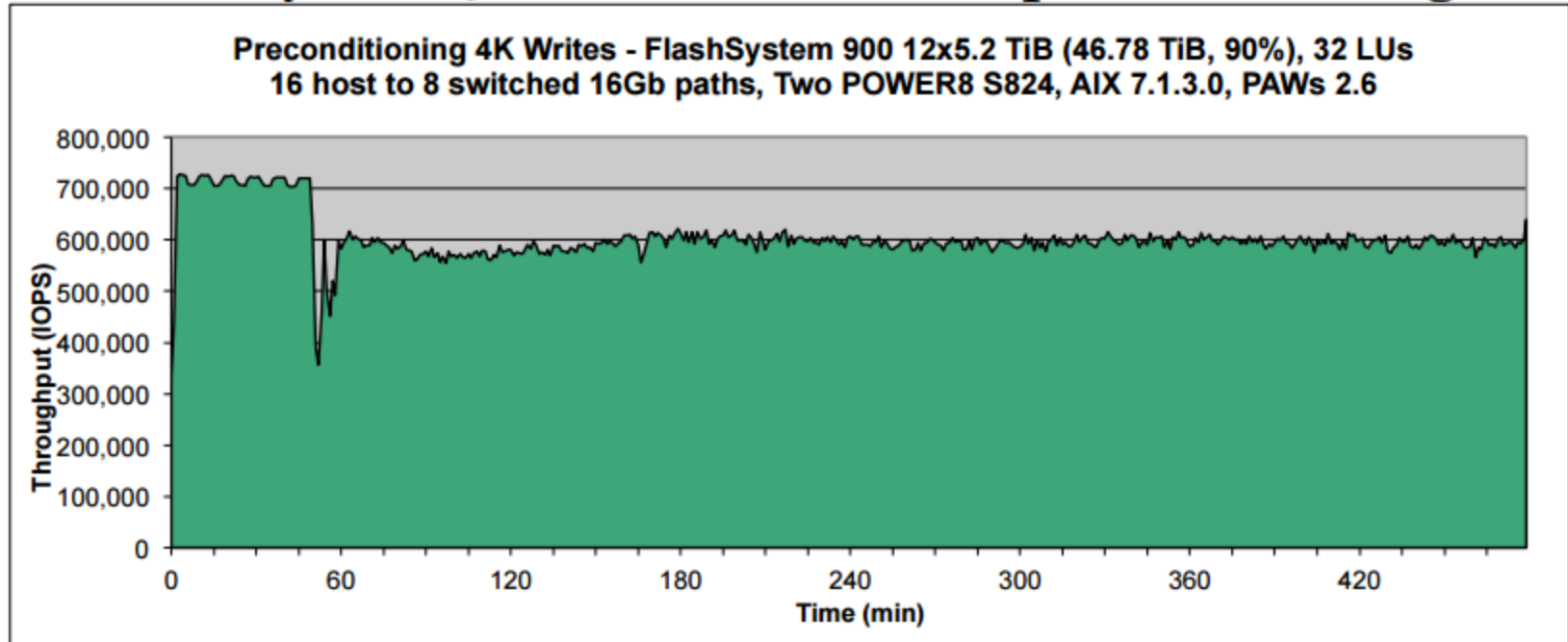
Name	Date modified	Type	Size
errorlog.html	9/16/2014 4:21 PM	Chrome HTML Do...	1 KB
flatfile.html	9/16/2014 7:40 PM	Chrome HTML Do...	3,684 KB
histogram.html	9/16/2014 7:40 PM	Chrome HTML Do...	739 KB
localhost.html	9/16/2014 7:40 PM	Chrome HTML Do...	1,502 KB
localhost-0.histogram.html	9/16/2014 7:40 PM	Chrome HTML Do...	740 KB
localhost-0.html	9/16/2014 7:40 PM	Chrome HTML Do...	1,501 KB
localhost-0.stdout.html	9/16/2014 7:40 PM	Chrome HTML Do...	255 KB
localhost-1.histogram.html	9/16/2014 7:40 PM	Chrome HTML Do...	740 KB
localhost-1.html	9/16/2014 7:40 PM	Chrome HTML Do...	1,501 KB
localhost-1.stdout.html	9/16/2014 7:40 PM	Chrome HTML Do...	255 KB
localhost-2.histogram.html	9/16/2014 7:40 PM	Chrome HTML Do...	740 KB
localhost-2.html	9/16/2014 7:40 PM	Chrome HTML Do...	1,501 KB
localhost-2.stdout.html	9/16/2014 7:40 PM	Chrome HTML Do...	255 KB
localhost-3.histogram.html	9/16/2014 7:40 PM	Chrome HTML Do...	740 KB

