Technical Report

FlexPod - Epic for Large Hospitals

Bobby Oommen, NetApp
June 2023 | TR-4975

In partnership with:

Abstract

This technical report shows the end-to-end performance capability from FlexPod which is a joint solution between Cisco and NetApp demonstrating the need for Electronic Health Record (EHR) application with Intersystem database. The document focuses storage layout and best practices which needs to be followed while setting up the application on a FlexPod solution for large hospitals.

TABLE OF CONTENTS

Solution Overview

Introduction

NetApp offers a single data management solution for all enterprise applications, including healthcare, and guides hospitals toward digital transformation. This solution primarily discusses the best practices customers need to follow while setting up an EHR application on FlexPod.

Audience

This document is intended for Cisco & NetApp, solutions engineers (SEs), professional services personnel, and anyone with a solid understanding of NetApp ONTAP and Cisco Systems and a fair understanding of EHR systems to configure a FlexPod solution for large hospitals.

Solution Benefits

This FlexPod Data Center solution with end-to-end NVMe configuration covers all the nuances needed to run an EMR application for large hospitals.

- **Data Protection & Business Continuity:** Using immutable NetApp storage Snapshots, customers can protect the applications allowing instantaneous backup windows without impacting the business, alleviating the impact of ransomware and other security concerns. Additionally, the Snapshots can be leveraged for spinning up application copies or meeting any cloud burst requirements.
- **Efficiency:** The modularized architecture provided by Cisco & NetApp helps the environment in terms of carbon footprint, less IT waste during hardware upgrades, using lesser infrastructure footprint through the latest CPU architectures, and better capacity utilization from storage solutions.
- **Simple & Smart:** Through Cisco Intersight and NetApp BlueXP, customers can manage their infrastructure stack through global policies and templates by integrating hybrid multi-cloud scenarios to complement their mission-critical on-prem solutions.

Solution Topology

This section describes the logical topology of the FlexPod Data Center architecture, which was used for the solution validation. FlexPod is an engineered solution with hardware and software components that can extend to any cloud provider, including NetApp ONTAP storage, Cisco Nexus networking, Cisco MDS storage networking, and Cisco Unified Computing System (Cisco UCS).

In the current world of digital transformation, like any vertical, clinical organizations strive to improve patient experiences and outcomes. With FlexPod, you get a secure, scalable platform that drives efficiency and empowers your staff to make more informed decisions faster, so they can provide better patient care by giving:

  - Optimize operations and get faster insights for better patient outcomes.
  - Streamline your imaging apps with scalable, reliable infrastructure.
  - Deploy quickly and efficiently with a proven approach for healthcare-specific apps like EHR.

