Reference Architecture

Data Center | Hybrid Cloud

Building a VMware Hybrid Cloud Platform

Using hardware ingredients from Intel, hybrid cloud software from VMware, and VMware Cloud on AWS, enterprises can accelerate workload deployment, simplify management, and boost their competitive edge

Intel Data Platform Group

Authors

Patryk Wolsza
vExpert, Cloud Solutions Architect

Karol Brejna
Senior Architect

Marcin Hoffmann
Cloud Solutions Engineer

Lukasz Sitkiewicz
Software Engineer

Contributors

Marek Małczak
Cloud Solutions Engineer

Ewelina Kamyszek
Tech. Undergrad Intern

Piotr Grabuszynski
Software Engineer

VMware

Authors

Rick Walsworth
VMware Cloud Sr. Product Line Marketing Manager

Enrique Corro
Data Science Staff Engineer, Office of the CTO

Christopher Martin
Cloud Solutions Engineer, VMware Cloud

Executive Summary

To be competitive and up to date with new technologies, all companies must overcome key challenges to accelerate their product development, reduce downtime and maintenance overhead, and compete more successfully at lower cost. Technology is now a centerpiece of every new change; the traditional approach for hosting applications and services cannot deliver innovation at the pace businesses require.

Meeting this challenge involves accelerating the entire software and hardware provisioning, deployment, and maintenance lifecycle along with application development, testing, and delivery. End-user self-service solutions are expected to reduce the time to market even more. VMware Cloud Foundation and Intel address these new market requirements by offering an easily deployable and manageable hybrid cloud platform for managing virtual machines (VMs) and orchestrating containers. This solution provides infrastructure and operations across private and public clouds with excellent performance and reliability from Intel® hardware components used for critical on-premises infrastructure.

This reference architecture describes the following:

- How to prepare and deploy a VMware Cloud Foundation environment connected to VMware Cloud on Amazon Web Services.
- How to take advantage of relevant Intel® technology such as Intel® Optane™ persistent memory.

Because the term “cloud computing” is now often associated with both VMs and the use of containerization, this reference architecture illustrates a variety of applications. These include popular VM-based warehousing solutions such as Oracle and Microsoft SQL, as well as container-based analytics and artificial intelligence (AI) workloads. These use cases highlight the flexibility of the solution and overall performance.

The intended audience for this reference architecture includes system administrators and system architects. Some experience with virtualization technology, networking (VMware NSX-T), and Kubernetes is assumed, as is an understanding of machine-learning and AI core concepts.

Table of Contents

Introduction: Why VMware Cloud Foundation for Your Hybrid Cloud?

Hybrid cloud infrastructure combines the benefits of an on-premises infrastructure with the flexibility and instant availability of the public cloud. This approach is gaining popularity as businesses adapt their existing computing resources to ever-changing demands and market needs. While there are many examples of services that are ideal for the public cloud, some workloads are better suited to staying on-premises (such as sensitive data that is required for machine-learning models). Increasingly, enterprises want a hybrid cloud option—as shown in Figure 1—for this flexibility and business agility. Hybrid cloud is becoming especially important as artificial intelligence (AI) and machine-learning workloads become increasingly prevalent.

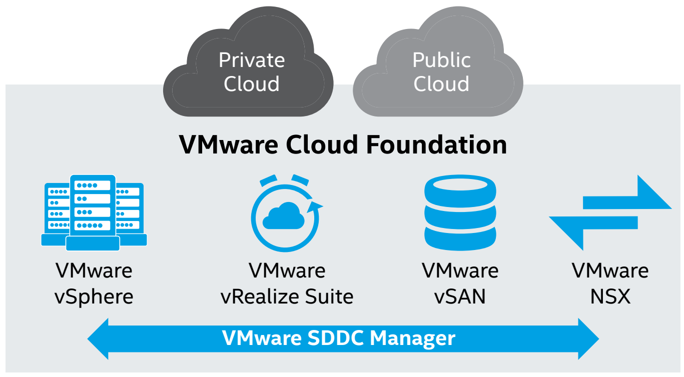


Figure 1. VMware Cloud Foundation is a cloud solution that is managed through VMware SDDC Manager and built on VMware vSphere, vRealize Suite, vSAN, and NSX.

With the end-to-end solution that Intel and VMware offer, enterprises can quickly launch database processing and AI, and scale workloads to accommodate future needs. The unified cloud solution presented in this reference architecture (see Figure 2, on the next page) can run containerized applications and traditional VMs, located in an on-premises data center as well as in the public cloud, such as on Amazon Web Services (AWS). The hybrid cloud nature of the solution allows enterprises to extend available resources and easily migrate workloads from on-premises to the cloud and back.