# File Contents (Summary file)

Sep 16, 2014	interval	i/o rate	MB/sec 1024**2	bytes i/o	read pct	resp time	read resp	write resp	resp max	resp stddev	queue depth	cpu% sys+u	cpu% sys
16:19:08.058	1	74950.00	36.60	512	10.04	0.341	0.541	0.318	426.042	1.776	25.5	0.0	4.3
16:19:09.018	2	103950.00	50.76	512	10.01	0.258	0.437	0.238	6.256	0.257	26.8	8.2	6.4
16:19:10.016	3	118267.00	57.75	512	10.13	0.233	0.464	0.207	46.793	0.540	27.5	13.6	7.6
16:19:11.030	4	121366.00	59.26	512	9.88	0.230	0.471	0.204	195.091	0.932	27.9	12.1	8.2
16:19:12.020	5	110907.00	54.15	512	9.93	0.245	0.425	0.225	6.435	0.258	27.2	5.1	5.8
16:19:13.017	6	105364.00	51.45	512	9.99	0.261	0.427	0.243	7.380	0.248	27.5	8.3	5.4
16:19:14.016	7	125217.00	61.14	512	9.95	0.220	0.474	0.192	29.523	0.637	27.5	10.2	8.2
16:19:15.030	8	116730.00	57.00	512	10.10	0.239	0.486	0.212	149.333	0.830	27.9	12.5	7.8
16:19:16.018	9	117267.00	57.26	512	10.09	0.232	0.423	0.210	11.671	0.265	27.2	8.0	7.1
16:19:17.016	10	126531.00	61.78	512	10.01	0.218	0.462	0.190	16.961	0.605	27.5	10.4	7.5
16:19:18.002	11	111330.00	54.36	512	10.04	0.247	0.426	0.227	11.312	0.261	27.5	9.5	7.5
16:19:19.016	12	125546.00	61.30	512	10.15	0.219	0.471	0.191	62.544	0.619	27.5	12.8	8.6
16:19:20.017	13	108123.00	52.79	512	10.06	0.255	0.437	0.234	13.263	0.272	27.5	6.6	4.7
16:19:21.016	14	118485.00	57.85	512	10.03	0.233	0.459	0.207	16.959	0.565	27.6	10.2	7.4
16:19:22.030	15	127885.00	62.44	512	9.98	0.218	0.460	0.192	31.209	0.660	27.9	12.0	9.9
16:19:23.018	16	108483.00	52.97	512	9.99	0.251	0.449	0.229	301.127	0.958	27.2	7.7	5.8
16:19:24.016	17	108807.00	53.13	512	9.77	0.253	0.494	0.227	78.131	0.903	27.6	10.8	6.3
16:19:25.018	18	114688.00	56.00	512	9.96	0.240	0.423	0.220	9.425	0.248	27.5	9.0	6.7
16:19:26.016	19	116589.00	56.93	512	10.01	0.236	0.486	0.208	31.922	0.648	27.5	8.1	6.7
16:19:27.017	20	109109.00	53.28	512	9.98	0.252	0.429	0.233	13.926	0.265	27.5	11.8	6.0
16:19:28.016	21	123767.00	60.43	512	9.99	0.222	0.448	0.197	15.685	0.566	27.5	8.4	8.5
16:19:29.030	22	124040.00	60.57	512	9.90	0.225	0.479	0.197	112.039	0.749	27.9	13.2	8.7
16:19:30.018	23	113921.00	55.63	512	9.94	0.238	0.414	0.219	12.076	0.254	27.2	8.4	7.9

# Write Cliff (Preconditioning is over)

## FlashSystem 900: Random Write preconditioning



Max write IOPS = 700+ K IOPS (sustained over 45 mins)

Long-term write IOPS is about 600 K IOPS

*Note that this long-term IO rate is consistent with the 100% 4KB write response time curve in later pages*

