VICOM INFINITY

LEN SANTALUCIA

LSANTALUCIA@VICOMINFINITY.COM

Arty Ecock

Arty.Ecock@gmail.com

LARGER NVME EDEVICE PAGE SPACE ALLOCATION

Agenda

1. Introduction to VICOM-INFINITY
2. NVMe Evaluation Objective(s)
3. Testing Environment and NVMe Configuration
4. PAGING commentary
5. z/VM NVMe EDEVICE Support Overview
6. Performance Testing Methodology and Observations
7. NVMe Fault Failure & Recovery Testing
8. Summary & Conclusion

DRIVING IBM Z INNOVATION AND PLATFORM LONGEVITY

THROUGH LINUX FOUNDATION OPEN MAINFRAME PROJECT LEADERSHIP AND CHAIRPERSONSHIP

Distributions

Supported Versions

ubuntu Supported by Canonical

SUSE

redhat

Community Versions

debian

openSUSE

fedora

CentOS

Virtualization

KVM

VM

LPAR

DPM

Docker)

LXD (Ubuntu)

Languages

python

Ruby

php

R

ERLANG

Scala

Clojure

JS

OCaml

Java

Swift

Runtimes

nodeJS

RAILS

Zend framework (PHP)

OpenJDK

LLVM

Apache Tomcat

Management

CHEF

puppet labs

ANSIBLE

juju

openstack

kubernetes

vmware vRealize

urban{code}

Database

MariaDB

PostgreSQL

mongoDB

cassandra

IBM Cloudant

redis

MySQL

ORACLE Diamond Partner

DB2

Analytics

Spark

elasticsearch

Apache Solr

BLU Acceleration

THE LINUX FOUNDATION

OPEN MAINFRAME PROJECT

FULL RANGE OF SERVICES FOR IBM Z SYSTEMS

- Architect and Design
- Capacity Planning & Modeling
- Disaster Recovery Planning & Implementation
- Installation Planning & Implementation
- Software Migration & Installation
- System Upgrade, Migration, & Conversion Services
- Pervasive Encryption
- Parallel Sysplex
- IBM Maintenance Services
- IBM Software & Defect Support Services
- IBM Professional Services
- System Tuning
- Training
- Staff Augmentation
- Modernization

FULL RANGE OF SERVICES FOR IBM STORAGE SYSTEMS

- Architecture and Design
- Capacity Planning & Modeling
- Disk/Tape/SAN/NAS Migration Planning, Management, & Performance Tuning
- Disaster Recovery Planning & Implementation
- Installation Planning & Implementation
- Data Migration
- Data Recovery
- Safeguarded Copy
- Data Center Fail Over, Stay and Return

CUSTOMER SUCCESS STORIES…AND MANY MORE

Morgan Stanley

Memorial Sloan Kettering Cancer Center

DTCC
Securing Today. Shaping Tomorrow.

BNY MELLON

Verisk

Supreme Court of Virginia Richmond

ADP

Credit Suisse

NEW YORK STATE

AIG

ca technologies

Police Department City of New York

Brown Brothers Harriman

conEdison

NVMe EVALUATION OBJECTIVES

Evaluate Larger NVMe EDEVICE Page Space Allocations

- Target size: 750GB or larger
- Use PAV aliases for NVMe paging

NVMe = “Non-Volatile Memory express”

NVMe is the storage protocol (think of iSCSI)