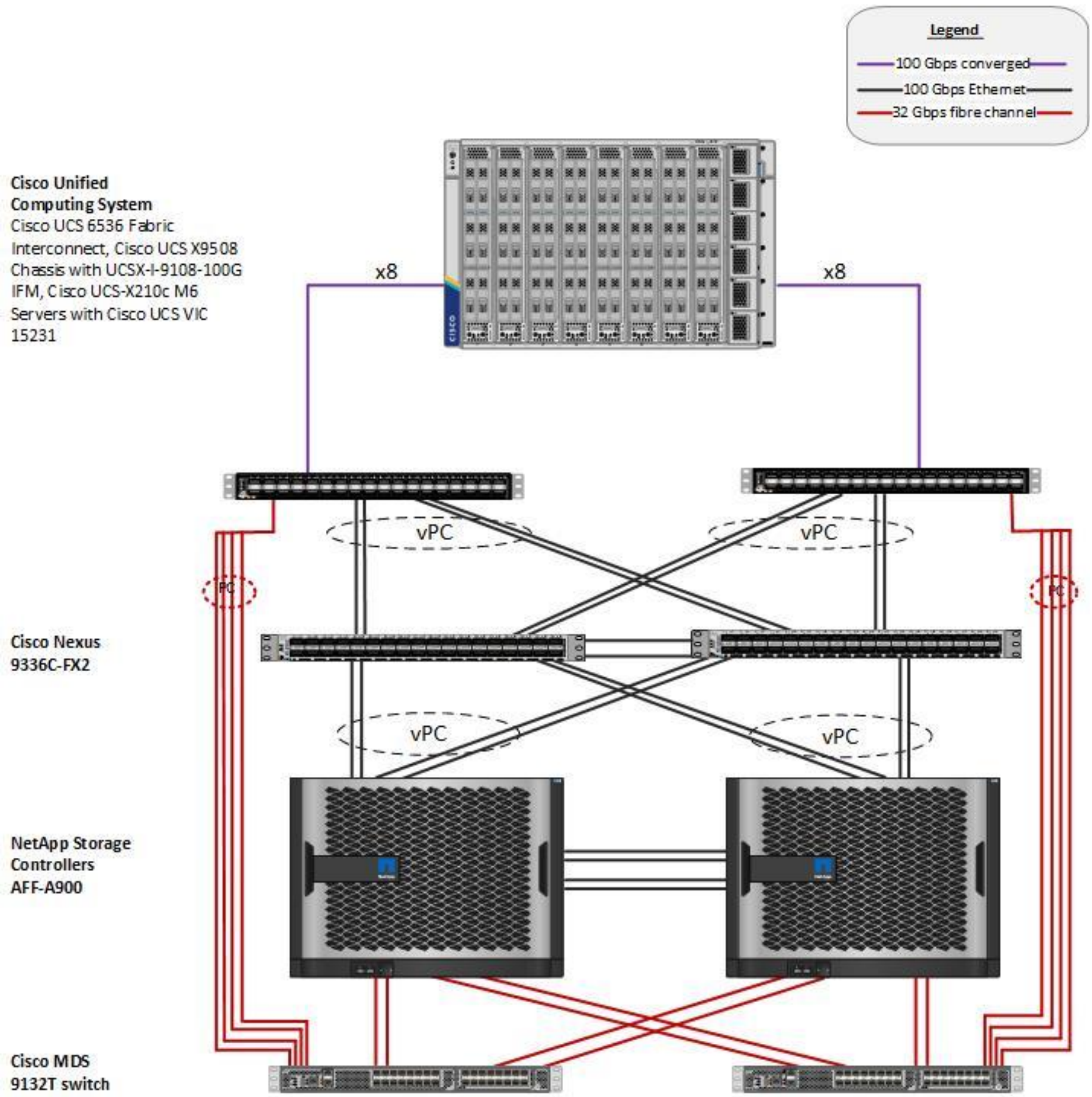


Figure 1) Topology Diagram for FlexPod

Topology details

- Four 2-port 40/100G cards (P/N X91148A) for data, 6 * 2-port 40/100G RoCE cards (X91153A) for cluster interconnect and disk shelf access, and 2 * 4-port 32G FC cards (P/N X91135A) on NetApp AFF A900, Cisco VIC 15231 adapters with two 100G ports on each X210c servers.
- 16 * 100G links connected between Cisco Fabric Interconnects (UCS-FI-6536) and Intelligent Fabric Module (IFM) to exploit the full bandwidth of both IFMs when the chassis is populated with eight compute nodes.
- Eight 100G links connected between NetApp AFF A900 and Cisco Nexus 9336-FX2 to get full bandwidth on the storage.

- Eight 100G links between Fabric Interconnects and Nexus Switches to match the ethernet bandwidth of storage controllers.
- Eight 32G FC links between Fabric Interconnects and Cisco MDS switches.
- Eight 32G FC links between Cisco MDS switches and NetApp AFF A900.

Solution Components

The solution comes with sustainable modular architecture on both compute and storage, helping with carbon footprint and environmental waste while reducing Data Center footprints.

NetApp AFF A900

Built upon a high-resilient modular architecture, the AFF A900 delivers reliability, availability, and serviceability for critical business applications. The AFF A900, which is a high-end storage offering from NetApp, has an enhanced performance and high-performance I/O density in an 8U HA form factor which gives multi-protocol support including NVMe/FC and NVMe/TCP through 100GbE, 25GbE, and 32Gb FC. Combined with the ONTAP Enterprise Edition software, the AFF A900 system delivers a premium data management solution with built-in data protection and security while providing maximum flexibility and simplicity. Furthermore, as Data Centers consume significant power and emit greenhouse emissions, NetApp provides lifetime carbon footprint estimates of the storage to help customers to create the environmental impacts.

Cisco UCS® X-Series

The Cisco UCS® X-Series Modular system is designed for traditional scale-out and enterprise workloads providing operational efficiency, agility, and sustainability. X-series is built for cloud scale and powered by the Cisco Intersight™, allowing customers to manage their Data Center resources distributed globally like a cloud platform. X-Series has a future-ready X9508 chassis with a midplane-free design with vertically aligned compute nodes [X210c] connecting to horizontally aligned IO modules at the rear. The X-Fabric modules can host GPU cards that could be shared with any of the X210c compute nodes in the chassis. In addition, the switching intelligence is built into the Fabric modules, not the chassis itself, enabling customers to independently replace either the compute node, GPU nodes, or the intelligent Fabric IO modules.

Compared to a rack server solution, this design reduces the e-waste significantly, where customers could keep all the other components, i.e., fans, power supplies, and chassis. Furthermore, the design came with zone-based cooling on the chassis and distributed sensors which allow users to control the speed of the fans and the power supplies through Intersight policies giving better power & cooling efficiency. Apart from the policies on the chassis, the compute nodes also have their own BIOS policies, which allow users to set different power policies and get the best use of the infrastructure, ensuring end-to-end sustainability.

Hardware and Software Revisions

This solution can be extended to any FlexPod environment running supported software, firmware, and hardware versions as defined in the NetApp Interoperability Matrix Tool, UCS Hardware and Software Compatibility, and VMware Compatibility Guide. The following table shows the on-premises FlexPod hardware and software revisions.

Component	Product	Version
Compute	Cisco UCS X210c M6	5.1(0.230054)
	Cisco UCS Fabric Interconnects 6536	4.2(2d)
	Cisco VIC 15231	5.3(1.230031)
	CPU	Platinum 8352Y CPU @ 2.20GHz
Network	Cisco Nexus 9336C-FX2 NX-OS	9.3(9)
	Cisco MDS 9132T	9.2.2
Storage	NetApp AFF A900	ONTAP 9.12.1P2
Software	fnic ethernet driver	2.0.0.87b-238.0
	Cisco Intersight Assist Virtual Appliance	1.0.9-342
	RHEL	8.7

Table 1 On-premises FlexPod Hardware and Software Revisions

Installation and Configuration

On-premises FlexPod Deployment

To understand FlexPod with UCS X-Series, VMware, and NetApp ONTAP design details, refer to FlexPod Datacenter with Cisco UCS X-Series Design Guide. This document provides design guidance around incorporating the Cisco Intersight—managed UCS X-Series platform within the FlexPod infrastructure. For deploying the on-premises FlexPod, refer to the following deployment guide using Ansible automation.