# Phase 1: Data Conditioning and Aging

- **Conditioning is 2 passes:**
  - ◆ 5:1 reducible (compressible)
  - ◆ Sequential small writes randomized in size around average expected I/O size
  - ◆ 5% of data volume
- **Aging is 5 passes:**
  - ◆ 5:1 reducible (compressible)
  - ◆ Random small writes randomized in size around average expected I/O size
  - ◆ 1% of data volume



# Data Conditioning Script

compratio=5

\*SD:

sd=default, openflags=o\_direct

sd=sd1,lun=\\.\PhysicalDrive1

sd=sd2,lun=\\.\PhysicalDrive2

sd=sd3,lun=\\.\PhysicalDrive3

...

sd=sd37,lun=\\.\PhysicalDrive37

\*WD: Workload Definitions

wd=wd1,sd=sd\*,seekpct=sequential

\*RD: Run Definitions

\*Will run 1 test IO size with random (4k, 8k, 16k) for approximately 3300 GB

rd=rd1,wd=wd1,iorate=max,elapsed=1950,interval=5,rdpct=0,xfersize=(4k,128k,4k),threads=(16)

Run this 2 runs, the sequential IO specification localizes the data placement across 5% of the system, the 3200 elapsed is based on the IOPS for 16K size divided into the 3.3 GB (5%).

# Data Aging Script

compratio=5

\*SD:

sd=default, openflags=o\_direct

sd=sd1,lun=\\.\PhysicalDrive1

sd=sd2,lun=\\.\PhysicalDrive2

sd=sd3,lun=\\.\PhysicalDrive3

...

sd=sd37,lun=\\.\PhysicalDrive37

\*WD: Workload Definitions

wd=wd1,sd=sd\*,seekpct=random

\*RD: Run Definitions

\*Will run 1 test IO size with random (4k, 128k, 16k) for approximately 660 GB

rd=rd1,wd=wd1,iorate=max, elapsed=390, interval=1, rdpct=0, xfersize=(4k,128k,16k),  
threads=(16)

Run this 5 runs, the sequential IO specification localizes the data placement across 5% of the system, the 382 elapsed is based on the IOPS for 16K size divided into the 660 GB (1%).

## Phase 2: Baseline Workload Tests

- Once the AFA is brought to a state that is considered a start point, actual testing can start
- The next test is the baseline workload
  - ◆ 20/80 read/write ratio, all sequential, mixed block sizes matching your actual distribution, data that exhibits a 5:1 data reduction ratio, and displays some I/O banding with drift
  - ◆ 65/35 read/write ratio, all random, mixed block sizes matching your actual distribution, data that exhibits a 5:1 data reduction ratio, and displays some I/O banding with drift
  - ◆ 35/65 read/write ratio, all random, mixed block sizes matching your actual distribution, data that exhibits a 5:1 data reduction ratio, and displays some I/O banding with drift
  - ◆ 80/20 read/write ratio, all sequential, mixed block sizes matching your actual distribution, data that exhibits a 5:1 data reduction ratio, and displays some I/O banding with drift

# Banding and Drift

- Also known as spatial and temporal data
- Spatial is sequential data
- Temporal is data that is accessed near the same time
- Buffering/caching usually deals best with temporal data

# Allowing for Banding and Drift

## These modifications allow vdbench tests simulate the effects of Banding and Drift

### ◆ **compratio=2.3**

- ◆ Result: Data generated by vdbench will be reducible by about 65%
- ◆ Reason: IBM recommends to only enable compression for compressible workloads

### ◆ **Hotband=(x,y) with x and y representing location ranges for each storage definition**

- ◆ Result: I/O is concentrated within the specified data ranges
- ◆ Reason: I/O concentration more closely resembles real-workload temporal locality

### ◆ **rhpct=50, whpct =25**

- ◆ Result: vdbench I/O patterns produce cache hits to the values specified
- ◆ Reason: More representative of the cache hits that result from RTC optimizations with temporal locality

# Writes in AFA

- Usually large writes will be broken up
- 4-8K writes are common
- A large write that would be written sequentially to HDD will be “shot gunned” to all available free blocks across several flash modules due to wear leveling