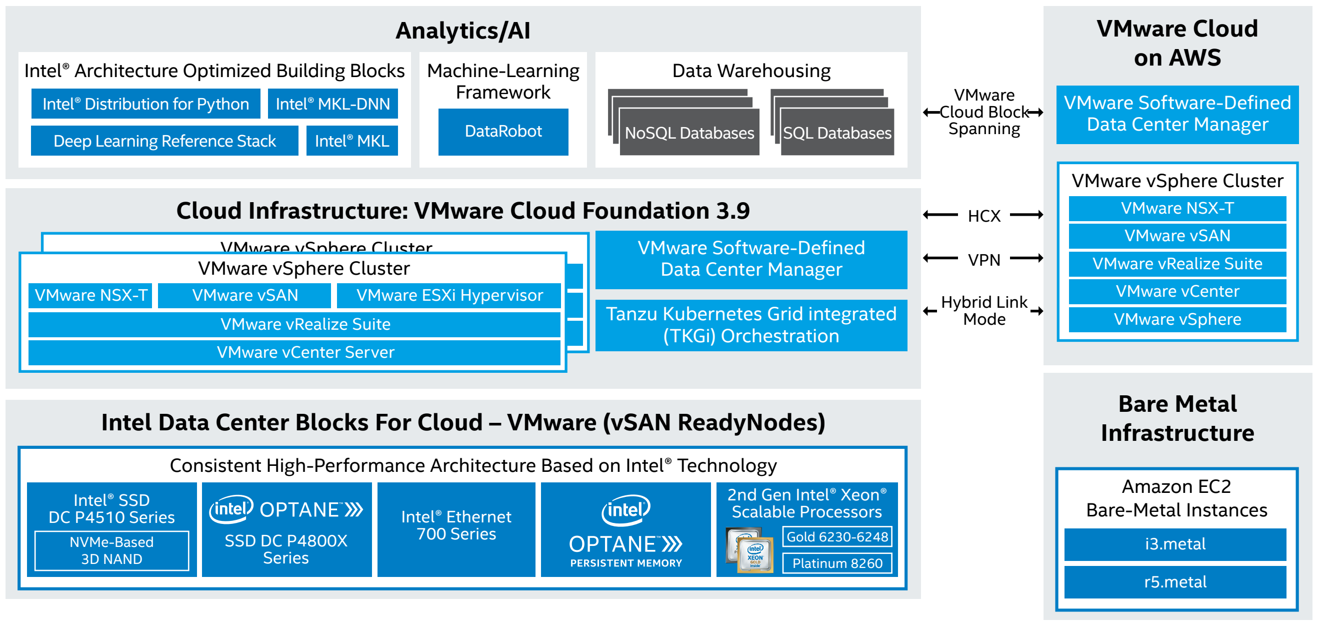

Analytics/AI
Intel® Architecture Optimized Building Blocks
Intel® Distribution for Python
Intel® MKL-DNN
Deep Learning Reference Stack
Intel® MKL
Machine-Learning Framework
DataRobot
Data Warehousing
NoSQL Databases
SQL Databases
Cloud Infrastructure: VMware Cloud Foundation 3.9
VMware vSphere Cluster
VMware vSphere Cluster
VMware NSX-T
VMware vSAN
VMware ESXi Hypervisor
VMware vRealize Suite
VMware vCenter Server
VMware Software-Defined Data Center Manager
Tanzu Kubernetes Grid integrated (TKGi) Orchestration
Intel Data Center Blocks For Cloud – VMware (vSAN ReadyNodes)
Consistent High-Performance Architecture Based on Intel® Technology
Intel® SSD DC P4510 Series
NVMe-Based 3D NAND
OPTANE SSD DC P4800X Series
Intel® Ethernet 700 Series
OPTANE PERSISTENT MEMORY
2nd Gen Intel® Xeon® Scalable Processors
Gold 6230-6248
Platinum 8260
VMware Cloud Block Spanning
HCX
VPN
Hybrid Link Mode
VMware Cloud on AWS
VMware Software-Defined Data Center Manager
VMware vSphere Cluster
VMware NSX-T
VMware vSAN
VMware vRealize Suite
VMware vCenter
VMware vSphere
Bare Metal Infrastructure
Amazon EC2 Bare-Metal Instances
i3.metal
r5.metal

Figure 2. Reference architecture building blocks for the VMware hybrid cloud platform.

Solution Overview

This reference architecture provides detailed configuration information for building a hybrid cloud. At a high level, the reference architecture consists of an optimized combination of Intel® hardware and VMware software.

Hardware Overview

The hardware stack for the solution is built on Intel® Server Board S2600WF0R platforms. The platforms include the latest generation of Intel® Xeon® Gold processors and Intel® Xeon® Platinum processors. For high-performance, all-flash software-defined storage, the reference architecture includes Intel® Optane™ SSD DC P4800X Series drives and Intel® SSD DC P4510 Series with NVM Express (NVMe) drives combined with Intel® Optane™ persistent memory. Intel Optane PMem introduces innovative memory technology that delivers large-capacity system memory and persistence. For an accelerated software-defined network, the platforms use 25 Gb/s Intel® Ethernet Converged Network Adapters X710-DA2.

Software Overview

From a software perspective, the solution consists of VMware Cloud Foundation, including VMware vSphere, VMware vSAN, VMware vRealize Suite, VMware NSX-T Data Center, and VMware Software-Defined Data Center (SDDC) Manager to provide infrastructure-as-a-service (IaaS) capabilities. In addition, Tanzu Kubernetes Grid Integrated (TKGi)[i] is used for a Kubernetes-based container solution.

VMware Cloud on AWS is used as the end destination for the hybrid architecture. VMware Hybrid Cloud Extension (HCX) enables VM migration, workload rebalancing, and protection between on-premises and cloud. In addition to business continuity, it provides network extension for multi-tier applications without changing the VM properties.

Technology Descriptions

This section describes the building blocks in each of the reference architecture's layers: hardware, cloud infrastructure, database building blocks, and analytics/AI building blocks. The platform benefits from Intel® Optane™ technology used throughout the entire stack, as shown in Figure 3.