All NVMe devices are SSD, but not all SSD are NVMe

TESTING ENVIRONMENT

"Development" LPAR

Minor Production workload - 2 web servers

Several Linux guests (RHEL, Ubuntu, Alma), VSEn

z15 T02 (BC), 40G, 8 IFLs

DS8k storage – DS8882 model 5334-983, 16GB GBICs,

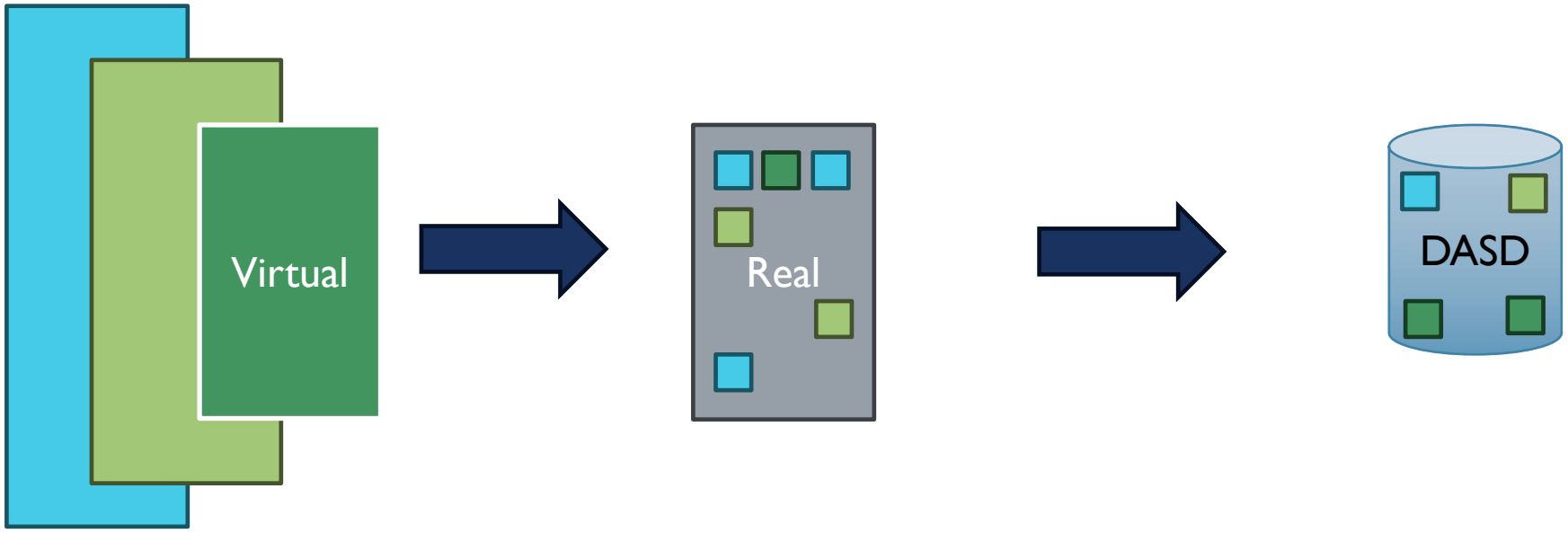
4 FICON paths to DASD, 2 FCP paths to Brocade Switch

NVMe storage – (2) Intel SSDPE2KX040T801 (4 TB each)

PAGING

"PAGING" is the workhorse behind Storage Virtualization

- Virtual Storage presents the illusion of large, contiguous Address Spaces
  - PAGING allows Real Storage to appear dimensionally transcendent



PAGING

How much PAGE space do you need?

- ”It depends” – Bill Bitner
  - What is the virtual memory footprint of all guests that must run simultaneously?
  - Other factors include VDISK usage
  - Add 10-15% (of real storage size) for NSS, PGMBK, CP Directory
  - Add additional for future growth
  - (Use Bruce Hayden’s VIR2REAL package from the VM Downloads Page)
- “Enough”
  - “I’ll never exceed a certain virtual memory capacity”
- “None”
  - “I have sufficient real memory to avoid paging”

PAGING

What happens when PAGE space is exhausted?