Compute and Storage layout

Both Cisco and NetApp storage features can provide multi-tenancy, which allows the creation of virtual components resulting in workload isolation & security.

Network Port Virtualization on X210c

The VIC adapters that do the IO traffic can be sliced into multiple virtual adapters (vHBA) to create IO parallelization and traffic isolation. For this solution:

- Four vHBAs for NVMe were created on each Fabric, two for data and two for journal
- Four vHBAs for FCP were created on each Fabric for application, two for data, and two for journal
- One vHBA was used for FC SAN Boot from each port

VIC Adapter (HBA)	Type	Protocol	Virtual Port Name (vHBA)
Port1, Fabric-A	Data	NVMe	host1-nvme-A1, host1-nvme-A2
	Journal		host1-nvme-A3, host1-nvme-A4
	Data	FCP	host1-fcp-A1, host1-fcp-A2,
	Journal		host1-fcp-A3, host1-fcp-A4
	SANBoot		host1-fcboot-a
Port1, Fabric-A	Data	NVMe	host1-nvme-B1, host1-nvme-B2
	Journal		host1-nvme-B3, host1-nvme-B4
	Data	FCP	host1-fcp-B1, host1-fcp-B2,
	Journal		host1-fcp-B3, host1-fcp-B4
	SANBoot		host1-fcboot-b

NetApp A900 configuration

Pre-requisites

- All the required licenses on the NetApp storage – FCP, NVMe
- Adapters supporting FC, NVMe

Storage Layout

This section describes the ONTAP configuration needed to do the storage layout for this solution. The following steps were used:

- One SVM for the FC boot for the host server
- One volume was created on one of the controllers with one LUN for SAN boot.
- Two SVMs, one for the data traffic and the other for journal files, were done to complete the isolation of data and journal traffic using separate LIFs [Logical Interfaces].
- 32 volumes, 16 on each A900 controller, were created to distribute the data traffic and the optimal CPU parallelization on the storage.
- 32 namespaces were created, one on each volume for data traffic.
- One subsystem tied to host1 was used to map all the data namespaces.

1. Configure SVM with NVMe/FC service running.

```
vserver create –vserver vs1 –rootvolume vs1root –aggregate aggr1_01 –rootvolume-security-style unix
vserver create –vserver vs2 –rootvolume vs2root –aggregate aggr1_02 –rootvolume-security-style unix

vserver add-protocols -vserver vs1 -protocols nvme,fcp
vserver add-protocols -vserver vs2 -protocols nvme,fcp
vserver fcp create -vserver vs1 -status-admin up
vserver fcp create -vserver vs2 -status-admin up
vserver nvme create -vserver vs1 -status-admin up
vserver nvme create -vserver vs2 -status-admin up
```

2. Create a FlexVol volume for Boot, Data, and Journal files. The namespaces were created, each with 1TB, and NetApp recommends using larger sizes to accommodate future data growth.

```
volume create -vserver vs1 -aggregate aggr1_01 -volume boot1 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_01 -volume data1 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_02 -volume data2 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_01 -volume data3 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_02 -volume data4 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_01 -volume data5 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_02 -volume data6 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_01 -volume data7 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_02 -volume data8 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_01 -volume data9 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_02 -volume data10 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_01 -volume data11 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_02 -volume data12 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_01 -volume data13 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_02 -volume data14 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_01 -volume data15 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_02 -volume data16 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_01 -volume data17 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_02 -volume data18 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_01 -volume data19 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_02 -volume data20 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_01 -volume data21 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_02 -volume data22 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_01 -volume data23 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_02 -volume data24 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_01 -volume data25 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_02 -volume data26 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_01 -volume data27 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_02 -volume data28 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_01 -volume data29 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_02 -volume data30 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_01 -volume data31 -size 1t -state online
volume create -vserver vs1 -aggregate aggr1_02 -volume data32 -size 1t -state online
volume create -vserver vs2 -aggregate aggr1_01 -volume jrn1 -size 500g -state online
volume create -vserver vs2 -aggregate aggr1_02 -volume jrn2 -size 500g -state online
volume create -vserver vs2 -aggregate aggr1_01 -volume jrn3 -size 500g -state online
volume create -vserver vs2 -aggregate aggr1_02 -volume jrn4 -size 500g -state online
volume create -vserver vs2 -aggregate aggr1_01 -volume jrn5 -size 500g -state online
volume create -vserver vs2 -aggregate aggr1_02 -volume jrn6 -size 500g -state online
volume create -vserver vs2 -aggregate aggr1_01 -volume jrn7 -size 500g -state online
volume create -vserver vs2 -aggregate aggr1_02 -volume jrn8 -size 500g -state online
```

3. Create FC LUN for SAN Boot.

```
lun create -vserver vs1 -path /vol/boot1/boot1 -size 200g -ostype linux
```


4. Create igroup for SAN boot and map the LUN. The interface names - host1-fcboot-a, host1-fcboot-b should be replaced with the associated wwpn.

```
igroup create -vserver vs1 -igroup boot1 -protocol fcp -ostype linux -initiator host1-fcboot-a,
host1-fcboot-b
lun map -vserver vs1 -path /vol/boot1/boot1 -igroup boot1
```