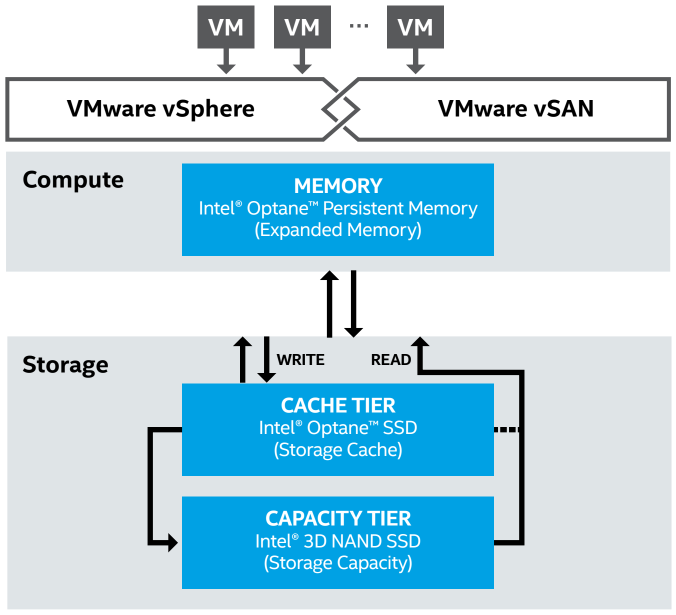

VM
VM
...
VM
VMware vSphere
VMware vSAN
Compute
MEMORY
Intel® Optane™ Persistent Memory (Expanded Memory)
Storage
WRITE
READ
CACHE TIER
Intel® Optane™ SSD (Storage Cache)
CAPACITY TIER
Intel® 3D NAND SSD (Storage Capacity)

Figure 3. The placement of Intel® Optane™ SSDs and Intel® Optane™ persistent memory within the architecture.
Source: principledtechnologies.com/VMware/VMware-HCI-Intel-Optane-VDI-0420.pdf

[i] Note: TKGi was formerly known as VMware Enterprise PKS. While this document uses the name TKGi in the text, we reference PKS documentation throughout this document because the rebranding took place after the reference architecture was completed.

Intel Hardware

Intel Optane Persistent Memory

Intel Optane PMem represents a new class of memory and storage technology. It is designed to improve the overall performance of the server by providing large amounts of persistent storage with low-latency access. Intel Optane PMem modules are DDR4-socket compatible and are offered in sizes not available with typical DDR4 DRAM products: 128, 256, and 512 GB per module. The default Base configuration does not use PMem. However, the Base configuration can be upgraded to use PMem modules without any additional hardware changes to gain a significant performance boost.[1]

Intel Optane PMem can work in two different operating modes (Memory Mode and App Direct Mode), depending on business need and/or application support, as shown in Figure 4. Regardless of the mode used, PMem provides capacity that is unavailable when using traditional DRAM modules. Operating modes can be configured using the platform BIOS or memory management tools. App Direct-Dual Mode is also possible by partitioning the memory pool. For additional information and guidelines, visit the Intel Optane persistent memory Quick Start Guide.

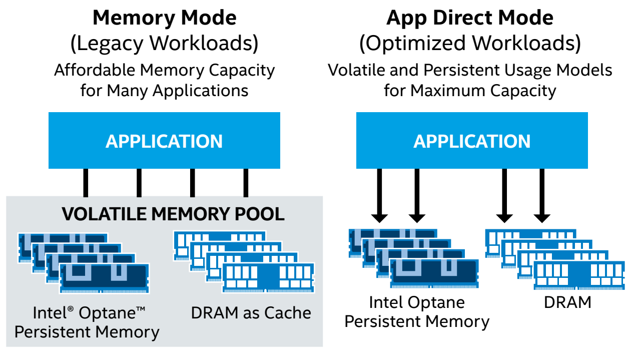

Memory Mode
(Legacy Workloads)
Affordable Memory Capacity for Many Applications
APPLICATION
VOLATILE MEMORY POOL
Intel® Optane™ Persistent Memory
DRAM as Cache
App Direct Mode
(Optimized Workloads)
Volatile and Persistent Usage Models for Maximum Capacity
APPLICATION
Intel Optane Persistent Memory
DRAM

Figure 4. Intel® Optane™ persistent memory operating modes.
Source: servethehome.com/2nd-gen-intel-xeon-scalable-launch-cascade-lake-details-and-analysis/intel-optane-dcpmm-memory-and-app-direct-modes

Memory Mode for Expansion of Regular DDR4 DRAM

This mode is ideal to expand the capacity of memory available to support more or larger VMs in virtual desktop infrastructure (VDI) deployments. This mode can also support higher quantities of "hot" data available for processing with in-memory databases, analytics, and other demanding workloads. It allows organizations to maintain larger amounts of data closer to the processor, with consistent, near-DRAM performance. In this mode, the PMem is visible to all applications (including the OS) as volatile memory, just as if it was regular DRAM. Its content is lost when the system is powered off. At the hardware level, there is a combination of PMem modules and standard DRAM; the DRAM acts as a cache for most frequently accessed data, and the PMem is used for providing capacity. Typically, the DRAM-to-PMem ratio ranges from 1:4 to 1:8. Only the PMem module's size is visible and usable by the applications and the OS. Since the CPU memory controller handles all the calls and manages the use of DRAM for cache, there are no additional requirements for the OS or applications to use Memory Mode. They are unaware that two types of memory are installed. This setup allows higher memory capacity and is more cost-effective. However, it can be slower with random access workloads than a much more expensive system with the same amount of DRAM-only memory.