- You PAGE to SPOOL
- Your system ABENDs
- (You look for a new job)

PAGING

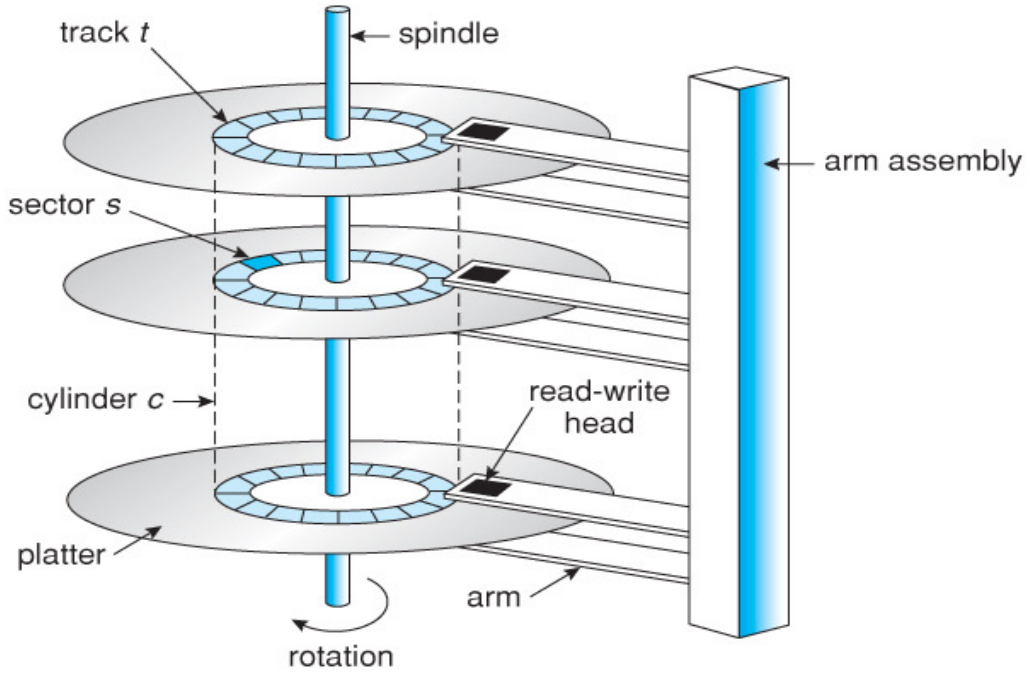
PAGING “costs”

- DASD – Hardware costs
- How much overcommit (TVR) must you maintain? – Return On Investment
  - 2:1, 2.5:1, even 3:1 desirable
- How efficient (in terms of CPU overhead) is PAGING?
- If the system crashes due to PAGE space exhaustion, what is the financial impact to my business?
  - (What is the “social” impact to my business? Can further investment erase the stigma?)

Correlation/Harmonization between costs and availability

(MY) PAGING DEVICES

"A resplendent mélange of mechanical engineering and electronics"



(MY) PAGING DEVICES

(E)CKD – (Extended) Count Key Data

- Data can be located using a user-defined "key"
- Variable-length records, variable length keys (very space efficient)
- Resulted in less CPU and memory requirements

FBA – Fixed Block Architecture

- Data is always contained in fixed-length blocks
- Blocks are addressed by their relative block number

(MY) PAGING DEVICES

IBM 2305-1 FIXED HEAD STORAGE "DRUM" (CKD)

3 MB/sec

5.4 million bytes/module

(MY) PAGING DEVICES

IBM 3310 DASD (FBA)

1 MB/sec

64.5 MB/drive (actuator)

(MY) PAGING DEVICES

IBM 3330 DASD (CKD)