5. Depending on the protocol, either FCP or NVMe could be used, and both steps are outlined below.

    a. Add NVMe namespaces.

```
vserver nvme namespace create -vserver vs1 -path /vol/data1/data1 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data2/data2 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data3/data3 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data4/data4 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data5/data5 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data6/data6 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data7/data7 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data8/data8 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data9/data9  -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data10/data10 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data11/data11 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data12/data12 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data13/data13 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data14/data14 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data15/data15 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data16/data16 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data17/data17 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data18/data18 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data19/data19 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data20/data20 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data21/data21 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data22/data22 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data23/data23 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data24/data24 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data25/data25 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data26/data26 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data27/data27 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data28/data28 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data29/data29 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data30/data30 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data31/data31 -size 800g -ostype linux
vserver nvme namespace create -vserver vs1 -path /vol/data32/data32 -size 800g -ostype linux
vserver nvme namespace create -vserver vs2 -path /vol/jrn1/jrn1 -size 250g -ostype linux
vserver nvme namespace create -vserver vs2 -path /vol/jrn2/jrn2 -size 250g -ostype linux
vserver nvme namespace create -vserver vs2 -path /vol/jrn3/jrn3 -size 250g -ostype linux
vserver nvme namespace create -vserver vs2 -path /vol/jrn4/jrn4 -size 250g -ostype linux
vserver nvme namespace create -vserver vs2 -path /vol/jrn5/jrn5 -size 250g -ostype linux
vserver nvme namespace create -vserver vs2 -path /vol/jrn6/jrn6 -size 250g -ostype linux
vserver nvme namespace create -vserver vs2 -path /vol/jrn7/jrn7 -size 250g -ostype linux
vserver nvme namespace create -vserver vs2 -path /vol/jrn8/jrn8 -size 250g -ostype linux
```

    b. Create FCP LUNs.

```
lun create -vserver vs1 -path /vol/data1/data1 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data2/data2 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data3/data3 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data4/data4 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data5/data5 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data6/data6 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data7/data7 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data8/data8 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data9/data9 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data10/data10 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data11/data11 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data12/data12 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data13/data13 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data14/data14 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data15/data15 -size 800g -ostype linux
```



```
lun create -vserver vs1 -path /vol/data16/data16 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data17/data17 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data18/data18 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data19/data19 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data20/data20 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data21/data21 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data22/data22 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data23/data23 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data24/data24 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data25/data25 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data26/data26 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data27/data27 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data28/data28 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data29/data29 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data30/data30 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data31/data31 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/data32/data32 -size 800g -ostype linux
lun create -vserver vs1 -path /vol/jrn1/jrn1 -size 200g -ostype linux
lun create -vserver vs1 -path /vol/jrn2/jrn2 -size 200g -ostype linux
lun create -vserver vs1 -path /vol/jrn3/jrn3 -size 200g -ostype linux
lun create -vserver vs1 -path /vol/jrn4/jrn4 -size 200g -ostype linux
lun create -vserver vs1 -path /vol/jrn5/jrn5 -size 200g -ostype linux
lun create -vserver vs1 -path /vol/jrn6/jrn6 -size 200g -ostype linux
lun create -vserver vs1 -path /vol/jrn7/jrn7 -size 200g -ostype linux
lun create -vserver vs1 -path /vol/jrn8/jrn8 -size 200g -ostype linux
```

6. Depending on the protocol, create an igroup or NVMe subsystem

    a. Create subsystem for NVMe

```
nvme subsystem create -vserver vs1 -subsystem host1 -ostype linux -default-io-queue-count 15 -
default-io-queue-depth 128
nvme subsystem create -vserver vs2 -subsystem host1 -ostype linux -default-io-queue-count 15 -
default-io-queue-depth 128
```

    b. Map NVMe nqn to the subsystem. The hostnqn was taken from the RHEL server under the path /etc/nvme/hostnqn file.

```
nvme subsystem host add -vserver vs1 -subsystem host1 -host-nqn nqn.2014-08.sa.perf:nvme:host1 -
io-queue-count 15 -io-queue-depth 128
nvme subsystem host add -vserver vs2 -subsystem host1 -host-nqn nqn.2014-08.sa.perf:nvme:host1 -
io-queue-count 15 -io-queue-depth 128
```

    c. Create igroup for FCP data and journal LUNs. The interface names list in the command syntax should be replaced with the associated wwpn.

```
igroup create -vserver vs1 -igroup data -protocol fcp -ostype linux -initiator host1-FCP-A1,
host1-FCP-A2, host1-FCP-B1, host1-FCP-B2
igroup create -vserver vs1 -igroup jrn -protocol fcp -ostype linux -initiator host1-FCP-A3,
host1-FCP-A4, host1-FCP-B3, host1-FCP-B4
```