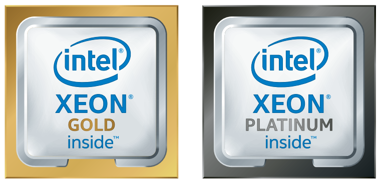
App Direct Mode for Low-Latency Persistent Memory

In-memory databases, analytics frameworks, and ultrafast storage applications are the types of workloads that greatly benefit from using App Direct Mode. In this mode, the DRAM and the PMem are both counted in the total system memory. This mode requires additional support from the OS and applications because there are two types of memory that can be used by the system independently. Low-latency operations should be directed to DRAM, while data that needs to be persistent—or structures that are very large—can be routed to the PMem. In this mode, the data stored on PMem modules is persistent, even when the system is powered off. Applications can access the PMem using standard OS storage calls as if it was a regular file system on a normal storage device (using this approach does not require the application to be App Direct-aware). Such usage can provide a noticeable speed boost, but PMem can provide even greater performance by accessing it as if it was DRAM. This is called the DAX Mode (Direct Access) and does not rely on file system calls, which makes it significantly faster. DAX Mode enables data to be written in less than a microsecond.[2] However, this mode requires the application to be written in a way that makes it PMem-aware.

2nd Generation Intel® Xeon® Scalable Processors

Today's modern enterprises are processing ever-increasing amounts of data. They need the compute power that can meet the data-centric demands of analytics, AI, and in-memory database workloads. 2nd Generation Intel Xeon Scalable processors are workload optimized for exactly these types of applications, with up to 56 cores per CPU and 12 DDR4 memory channels per socket. What's more, these processors support Intel Optane PMem, which enables affordable system memory expansion.

This reference architecture is available in a "Base" and a "Plus" configuration. The Base configuration uses the Intel® Xeon® Gold 6248 processor and optimally balances price and performance for mainstream workloads. The Plus configuration uses the Intel® Xeon® Platinum 8268 processor, which can efficiently handle high-density deployments and data-intensive, latency-sensitive workloads. Enterprises that need even higher performance can replace the default CPU with a higher-number SKU in either configuration.



This reference architeure features the Intel Xeon Gold processor for the Base configuration and the Intel Xeon Platinum processor for the Plus configuration.

Intel® SSD Data Center Family: Intel Optane SSDs and Intel® 3D NAND SSDs

To obtain the best performance from VMware vSAN, it is recommended to use high-performance Intel Optane SSDs for the cache layer, while the capacity layer can use large-capacity NVMe-based 3D NAND SSDs.

Intel Optane SSDs' unique design provides low latency, at least 30 drive-writes-per-day endurance.[3] These characteristics make them ideal for write-heavy cache functions.[4] Faster caching means enterprises can affordably and efficiently process bigger datasets to uncover important business insights.

The Intel® SSD DC P4510 Series is available in large capacities and uses Intel's 64-layer TLC 3D NAND technology to double the capacity available compared to its predecessor, the Intel® SSD DC P4500 Series. This increased density is key to supporting read-intensive operations. This SSD also provides high reliability and performance consistency.

Intel® VMD Technology for NVMe Drivers

Intel® Volume Management Device (Intel® VMD) enables serviceability of NVMe-based SSDs by supporting hot swap replacement from the PCIe bus without shutting down the system. It also provides error management and LED management (see Figure 5). Intel VMD is implemented using hardware logic provided inside the Intel® Xeon® processor. VMware vSphere 6.7 can use these functions with vSAN or other local or direct-attach storage (DAS) without the need to install additional vSphere Installation Bundles (VIBs). Intel VMD is a robust solution to NVMe SSD hot plug, but its unique value is that Intel is sharing this technology across the ecosystem, including system OEMs/ODMs, BIOS writers, PCIe switch vendors, SSD vendors, and OS and ISV vendors.

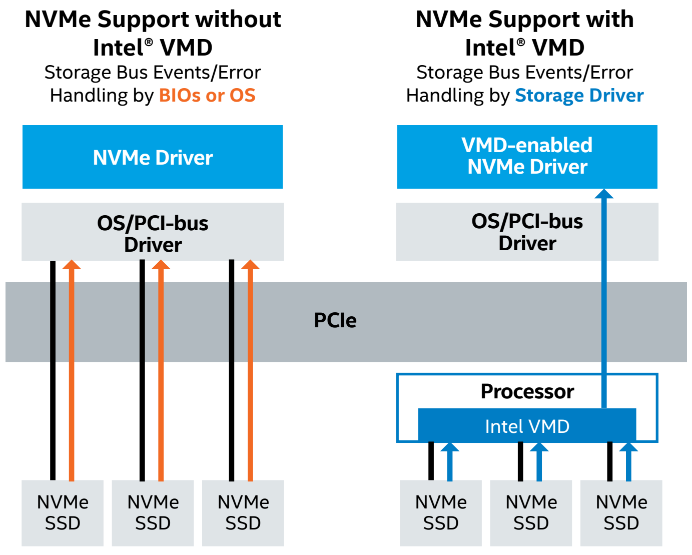

NVMe Support without Intel® VMD
Storage Bus Events/Error Handling by BIOs or OS
NVMe Driver
OS/PCI-bus Driver
NVMe SSD
NVMe SSD
NVMe SSD
NVMe Support with Intel® VMD
Storage Bus Events/Error Handling by Storage Driver
VMD-enabled NVMe Driver
OS/PCI-bus Driver
PCIe
Processor
Intel VMD
NVMe SSD
NVMe SSD
NVMe SSD

Figure 5. Intel® VMD handles the physical management of storage devices.
Source: colfax-intl.com/Servers/CX1265i-NVMe-XR7

Intel® Ethernet Connections and Adapters

To accelerate the performance of the VMware hybrid cloud platform, this reference architecture uses the Intel® Ethernet 700 Series. These Intel Ethernet products deliver validated performance that can meet enterprise-level requirements for data resiliency, service reliability, and ease-of-provisioning.[5,6,7,8]

For the physical networking layer, this reference architecture recommends using two switches for the data plane and one switch for the management plane. Data plane switches should support VLANs and Jumbo Frames. An enterprise-level router solution is also required to provide routing capabilities for the multiple VLANs required by VMware Cloud Foundation.