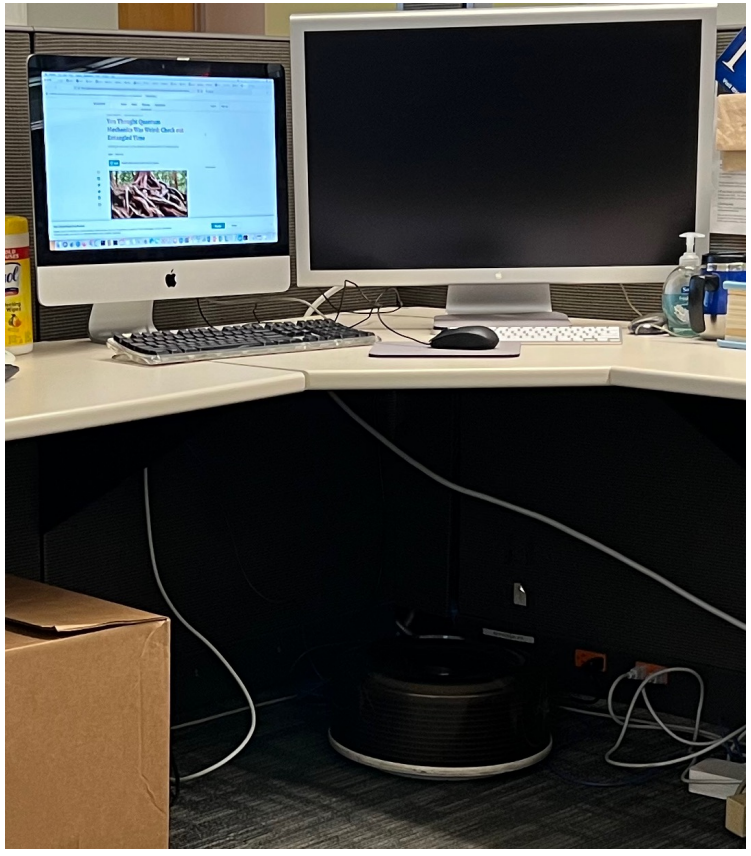
806,000 Bytes/sec

1.6 Billion bytes

200 MB/pack

(MY) PAGING DEVICES

IBM 3330 DASD (CKD)



806,000 Bytes/sec

1.6 Billion bytes

200 MB/pack

(MY) PAGING DEVICES

IBM 3350 DASD (CKD)

1198 KB/sec

317.5 MB/drive

(MY) PAGING DEVICES

IBM 3370 DASD (FBA)

1.86 MB/sec

571.3 – 729.8 MB/unit

(MY) PAGING DEVICES

IBM 3380 DASD (CKD)

3 MB/sec

2.52 – 5.04 GB

”Pain is inevitable. Suffering is optional.”

(MY) PAGING DEVICES

IBM 3390 DASD (CKD) – IBM 2105 ”Shark” (CKD) – DS8xxx DASD (CKD)

Emulated ECKD & RAID

(MY) PAGING DEVICES

Intel SSDPE2KX040T801 (FBA)

~~77~~ 3,000 MB/sec

4 Terabytes

2.5-inch form factor
1.6 ounces

PAGING (COMMENTS)

- PAGING is “cheap”, and z/VM pages very efficiently and effectively
- DASD is an investment, especially for its RAS characteristics
- NVMe is dirt cheap, and has spectacular performance, however caveat emptor

PAGING SUPPORT FOR NVME

- NVMe devices supported using EDEVICE emulation
- SET EDEVICE nnnn TYPE FBA ATTRIBUTES NVME PCIFUNCTION n
- SET EDEVICE nnnn TYPE FBA ATTRIBUTES NVME ALIAS PCIFUNCTION n
- 1 TB "segment" per EDEVICE
  - 4 TB NVMe requires 4 EDEVICES
- NVMe is a "multi-lane super-highway"; exploit ALIAS feature (multiple exposures)
  - SET PAGING ALIAS ON

TESTING NVME CONFIGURATION

1. SET MDC SYSTEM OFF
2. SET EDEVICE AD00 TYPE FBA ATTRIBUTES NVME PCIFUNCTION 1
3. SET EDEVICE AD01 TYPE FBA ATTRIBUTES NVME ALIAS PCIFUNCTION 1
4. ...
5. SET EDEVICE AD08 TYPE FBA ATTRIBUTES NVME ALIAS PCIFUNCTION 1
6. VARY ONLINE AD00-AD08
7. ATTACH AD00 *
8. CPFMTXA AD00 NVMExD ALLOCATE …
9. DETACH AD00
10. ATTACH AD00 SYSTEM
11. … (add volser NVMExD to EXTENT CONTROL, device type 9336-10 (FBA))
12. … (allocate MDISKs to CMS and LINUX users using DIRMAINT)
13. Repeat 2-12 for BD00-BD08, CD00-CD08, DD00-DD08