7. Depending on the protocol, Map the storage devices to the host system.

    a. Map NVMe namespaces to the `"host1"` subsystem

```
vserver nvme subsystem map add -vserver vs1 -path /vol/data1/data1 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data2/data2 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data3/data3 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data4/data4 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data5/data5 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data6/data6 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data7/data7 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data8/data8 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data9/data9 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data10/data10 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data11/data11 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data12/data12 -subsystem host1
```



```
vserver nvme subsystem map add -vserver vs1 -path /vol/data13/data13 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data14/data14 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data15/data15 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data16/data16 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data17/data17 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data18/data18 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data19/data19 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data20/data20 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data21/data21 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data22/data22 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data23/data23 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data24/data24 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data25/data25 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data26/data26 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data27/data27 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data28/data28 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data29/data29 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data30/data30 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data31/data31 -subsystem host1
vserver nvme subsystem map add -vserver vs1 -path /vol/data32/data32 -subsystem host1
vserver nvme subsystem map add -vserver vs2 -path /vol/jrn1/jrn1 -subsystem host1
vserver nvme subsystem map add -vserver vs2 -path /vol/jrn2/jrn2 -subsystem host1
vserver nvme subsystem map add -vserver vs2 -path /vol/jrn3/jrn3 -subsystem host1
vserver nvme subsystem map add -vserver vs2 -path /vol/jrn4/jrn4 -subsystem host1
vserver nvme subsystem map add -vserver vs2 -path /vol/jrn5/jrn5 -subsystem host1
vserver nvme subsystem map add -vserver vs2 -path /vol/jrn6/jrn6 -subsystem host1
vserver nvme subsystem map add -vserver vs2 -path /vol/jrn7/jrn7 -subsystem host1
vserver nvme subsystem map add -vserver vs2 -path /vol/jrn8/jrn8 -subsystem host1
```

b. Map FCP data and journal LUNs to `"data and jrn"` igroups

```
lun map -vserver vs1 -path /vol/data1/data1 -igroup data
lun map -vserver vs1 -path /vol/data2/data2 -igroup data
lun map -vserver vs1 -path /vol/data3/data3 -igroup data
lun map -vserver vs1 -path /vol/data4/data4 -igroup data
lun map -vserver vs1 -path /vol/data5/data5 -igroup data
lun map -vserver vs1 -path /vol/data6/data6 -igroup data
lun map -vserver vs1 -path /vol/data7/data7 -igroup data
lun map -vserver vs1 -path /vol/data8/data8 -igroup data
lun map -vserver vs1 -path /vol/data9/data9 -igroup data
lun map -vserver vs1 -path /vol/data10/data10 -igroup data
lun map -vserver vs1 -path /vol/data11/data11 -igroup data
lun map -vserver vs1 -path /vol/data12/data12 -igroup data
lun map -vserver vs1 -path /vol/data13/data13 -igroup data
lun map -vserver vs1 -path /vol/data14/data14 -igroup data
lun map -vserver vs1 -path /vol/data15/data15 -igroup data
lun map -vserver vs1 -path /vol/data16/data16 -igroup data
lun map -vserver vs1 -path /vol/data17/data17 -igroup data
lun map -vserver vs1 -path /vol/data18/data18 -igroup data
lun map -vserver vs1 -path /vol/data19/data19 -igroup data
lun map -vserver vs1 -path /vol/data20/data20 -igroup data
lun map -vserver vs1 -path /vol/data21/data21 -igroup data
lun map -vserver vs1 -path /vol/data22/data22 -igroup data
lun map -vserver vs1 -path /vol/data23/data23 -igroup data
lun map -vserver vs1 -path /vol/data24/data24 -igroup data
lun map -vserver vs1 -path /vol/data25/data25 -igroup data
lun map -vserver vs1 -path /vol/data26/data26 -igroup data
lun map -vserver vs1 -path /vol/data27/data27 -igroup data
lun map -vserver vs1 -path /vol/data28/data28 -igroup data
lun map -vserver vs1 -path /vol/data29/data29 -igroup data
lun map -vserver vs1 -path /vol/data30/data30 -igroup data
lun map -vserver vs1 -path /vol/data31/data31 -igroup data
lun map -vserver vs1 -path /vol/data32/data32 -igroup data
lun map -vserver vs1 -path /vol/jrn1/jrn1 -igroup jrn
lun map -vserver vs1 -path /vol/jrn2/jrn2 -igroup jrn
lun map -vserver vs1 -path /vol/jrn3/jrn3 -igroup jrn
lun map -vserver vs1 -path /vol/jrn4/jrn4 -igroup jrn
lun map -vserver vs1 -path /vol/jrn5/jrn5 -igroup jrn
lun map -vserver vs1 -path /vol/jrn6/jrn6 -igroup jrn
```



```
lun map -vserver vs1 -path /vol/jrn7/jrn7 -igroup jrn
lun map -vserver vs1 -path /vol/jrn8/jrn8 -igroup jrn
```

8. Add interfaces for FC boot.

```
net int create -vserver vs1 -lif fc-boot-9a-n1 -data-protocol fcp -home-node netapp-01 -home-port
9a -status-admin up
net int create -vserver vs1 -lif fc-boot-9a-n2 -data-protocol fcp -home-node netapp-02 -home-port
9a -status-admin up
net int create -vserver vs1 -lif fc-boot-9b-n1 -data-protocol fcp -home-node netapp-01 -home-port
9b -status-admin up
net int create -vserver vs1 -lif fc-boot-9b-n2 -data-protocol fcp -home-node netapp-02 -home-port
9b -status-admin up
```