Data Plane Development Kit

Developed by Intel, Data Plane Development Kit (DPDK) is a set of Intel® architecture-optimized libraries and drivers that accelerate packet processing and the ability to create packet forwarders without the need for costly custom switches and routers. It gives application developers the ability to address data plane processing needs, all in software and on general-purpose Intel® processors. The DPDK can:

- Receive and send packets within a minimum number of CPU cycles
- Develop fast packet capture algorithms
- Run third-party fast path stacks
- Provide software pre-fetching, which increases performance by bringing data from memory into cache before it is needed

VMware infrastructure components that employ and benefit from DPDK include VMXNET3 para-virtual vNICs, VDS, and direct assignment features (Direct Path I/O or SR-IOV). VMware Cloud on AWS also supports DPDK.

To learn more about DPDK, visit the Intel® Developer Zone.

Cloud Infrastructure

VMware Cloud Foundation

VMware Cloud Foundation provides a simplified path to hybrid cloud through an integrated software platform for both private and public cloud environments. It offers a complete set of software-defined services for compute, storage, network, and security, along with application-focused cloud management capabilities. The result is a simple, security-enabled, and agile cloud infrastructure on-premises and in as-a-service public cloud environments.

VMware vRealize Suite

VMware vRealize Suite is a multicloud cloud management solution providing IT organizations with a modern platform for infrastructure automation, consistent operations, and governance based on DevOps and machine-learning principles.

VMware SDDC Manager

Software-Defined Data Center (SDDC) Manager manages the bring-up of the VMware Cloud Foundation system, creates and manages workload domains, and performs lifecycle management to keep the software components up to date. SDDC Manager also monitors the logical and physical resources of VMware Cloud Foundation.

VMware vSphere

VMware vSphere extends virtualization to storage and network services and adds automated, policy-based provisioning and management. vSphere is the starting point for building an SDDC platform. VMware vSphere 7 with Kubernetes enables streamlined development, agile operations, and accelerated innovation for all enterprise applications.

VMware NSX-T Data Center

NSX-T Data Center (formerly NSX-T) is the network virtualization platform that enables a virtual cloud network with a software-defined approach. Working like a network hypervisor, it reproduces a complete set of Layer 2 through Layer 7 networking services: routing, switching, access control, firewalls, quality of service (QoS), and Dynamic Host Configuration Protocol (DHCP) in software. All these components can be used in any combination to create isolated virtual networks on demand. The services can then be extended to a variety of endpoints within and across clouds.

Tanzu Kubernetes Grid integrated (TKGi)

Tanzu Kubernetes Grid Integrated Edition (TKGi), formerly known as VMware Enterprise PKS, is a Kubernetes-based container solution with advanced networking, a private container registry, and life cycle management. TKGi simplifies the deployment and operation of Kubernetes clusters so you can run and manage containers at scale on private and public clouds. With TKGi, you can provision, operate, and manage Kubernetes clusters using the TKGi Control Plane.

VMware Cloud on AWS

VMware Cloud on AWS is a hybrid cloud solution that allows easy extension, migration, modernization of applications, and protection of applications in the public cloud. The infrastructure is delivered by the same vSphere-based SDDC stack that is used on-premises. VMware Cloud on AWS includes vSphere and vCenter, vSAN, vRealize, NSX-T Data Center, HCX, and VMware Site Recovery Manager (SRM) to provide hybrid connectivity and disaster-recovery-as-a-service (DRaaS). Everything is offered as a service to customers to rapidly deploy infrastructure on a modern Nitro system-based Amazon EC2 elastic, bare-metal infrastructure. The solution takes advantage of existing tools, processes, and familiar VMware technologies, along with native integration with AWS services. This makes it easy to adopt, greatly reduces service disruption associated with migrating critical services to the cloud, and eliminates the need for rearchitecting the environment to suit a public cloud infrastructure.

The enterprise-grade infrastructure is delivered as a service, with the SDDC provision time under two hours[9] and has pre-configured vSAN storage, networking, compute, and security. VMware Cloud on AWS can also autoscale nodes as needed, depending on CPU, memory, and storage requirements. Typically, autoscaled nodes can be scaled up or down in just a few minutes.

VMware HCX

VMware HCX is an application mobility platform that is designed for simplifying application migration, workload rebalancing, and business continuity across data centers and clouds. It enables customers to migrate workloads between public clouds and data centers without any modification to applications or VM configurations. It provides full compatibility with the VMware software stack and helps make the migration simple, highly secure, and scalable.

The HCX Multi-Site Service mesh provides a security-enabled pipeline for migration, extension, and VM protection between two connected VMware HCX sites (see Figure 6). It can be used to extend VLANs and retain IP and MAC addresses, as well as existing network policies, during migration between two sites. It also enables flexibility when planning complex, growing workloads across physical sites.