ALLOCATIONS - DIRMAINT

- DIRMAINT EXTENT CONTROL entries for 4 1T volumes:

```
NVME1A      NVME1A          0512            END         9336-10
NVME1B      NVME1B          0512            END         9336-10
NVME1C      NVME1C          0512            END         9336-10
NVME1D      NVME1D          0512            END         9336-10
```

ALLOCATIONS - DIRMAINT

DIRMAINT allocates (E)CKD minidisks in **cylinder** units: (3390: 180 pages/cylinder)

Let's allocate a 10MB minidisk:

A 10 MegaByte (10 * 1024 * 1024 bytes) ECKD MDISK is ~ 14 cylinders

(14 * 180 * 4096 = 10,321,920 bytes)

DIRMAINT allocates FBA minidisks in **FBA 512-byte block** units:

A 10 MegaByte (10 * 1024 * 1024 byte) FBA MDISK is precisely (10,485,760 / 512 = 20,480) FBA 512-byte blocks

Much more amenable to allocation by MB or GB

ALLOCATIONS - DIRMAINT

```
NVME3D   9336                                0          31          32 Gap
         RHEL8NVM 0200                      32    13107231    13107200
         RHEL8NVM 0191            13107232    13355231      248000
         RHEL9NVM 0191            13355232    23840991    10485760
         RHEL9NVM 0201            23840992    23940991      100000
                                  23940992   118212831    94271840 Gap
         RHEL9NVM 0200           118212832   223070431   104857600
         ALMA1    0201           223070432   449562847   226492416
         ALMA1    0202           449562848   676055263   226492416
         ALMA1    0203           676055264   902547679   226492416
         ALMA1    0204           902547680  1129040095   226492416
         ALMA1    0205          1129040096  1355532511   226492416
                                1355532512  1953508863   597976352 Gap
0-------------------------------------------------------------------
-------
NVME4D   9336                                0          31          32 Gap
         CMSGLD   0191                      32    10485791    10485760
                                   10485792    11199999      714208 Gap
         $PAGE$   DD00            11200000  1638400007  1627200008
                                1638400008  1953508863   315108856 Gap
0-------------------------------------------------------------------
```

ALLOCATIONS – DIRMAINT

We blatantly violated several fundamental "rules" of PAGE space allocation:

- Never mix PAGE with other allocation types on the same volume (we have lots of PERM - ~ 3.2T)
- Never mix device types for PAGE (we used 3390 and FBA EDEV)
- Always keep PAGE extents the same size across devices

Observation

- Traditionally, most expensive devices were devoted to paging (drums, DASD, Flash)
- NVMe is quite the opposite

ALLOCATIONS - CPFMTXA

CPFMTXA allocates space on (E)CKD volumes in cylinder units

CPFMTXA allocates space on FBA volumes in (4k) page units (NOT FBA block units)

To convert from FBA block units to page units, divide the FBA block units by 8

Example: If we wanted to start the PAGE space at FBA block 11,200,000 we need to convert to pages

So, 11,200,000 / 8 = 1,400,000 pages = starting extent of PAGE space (in page units)

- Where did "11,200,000" come from? (DIRM DIRMAP, DIRM FREE, etc.)