9. Add storage interfaces if you are using NVMe setup

```
net int create -vserver vs1 -lif fc-nvme-9c-n1 -data-protocol fc-nvme -home-node netapp-01 -home-
port 9c -status-admin up
net int create -vserver vs1 -lif fc-nvme-9d-n1 -data-protocol fc-nvme -home-node netapp-01 -home-
port 9d -status-admin up
net int create -vserver vs1 -lif fc-nvme-9c-n2 -data-protocol fc-nvme -home-node netapp-02 -home-
port 9c -status-admin up
net int create -vserver vs1 -lif fc-nvme-9d-n2 -data-protocol fc-nvme -home-node netapp-02 -home-
port 9d -status-admin up

net int create -vserver vs2 -lif fc-nvme-9a-n1 -data-protocol fc-nvme -home-node netapp-01 -home-
port 9a -status-admin up
net int create -vserver vs2 -lif fc-nvme-9b-n1 -data-protocol fc-nvme -home-node netapp-01 -home-
port 9b -status-admin up
net int create -vserver vs2 -lif fc-nvme-9a-n2 -data-protocol fc-nvme -home-node netapp-02 -home-
port 9a -status-admin up
net int create -vserver vs2 -lif fc-nvme-9b-n2 -data-protocol fc-nvme -home-node netapp-02 -home-
port 9b -status-admin up
```

10. Add storage interfaces if you are using an FCP setup for data and journal files

```
net int create -vserver vs1 -lif fcp-9c-n1 -data-protocol fcp -home-node netapp-01 -home-port 9c-
status-admin up
net int create -vserver vs1 -lif fcp-9c-n2 -data-protocol fcp -home-node netapp-02 -home-port 9c-
status-admin up
net int create -vserver vs1 -lif fcp-9d-n1 -data-protocol fcp -home-node netapp-01 -home-port 9d-
status-admin up
net int create -vserver vs1 -lif fcp-9d-n2 -data-protocol fcp -home-node netapp-02 -home-port 9d-
status-admin up

net int create -vserver vs2 -lif fcp-9a-n1 -data-protocol fcp -home-node netapp-01 -home-port 9a-
status-admin up
net int create -vserver vs2 -lif fcp-9a-n2 -data-protocol fcp -home-node netapp-02 -home-port 9a-
status-admin up
net int create -vserver vs2 -lif fcp-9b-n1 -data-protocol fcp -home-node netapp-01 -home-port 9b-
status-admin up
net int create -vserver vs2 -lif fcp-9b-n2 -data-protocol fcp -home-node netapp-02 -home-port 9b-
status-admin up
```

Zoning on the Fabric switch

The zone was done to isolate the data and journal traffic to avoid any contention.

- Two vHBAs were used from each VIC adapter port for data and journal files.
- And from the storage side, two logical interfaces [lifs] were used for data and journal files on each Fabric.
- A separate zone was created for the boot device as well.

Zoning for NVMe setup

Fabric Name	Zone Name	Host vHBA (initiator)	Storage LIFs (Target)
Fabric-A	zone1 (data)	host1-nvme-A1 host1-nvme-A2	fc-nvme-9c-n1 fc-nvme-9c-n2
	zone2 (journal)	host1-nvme-A3 host1-nvme-A4	fc-nvme-9a-n1 fc-nvme-9a-n2
	zone3 (boot)	host1-fcboot-a	fc-boot-9a-n1 fc-boot-9a-n2
Fabric-B	zone1 (data)	host1-nvme-B1 host1-nvme-B2	fc-nvme-9d-n1 fc-nvme-9d-n2
	zone2 (journal)	host1-nvme-B3 host1-nvme-B4	fc-nvme-9b-n1 fc-nvme-9b-n2
	zone3 (boot)	host1-fcboot-b	fc-boot-9b-n1 fc-boot-9b-n2

Zoning for FCP setup

Fabric Name	Zone Name	Host vHBA (initiator)	Storage LIFs (Target)
Fabric-A	zone1 (data)	host1-fcp-A1 host1-fcp-A2	fcp-9c-n1 fcp-9c-n2
	zone2 (journal)	host1-fcp-A3 host1-fcp-A4	fcp-9a-n1 fcp-9a-n2
	zone3 (boot)	host1-fcboot-a	fc-boot-9a-n1 fc-boot-9a-n2
Fabric-B	zone1 (data)	host1-nvme-B1 host1-nvme-B2	fcp-9d-n1 fcp-9d-n2
	zone2 (journal)	host1-nvme-B3 host1-nvme-B4	fcp-9b-n1 fcp-9b-n2
	zone3 (boot)	host1-fcboot-b	fc-boot-9b-n1 fc-boot-9b-n2

SAN Configuration on the host

This section describes the host-side configuration required to best integrate with NetApp storage, conforming to [NetApp Interoperability Matrix Tool (IMT)]() to validate all software and firmware versions.

Discover ONTAP Storage on the host system running RHEL

Once the namespaces are mapped to the subsystem with the correct host-nqn numbers, the storage devices should be visible to the host. Based on the storage layout:

- A total of 40 namespaces [32 for data and 8 for journal] should appear on the NVMe host or 40 LUNs [32 for data and 8 for journal] for the FCP setup.
- The following configuration should be used, which gives the optimal performance in terms of IOPS and latency when you use NVMe or FCP
  - LVM was used by setting the stripe size and stripe width.
  - The data volume group was created with a stripe width of 32 and stripe size of 2M and using the same setting for the logical volume and utilized the entire space in a volume group.
  - The journal volume group was created with a stripe width of 8 and stripe size of 256k, using the same setting for the logical volume and utilizing the entire volume group space.
  - XFS filesystems were used for both data and journal volumes.