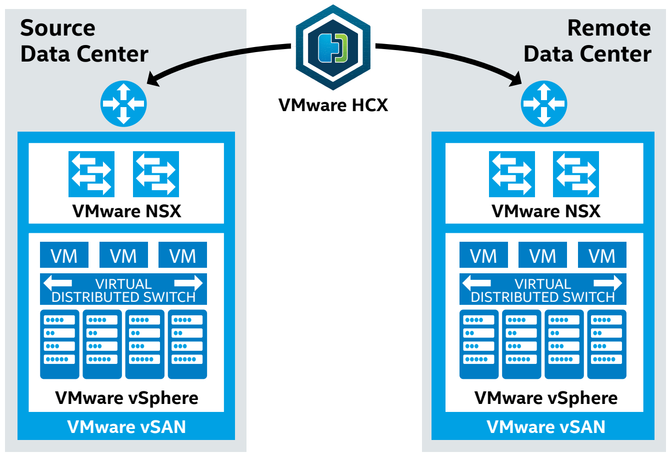


Figure 6. VMware HCX overview.
Source: docs.vmware.com/en/VMware-HCX/services/install-checklist/GUID-DE0AD0AE-A6A6-4769-96ED-4D200F739A68.html

Data Warehousing Building Blocks

Data warehouses are considered one of the core components of business intelligence. They are a central location to store data from one or more disparate sources as well as current and historical data. Numerous methods can be used to organize a data warehouse. Hardware, software, and data resources are the main components of this architecture, and VMware Cloud Foundation is an excellent platform on which to deploy data warehousing solutions (see Figure 7).

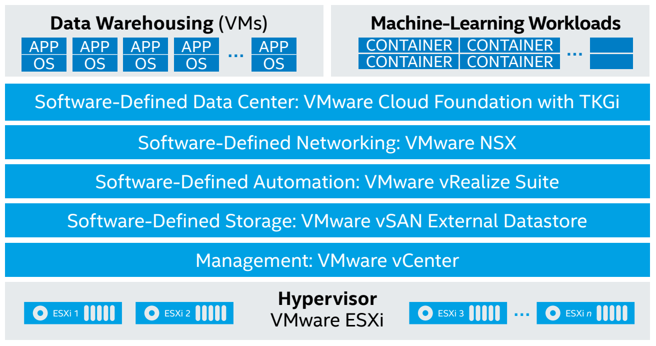

Data Warehousing (VMs)
APP OS
APP OS
APP OS
APP OS
...
APP OS
Machine-Learning Workloads
CONTAINER CONTAINER
CONTAINER CONTAINER
...
Software-Defined Data Center: VMware Cloud Foundation with TKGi
Software-Defined Networking: VMware NSX
Software-Defined Automation: VMware vRealize Suite
Software-Defined Storage: VMware vSAN External Datastore
Management: VMware vCenter
Hypervisor
VMware ESXi
ESXi 1
ESXi 2
ESXi 3
...
ESXi n

Figure 7. VMware Cloud Foundation is a platform that can be used for all data analytics, AI, and machine learning workloads.

To illustrate how the VMware hybrid cloud platform supports data warehousing, this reference architecture uses the popular industry-standard Oracle DB 19c and Microsoft SQL 2019 solutions as example workloads. In addition to traditional SQL services, VMware Cloud Foundation also accommodates NoSQL databases, and it is an efficient platform for running the Apache Hadoop framework and all of its related services that support big data and data mining.

The entire platform runs on vSAN, which provides additional storage policy configuration options in terms of data redundancy (multiple redundancy levels are available). vSAN can be used by both platform administrators and end users (such as when processing persistent volume claims on Kubernetes deployments) to obtain the maximum usage of the entire platform storage system.

Analytics and AI Building Blocks

Enterprises need high-performance data analytics and AI to remain competitive. They require flexible solutions that can run traditional data analytics and AI applications. The VMware hybrid cloud platform includes components that take advantage of performance optimizations for Intel hardware in a VMware infrastructure. Intel supports developing machine-learning workloads at multiple layers in the solution stack. These building blocks enable enterprises to quickly operationalize analytics, AI, and machine-learning workloads because they are already optimized for Intel architecture and have been verified with multiple production deployments. Therefore, enterprises can immediately begin to use them.

This reference architecture demonstrates how to train a machine-learning model and then how it can be deployed on a hybrid cloud cluster.

Intel® Distribution for Python

Python is a general-purpose programming language that is easy to master due to its simple syntax. It also includes a broad ecosystem of libraries (scientific, data transformation, and machine-learning). These features allow for both prototyping solutions (turning ideas into executable code) and running production-grade algorithms. Python is a popular choice for data science and machine-learning use cases (especially for deep learning). Intel provides a performance-oriented distribution of Python that can accelerate Python applications. Using the Intel® Distribution for Python, enterprises can:

- Achieve faster Python application performance—right out of the box—compared to the standard distribution, with minimal or no changes to code.[10]
- Accelerate NumPy, SciPy, and scikit-learn with integrated Intel® performance libraries such as Intel® Math Kernel Library (Intel® MKL) and Intel® Data Analytics Acceleration Library (Intel® DAAL).
- Access the latest vectorization and multithreading instructions, Numba and Cython, composable parallelism with Intel® Threading Building Blocks (Intel® TBB), and more.

See https://software.intel.com/en-us/distribution-for-python for more details.

Intel MKL

Intel MKL optimizes code with minimal effort for future generations of Intel processors. It is compatible with a wide variety of compilers, languages, operating systems, and linking and threading models. This ready-to-use math library accelerates math processing routines, increases application performance, and reduces development time by:

- Featuring highly optimized, threaded, and vectorized math functions that maximize performance on each Intel processor family.
- Using industry-standard C and Fortran APIs for compatibility with popular BLAS, LAPACK, and FFTW functions—no code changes required.
- Dispatching optimized code for each processor automatically without the need to branch code.
- Providing Priority Support that connects enterprises directly to Intel engineers for confidential answers to technical questions.