ALLOCATIONS - CPFMTXA

```
Actual CPFMTXA input:

CPFMTXA    DD00    NVME4D    ALLOCATE
PERM 4 1399999
PAGE 1400000 204800000                                  (~775GB of PAGE space)
END
```

ALLOCATIONS - CPFMTXA

Collaboration makes CPFMTXA more friendly:

```
CPFMTXA   DD00   NVME4D   ALLOCATE
PERM 5120M
PAGE 750G
END
```

ACTIVATING NEW PAGE SPACE

```
SET PAGING ALIAS ON
SET PAGING HPF ON
DETACH DD00
DEFINE CPOWNED SLOT 254 NVME4D
ATTACH DD00 SYSTEM
START DASD DD00 PAGE LINKS
```

PAGING TESTS

```
ind
AVGPROC-052% 0010
MDC READS-000010/SEC WRITES-000002/SEC HIT RATIO-039%
PAGING-60468/SEC
Q0-00002(00000)                                    DORMANT-00034
Q1-00000(00000)                E1-00000(00000)
Q2-00001(00001) EXPAN-002      E2-00000(00000)
Q3-00021(00008) EXPAN-002      E3-00000(00000)
PROC 0000-077% CP   VM         PROC 0002-074% CP   VL
PROC 0004-071% IFL  VH         PROC 0005-068% IFL  VH
PROC 0006-063% IFL  VM         PROC 0007-061% IFL  VM
PROC 0008-057% IFL  VL         PROC 0009-055% IFL  VL
PROC 000A-000% IFL  VL         PROC 000B-000% IFL  VL
LIMITED-00000
Ready; T=0.01/0.01 10:44:42
q alloc page
                  EXTENT       EXTENT   TOTAL   PAGES    HIGH     %
VOLID   RDEV       START          END   PAGES  IN USE    PAGE  USED
------ ---- ----------- ----------- ------- ------- ------- ----
NVME4D DD00     1400000   204800000    194M 21207K 25889K  10%
730PG1 521D           1       10016   1761K   1761K   1761K 100%
                                     ------- -------         ----
SUMMARY                                196M 22968K          11%
```

RESULTS

- Very high sustained paging rate
- Nothing "blew up"
- Expected service time per page was less than 1 millisecond
  - Actual measurements in the 0.1 millisecond range
- Exposed a configuration issue causing VLs
- Exposed several problems with our testing methodology
- Discovered amazing latent demand for disk space
  - We filled 4T in a week ("Nature abhors a vacuum")

NVME FAILURE & RECOVERY TESTING

```
SET PCIFUNCTION 00000001 RESET

  Device entered "Intervention Required" state
  Linux filesystem went into R/O mode immediately
  CMS was fine (Read Only) until RELEASE (then, "Boom")

  What happens to PAGE activity? System abend?
```

NVME FAILURE & RECOVERY TESTING

Recovery from "Intervention Required" state:

```
QUERY SYSTEM DD00
Detach MINIDISKS from all guests/userids
Drain PAGE/SPOOL
DETACH DD00 SYSTEM
VARY OFFLINE DD00
VARY OFFLINE SUBCHANNEL DD00
DELETE EDEVICE DD00
SET EDEVICE DD00 TYPE FBA ATTRIBUTES NVME PCIFUNCTION 1
VARY ONLINE DD00
ATTACH DD00 TO SYSTEM
START DASD DD00 PAGE LINKS
```

RECOVERY AFTER RESET

Data recovery varies:

- On 1 Linux test, needed to use "dd" command to write zeroes to device
  - "mkfs" was unable to create a new file system
- On other Linux tests, data was preserved 100%
- On CMS tests, data was preserved
- **Be prepared to restore all data**
- **Be prepared to write zeroes to device**
  - **Remember, no "dasdfmt" or "fdasd" for NVMe (FBA)**

LESSONS LEARNED

- Don't put NVMe paging definitions in SYSTEM CONFIG
- NVMe large paging is ideal for Test/Development environments
  - Large capacity, fast, cheap
- Use a Directory Manager

SUMMARY AND CONCLUSION

NVMe raises interesting options for inexpensive development environments

Kubernetes (or any) Lab-on-Demand?

- Many servers
- Quick provisioning
- Volatile ("throw away") environment

Lovely inexpensive PAGE device

Certainly "fast" and "cheap", but missing "RAS", crypto

THANK YOU