# 20/80 and 80/20 Read Percent Sequential

**compratio=2.3**

\*SD: Storage Definitions

sd=default, openflags=o\_direct

sd=sd1,lun=\\.\PhysicalDrive1

sd=sd2,lun=\\.\PhysicalDrive2

sd=sd3,lun=\\.\PhysicalDrive3

...

sd=sd37,lun=\\.\PhysicalDrive37

\*WD: Workload Definitions

wd=wd1,sd=sd\*,seekpct=sequential

\*RD: Run Definitions

\*Will run 40 tests (10 for 8k, 10 for 16k, 10 for 32k, 10 for 128k) for 120 seconds each  
(2\*4\*5)

rd=rd1,wd=wd1,iorate=max,elapsed=300,interval=1,pause=5,forrdpct=(20,80),forxfersize=(  
8k,16k,32k,128k),forthreads=(1,16,d), **Hotband=(x,y), rhpct=50, whpct =25**

# 35/65 and 65/35 Read Percent Random Workloads

compratio=2.3

\*SD: Storage Definitions

sd=default, openflags=o\_direct

sd=sd1, lun=\\.\PhysicalDrive1

sd=sd2, lun=\\.\PhysicalDrive2

sd=sd3, lun=\\.\PhysicalDrive3

...

sd=sd37, lun=\\.\PhysicalDrive37

\*WD: Workload Definitions

wd=wd1, sd=sd\*, seekpct=random

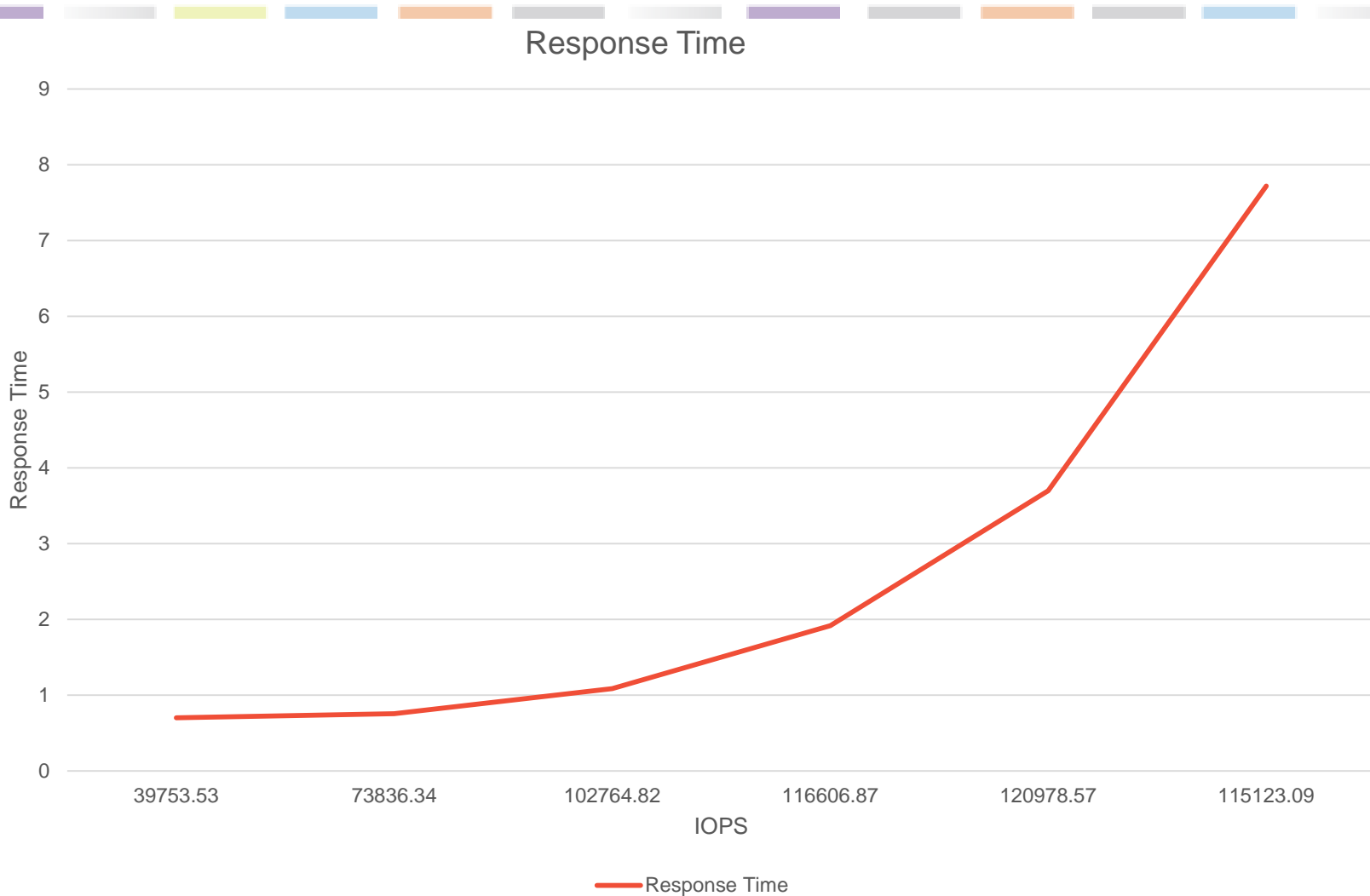
\*RD: Run Definitions

\*Will run 40 tests (10 for 8k, 10 for 16k, 10 for 32k, 10 for 128k) for 120 seconds each

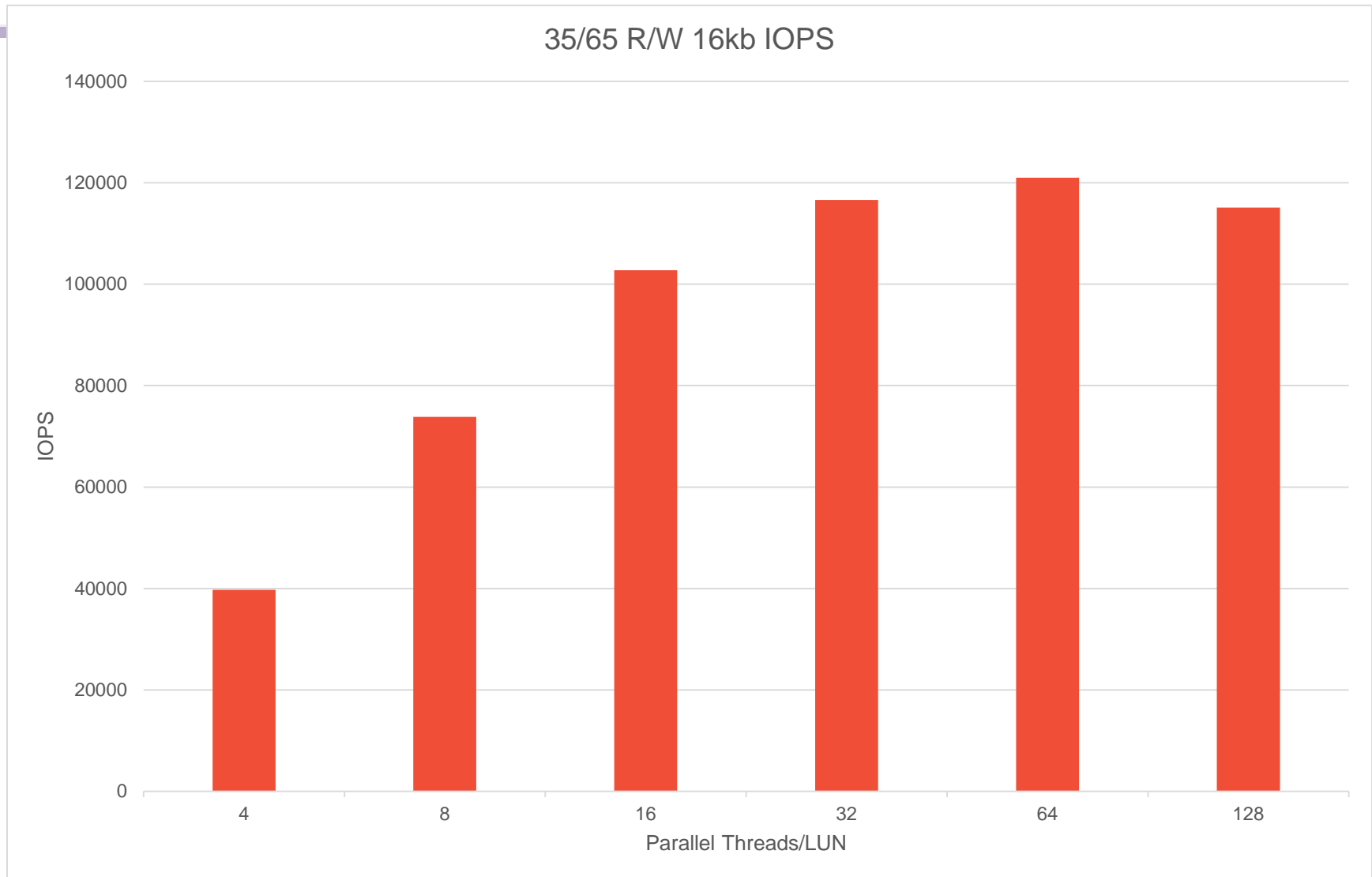
rd=rd1, wd=wd1, iorate=max, elapsed=300, interval=1, pause=5, forrdpct=(35,65), for xfersize=(8k,16k, 32k,128k), forthreads=(1,16,d), **Hotband=(x,y), rhpct=50, whpct =25**