See https://software.intel.com/en-us/mkl for more details.

Intel DAAL

Intel DAAL is a library of Intel architecture-optimized building blocks that cover all stages of data analytics. These stages include data acquisition from a data source, preprocessing, transformation, data mining, modeling, validation, and decision making. This library helps to reduce the time it takes to develop high-performance data science applications by supporting:

- Highly optimized machine-learning and analytics functions.
- Simultaneous ingestion of data and computing results for high-throughput performance.
- Batch, streaming, and distribution use models to meet a range of application needs.
- The same API for application development on multiple operating systems.

See https://software.intel.com/en-us/daal for more details.

Intel® Distribution of OpenVINO™ Toolkit

The Intel Distribution of OpenVINO toolkit provides developers with excellent neural network performance on a variety of Intel processors. It helps to further unlock cost-effective, real-time vision applications. The toolkit enables deep-learning inference and easy heterogeneous execution across multiple Intel architecture-based platforms, providing implementations across cloud architectures to edge devices, and across all types of computer vision accelerators—CPUs, integrated GPUs, Intel® Movidius™ Neural Compute Sticks, and Intel® field-programmable gate arrays (Intel® FPGAs)—using a common API. The OpenVINO toolkit of functions and preoptimized kernels helps speed time to market.

Deep Learning Reference Stack

The Deep Learning Reference Stack (see Figure 8) is an integrated, highly performant open source stack optimized for Intel® Xeon® Scalable processors. It was created to help AI developers deliver the best experience on Intel architecture. This stack reduces the complexity that is common with deep-learning software components, provides flexibility for customized solutions, and enables enterprises to quickly prototype and deploy deep-learning workloads.

The latest release of the Deep Learning Reference Stack (at the time of publication, DLRS V5.X) supports the following features:

- Optimized TensorFlow 1.X and TensorFlow 2.X, two versions of an end-to-end open source platform for machine learning.
- Optimized PyTorch, an open source machine-learning framework that accelerates the path from research prototyping to production deployment.
- PyTorch Lightning, which is a lightweight wrapper for PyTorch designed to help researchers set up all the boilerplate state-of-the-art training.
- Transformers, a state-of-the-art Natural Language Processing (NLP) for TensorFlow 2.X and PyTorch.
- Intel Distribution of OpenVINO toolkit, which delivers improved neural network performance on Intel processors, helping unlock cost-effective, real-time vision applications.
- Intel® Deep Learning Boost with Intel® Advanced Vector Extensions 512 Vector Neural Network Instructions, designed to accelerate deep neural network-based algorithms.
- Deep Learning Compilers, an end-to-end compiler stack.

This reference architecture demonstrates how to use the Deep Learning Reference Stack and shows the performance gains from using the version of TensorFlow optimized for Intel architecture.

DataRobot

This solution demonstrates DataRobot, a popular automated machine-learning platform that takes advantage of optimizations for Intel architecture.[11] Organizations worldwide use DataRobot to empower the teams they already have in place to rapidly build and deploy machine-learning models and create advanced AI applications. With a library of hundreds of powerful open source machine-learning algorithms, the DataRobot platform encapsulates many best practices and helps to accelerate and scale data science capabilities while increasing transparency, accuracy, and collaboration.

Several DataRobot features make it a popular choice in the AI market:

- **Selecting the proper model** for a given problem is often tedious and difficult. Automated machine learning provided by DataRobot makes it possible to quickly and efficiently build and train tens or even hundreds of algorithms. After the training is completed, DataRobot presents the models in a list, ranked in order of the selected performance metric. To make it even easier to choose a model, DataRobot automatically flags which model is most accurate and which model is best suited for deployment.
- **Model tuning** is easy with DataRobot. The tool automatically runs dozens of models with preset settings that have been thoroughly tested to verify that they result in highly accurate models. Enterprises can take advantage of this pre-work so they can focus on choosing the one that is most accurate for their data. DataRobot also makes it easy to manually tune models.
- DataRobot makes it easy to **explain AI** through human-friendly visual insights and automated model documentation with blueprints that describe each step in the modeling process and the algorithms used. Enterprises can evaluate any model using several tools.
- All DataRobot models are **ready for production** and can be deployed with a single click to make AI fully operational. Enterprises can monitor models using a centralized dashboard to view service health and usage in real-time. They can also manage model accuracy to easily understand which features have drifted and deploy updates with no service interruption.