Configure Multipathing

Device Mapper Multipathing (DM-Multipath) is used with the ANA driver for this solution when NVMe is used. For the boot LUNs in a SAN configuration, NetApp recommends to set the DM-Multipath. Before setting up DM-Multipath on your system, ensure that your system has been updated and includes the `device-mapper-multipath` package.

```
rpm -qa|grep multipath
device-mapper-multipath-libs-0.8.4-31.el8.x86_64
device-mapper-multipath-0.8.4-31.el8.x86_64
```

1. The configuration file is `/etc/multipath.conf` file. Update the configuration file as shown below.

```
defaults {
        find_multipaths yes
        user_friendly_names yes
        enable_foreign NONE
}
blacklist {
}
multipaths {
        multipath {
                wwid    3600a09803831377a623f5464554f7552
                alias   boot1
```

2. Enable and start the multipath services.

```
systemctl enable multipathd.service
systemctl start  multipathd.service
```

Note: For detailed information about the above steps, refer [here]().

Linux LVM Setup

The parameters used for the LVM setup are based on the testing done for this solution and are optimal for GenIO workloads on a NetApp storage system. A physical extent size of 2M was used for data, and 256k was used for journal files. The stripe size was set to 32 for data, and for the journal, it was set to 8.

The example below shows the device naming for an NVMe setup, and FCP same command syntax is used with the FCP device names.

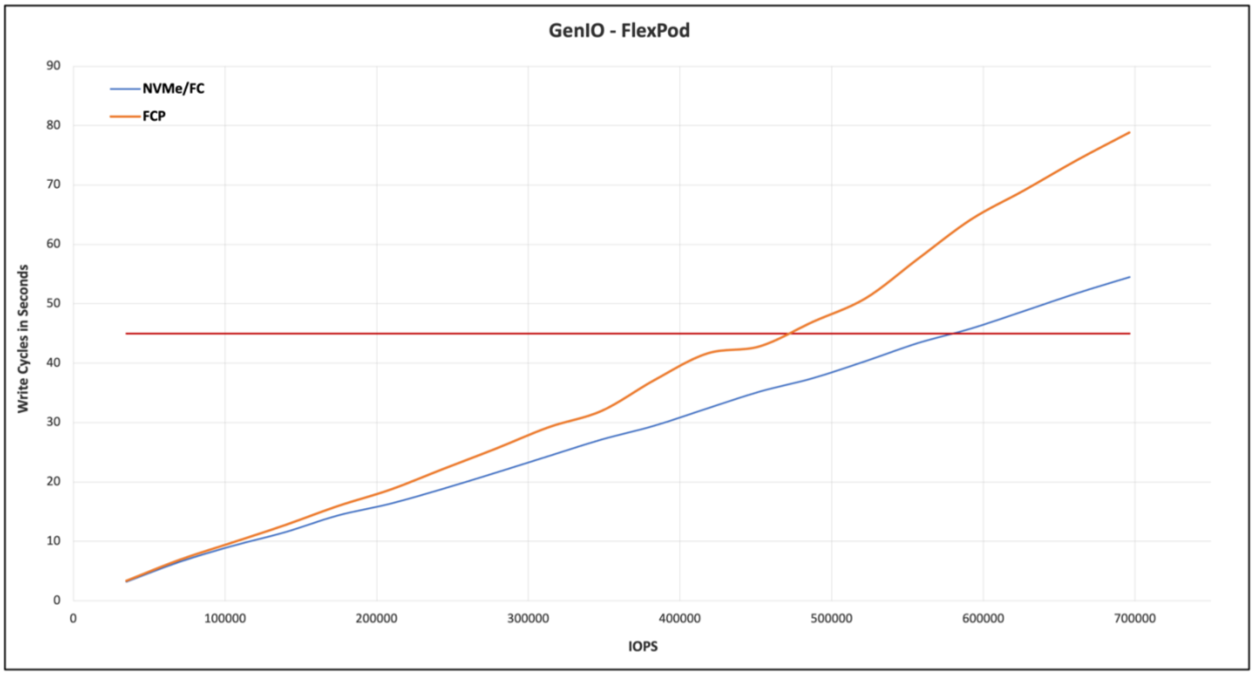
```
vgcreate -s 2M datavg /dev/nvme12n1 /dev/nvme12n10 /dev/nvme12n11 /dev/nvme12n12 /dev/nvme12n13
/dev/nvme12n14 /dev/nvme12n15 /dev/nvme12n16 /dev/nvme12n17 /dev/nvme12n18 /dev/nvme12n19
/dev/nvme12n2 /dev/nvme12n20 /dev/nvme12n21 /dev/nvme12n22 /dev/nvme12n23 /dev/nvme12n24
/dev/nvme12n25 /dev/nvme12n26 /dev/nvme12n27 /dev/nvme12n28 /dev/nvme12n29 /dev/nvme12n3
/dev/nvme12n30 /dev/nvme12n31 /dev/nvme12n32 /dev/nvme12n4 /dev/nvme12n5 /dev/nvme12n6
/dev/nvme12n7 /dev/nvme12n8 /dev/nvme12n9

lvcreate -n datalv -l 100%FREE -i 32 -I 2M datavg

vgcreate -s 256k jrnvg /dev/nvme1n1 /dev/nvme1n2 /dev/nvme1n3 /dev/nvme1n4 /dev/nvme1n5
/dev/nvme1n6 /dev/nvme1n7 /dev/nvme1n8

lvcreate -n jrnlv -l 100%FREE -i 8 -I 256k jrnvg

mkfs.xfs -K /dev/datavg/datalv
mkfs.xfs -K /dev/jrnvg/jrnlv

mkdir /prod1 ; mkdir /jrn1
mount -t xfs /dev/datavg/datalv /prod1
mount -t xfs /dev/jrnvg/jrnlv /jrn1
```