# Determine the Inflection Point



# A Better Method May be IOPS



# Phase 3: Functional Testing

- Applies to full SAN type models
- Will be dependent on functions available
  - ◆ Thin provisioning
  - ◆ Snapshots
  - ◆ Replication
  - ◆ Encryption – All 840s do this transparently
  - ◆ Copy
  - ◆ Backup
  - ◆ Etc.
- Not covered in this presentation since it is array specific

# Phase 4: Fault Injection Testing

---

- Determine failure scenarios to test
- Establish base line, steady state mixed workload
- Test Scenarios
- Example Scenarios
  - ◆ Fail a path
  - ◆ Fail a drive (or flash module)
  - ◆ If the array claims to support it, fail two drives (or flash modules) in the same RAID group simultaneously
  - ◆ Fail a controller
  - ◆ Fail a power supply (and fail a fan separately if the array supports that)
  - ◆ Perform a controller firmware upgrade
  - ◆ Expand capacity by adding a shelf online

# 4KB 20/80 load

4kb\_20\_80\_seq.cfg

compratio=2.3

\*SD: Storage Definitions

sd=default, openflags=o\_direct

sd=sd1,lun=\\.\PhysicalDrive1

sd=sd2,lun=\\.\PhysicalDrive2

sd=sd3,lun=\\.\PhysicalDrive3

sd=sd4,lun=\\.\PhysicalDrive4

sd=sd5,lun=\\.\PhysicalDrive5

sd=sd6,lun=\\.\PhysicalDrive6

sd=sd7,lun=\\.\PhysicalDrive7

sd=sd8,lun=\\.\PhysicalDrive8

sd=sd9,lun=\\.\PhysicalDrive9

sd=sd10,lun=\\.\PhysicalDrive10

\*WD: Workload Definitions

wd=wd1,sd=sd\*,seekpct=sequential

\*RD: Run Definitions

\*Will run 1 test 4kb sequential 2 hours long

rd=rd1,wd=wd1,iorate=max,elapsed=7200,interval=1,forrdpct=(20),forxfersize=(4k),

forthreads=(32), **Hotband=(x,y), rhpct=50, whpct =25**

# 8KB 65/35 Load

8kb\_65\_35\_ran.cfg

compratio=2.3

\*SD: Storage Definitions

sd=default, openflags=o\_direct

sd=sd11,lun=\\.\PhysicalDrive11

sd=sd12,lun=\\.\PhysicalDrive12

sd=sd13,lun=\\.\PhysicalDrive13

sd=sd14,lun=\\.\PhysicalDrive14

sd=sd15,lun=\\.\PhysicalDrive15

sd=sd16,lun=\\.\PhysicalDrive16

sd=sd17,lun=\\.\PhysicalDrive17

sd=sd18,lun=\\.\PhysicalDrive18

sd=sd19,lun=\\.\PhysicalDrive19

sd=sd20,lun=\\.\PhysicalDrive20

\*WD: Workload Definitions

wd=wd1,sd=sd\*,seekpct=random

\*RD: Run Definitions

\*Will run 1 test 8kb random 2 hours long

rd=rd1,wd=wd1,iorate=max,elapsed=7200,interval=1,forrdpct=(65),forxfersize=(8k),  
forthreads=(16), **Hotband=(x,y), rhpct=50, whpct =25**

# 64 KB 65/35 Load

## 64kb\_65\_35\_ran.cfg

\*SD: Storage Definitions

```
sd=default, openflags=o_direct  
sd=sd21,lun=\\.\PhysicalDrive21  
sd=sd22,lun=\\.\PhysicalDrive22  
sd=sd23,lun=\\.\PhysicalDrive23  
sd=sd24,lun=\\.\PhysicalDrive24  
sd=sd25,lun=\\.\PhysicalDrive25  
sd=sd26,lun=\\.\PhysicalDrive26  
sd=sd27,lun=\\.\PhysicalDrive27  
sd=sd28,lun=\\.\PhysicalDrive28  
sd=sd29,lun=\\.\PhysicalDrive29  
sd=sd30,lun=\\.\PhysicalDrive30
```

\*WD: Workload Definitions

```
wd=wd1, sd=sd*, seekpct=random
```

\*RD: Run Definitions

\*Will run 1 test 64kb 2 hours long

```
rd=rd1, wd=wd1, iorate=max, elapsed=7200, interval=1, forrdpct=(65), forxfersize=(64k),  
forthreads=(8)
```

# 128 KB 80/20 Load

128kb\_80\_20\_seq.cfg

compratio=2.3

\*SD: Storage Definitions

sd=default, openflags=o\_direct

sd=sd31,lun=\\.\PhysicalDrive31

sd=sd32,lun=\\.\PhysicalDrive32

sd=sd33,lun=\\.\PhysicalDrive33

sd=sd34,lun=\\.\PhysicalDrive34

sd=sd35,lun=\\.\PhysicalDrive35

sd=sd36,lun=\\.\PhysicalDrive36

sd=sd37,lun=\\.\PhysicalDrive37

sd=sd38,lun=\\.\PhysicalDrive38

sd=sd39,lun=\\.\PhysicalDrive39

\*WD: Workload Definitions

wd=wd1,sd=sd\*,seekpct=sequential

\*RD: Run Definitions

\*Will run 1 test 128kb 2 hours long

rd=rd1,wd=wd1,iorate=max,elapsed=7200,interval=1,forrdpct=(80),forxfersize=(128k),fort  
hreads=(4), **Hotband=(x,y), rhpct=50, whpct =25**



# Run Loads

Capture outputs to C:\vdbench\output as follows:

Use 4 different sessions and run one of the following commands in each:

```
c:\vdbench\vdbench -f 4kb_20_80_seq.cfg -o  
c:\vdbench\output\day_three_4kb -m 10
```

```
c:\vdbench\vdbench -f 8kb_65_35_ran.cfg -o  
c:\vdbench\output\day_three_8kb -m 10
```

```
c:\vdbench\vdbench -f 64kb_65_35_ran.cfg -o  
c:\vdbench\output\day_three_64kb -m 10
```

```
c:\vdbench\vdbench -f 128kb_80_20_seq.cfg -o  
c:\vdbench\output\day_three_128kb -m 9
```

# Do Failures

For our testing we did the following:

- Fail a path
- Reboot a controller
- Fail a controller (unrack controller)
- Fail a module
- We found the test only required 1.5 hour so we did these additional tests:
  - Fail a PSU (pull out cord)
  - Fail a battery (pull out battery module)

# Phase 5: Soak Test

- ◆ Use the 4 loads from phase 2
- ◆ Run each for 12 hours (48 consecutive hours)
- ◆ Review results for performance anomalies

# Summary

- Testing and comparing systems can be difficult
- Utilizing IDC guidelines testing is made more consistent
- Using tools like VDBench provides a repeatable and consistent test framework

The SNIA Education Committee thanks the following Individuals for their contributions to this Tutorial.

## Authorship History

Michael Ault, IBM:

Updates: Mike Ault to add banding and drift simulation

Presented:

2/24/2016 USENIX FAST 2/2016

8/8/2016 Flash Memory Summit

## Additional Contributors

*Please send any questions or comments regarding this SNIA Tutorial to [tracktutorials@snia.org](mailto:tracktutorials@snia.org)*