Data Generation

Dgen.pl is a Perl script data Generator for EHR's IO Simulator (aka **GenerateIO**). The tool (GenIO) is used to validate that storage is production ready by pushing storage to its limits and determining the headroom on storage controllers by ramping up until the write cycles go beyond 45 seconds. The workload characteristic of GenIO is a mix of reads and writes, generating an 8K workload for the data files and a 252k block size for the journal files. The QoS setting on the storage was kept wide open, and only one host server running RHEL 9.1 was used for this validation. Data inside the LUNs were generated with EHR's *Dgen.pl* script. The script is designed to create data like what would be found inside an EHR database.

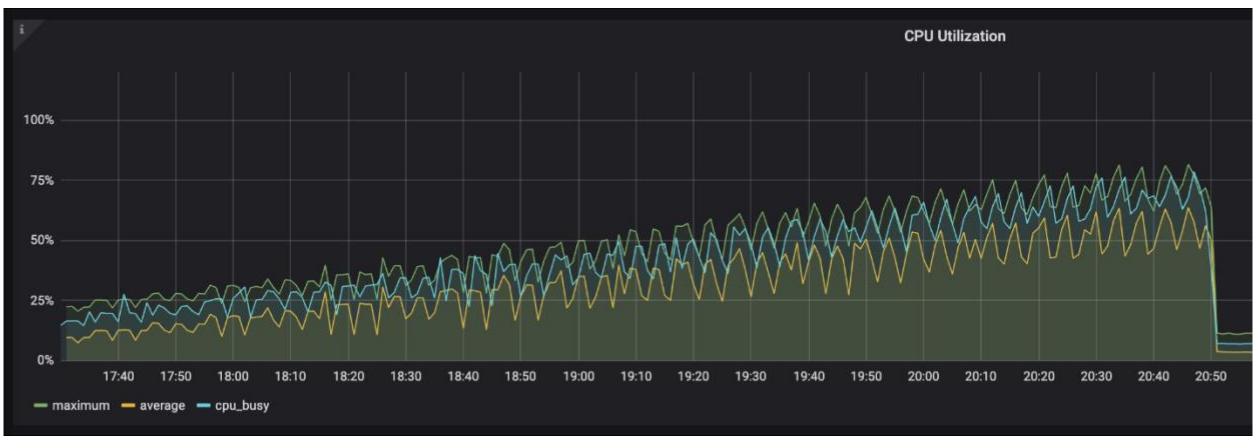
Performance Validation

The GenIO validation was done by setting the maxiops to 700,000 IOPS for NVMe and FCP setups from a single X210c host. All the best practices from a FlexPod configuration were based on the steps outlined in the CVD [Cisco validated design] in the "Installation and Configuration" section. The performance chart shows that AFF A900 NetApp storage in a FlexPod configuration can meet the requirements set by EHR applications for large hospitals, generating ~480K and ~590K GenIO IOPS for FCP and NVMe, respectively.



Storage Utilization

The average CPU utilization was around 50% during the peak load, and the NetApp storage was left with enough CPU headroom to accommodate more applications along with EHR to give a complete Data Center solution for healthcare systems.



Data Protection

Customers have widely used NetApp Snapshot in the context of data protection, addressing the backup window challenges Data Centers face. Additionally, as Snapshots are incremental, up to 1023 could be taken at the volume level, and any number of FlexClones can be created based on a Snapshot to meet any system refresh needs.

Conclusion

This engineered solution from Cisco and NetApp gives customers running EHR an architecture where you don't have to do guesswork. The GenIO IOPS numbers given by the solution helps users of EHR system to meet the requirements of large hospitals. This solution caters to the need of healthcare organizations. It enables them to take a step toward digital transformation and helps them manage their applications and workloads.

Acknowledgment

The author would like to thank continuous help and support by:

Roney Daniel, Technical Marketing Engineer, NetApp

Where to find additional information

To learn more about the information that is described in this document, review the following documents and websites:

- FlexPod Home Page: https://www.flexpod.com
- Cisco Validated Design and deployment guides for FlexPod: https://www.cisco.com/c/en/us/solutions/design-zone/data-center-design-guides/flexpod-design-guides.html
- SnapMirror Replication: https://docs.netapp.com/us-en/cloud-manager-replication/concept-replication.html
- FlexPod CVD for EHR: https://www.cisco.com/c/en/us/td/docs/unified_computing/ucs/UCS_CVDs/flexpod_xseries_vmw_epic.html
- NetApp Interoperability Matrix Tool: http://support.netapp.com/matrix/
- Cisco UCS Hardware and Software Interoperability Tool: http://www.cisco.com/web/techdoc/ucs/interoperability/matrix/matrix.html

Version History

Version	Date	Document version history
Version 1.0	June 2023	Initial version
Version 2.0	December 2023	Minor edit to conform to Epic guidelines

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations NetApp supports. Specific results depend on each customer's installation following published specifications.

NetApp