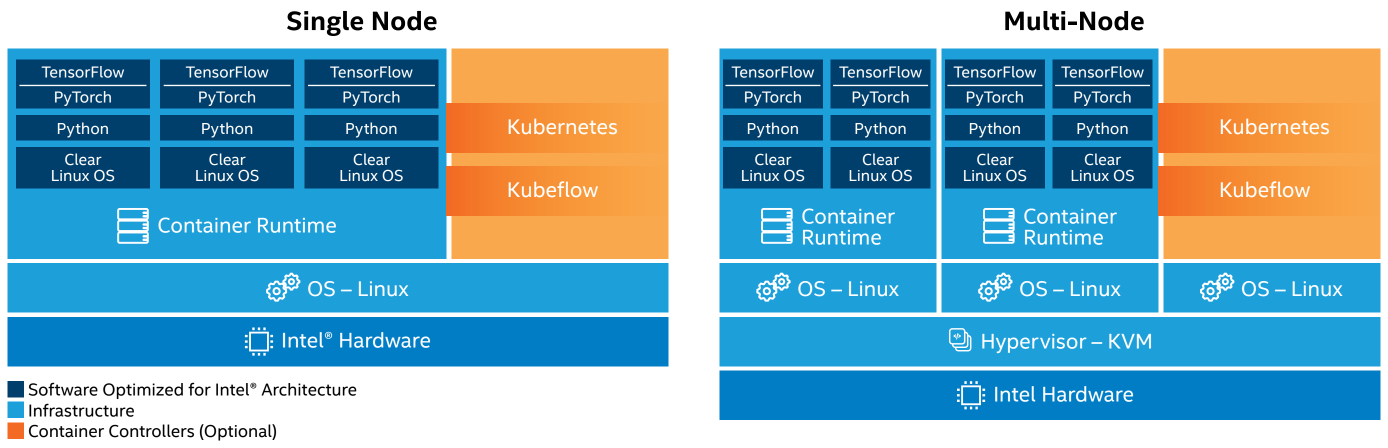


Figure 8. The Deep Learning Reference Stack accelerates AI deployment.

Self-service Application Catalog

This reference architecture allows for fast and easy deployment of applications. The solution is based on technologies such as VMware vSphere, TKGi, and Docker—all of which are de-facto industry standards and have wide community and enterprise adoption. This allows for leveraging open-sourced tools and frameworks even further. One component of the solution is a self-service application store called Bitnami Kubeapps, which is a web application designed for deploying and managing applications in Kubernetes clusters (see Figure 9).

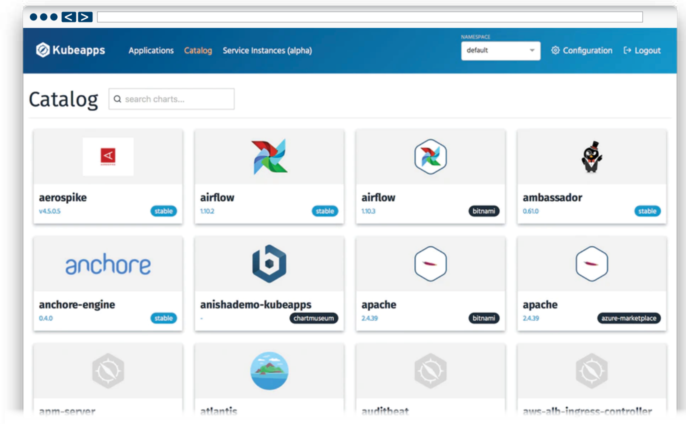


Figure 9. Bitnami Kubeapps applications catalog.

Kubeapps provides a built-in catalog of applications, which is based on Helm chart repositories that can be browsed and deployed. With this tool, you can add your own Helm chart repository using a Harbor registry to manage your own set of applications. In this way, users can choose from applications that their IT department has selected and packaged, or use any freely available repositories. This simple approach enables non-technical users to deploy and use the application they require (examples include Apache Spark and Jupyter Notebooks).

For more information on how to use Kubeapps, refer to the "Kubeapps Usage" section in Appendix A: Solution Features Validation and Benchmarking.

Platform-Verified Data Warehousing Workload Performance

This section discusses possible platform usage, features, and benchmark results for data warehousing using chosen database solutions.

Microsoft SQL Server on Linux Containers

There are official Microsoft container images for Microsoft SQL Server on Linux that make it possible to deploy Microsoft SQL Server using Docker Engine or Kubernetes for fast and easy application deployment. The Linux-based container uses Microsoft SQL Server 2017 Developer Edition on top of an Ubuntu 16.04 base image. There are existing Helm charts for automated deployment as well as ready-to-use Microsoft SQL applications available in the Bitnami Kubeapps self-service app store.

For details about Microsoft SQL Server on Linux, refer to the Microsoft SQL Server documentation.

The availability of Microsoft SQL Server running on a container, combined with automated TKGi clusters and Bitnami Kubeapp, creates a powerful platform for administrators, developers, and end users. Anyone can now deploy apps as well as the environment on demand and with ease.

Microsoft SQL Server on a Standalone VM—Overview

For the purposes of a data warehouse example, we tested Microsoft SQL Server using both the Base and Plus configurations. Multiple instances of HammerDB, located on a separate cluster, were used to generate load.

Microsoft SQL Server is a database management system that uses Transact-SQL (T-SQL) as a query language. T-SQL is an SQL language extension that allows the use of basic programming constructions like variables, loops, and conditional instructions. For test purposes, we used Microsoft SQL Server 2019.

HammerDB is an open source tool that uses specially generated tables and virtual users to measure and test the load of databases. It can connect to different types of database engines. Although it is possible to run a HammerDB instance directly on Microsoft SQL Server, we recommend creating a separate instance of Microsoft Windows Server and testing the Microsoft SQL Server databases remotely.

For more information, visit the official HammerDB webpage and GitHub.

Microsoft SQL Server with App Direct Mode—Configuration and Benchmark Results

PMem running in App Direct Mode retains its content through power cycles, even when system power goes down in the event of an unexpected power loss or system crash. There are two methods for using App Direct Mode:

- **Block Access Mode.** In this configuration, PMem behaves like standard storage. All data requests go through the regular storage stack (PMem is seen as just another hard drive from the OS perspective). The resulting storage space is very fast and can be used by any application.
- **Direct Access (DAX) Mode.** In this configuration, PMem modules operate like DRAM to achieve the lowest possible latency by bypassing the storage stack of the OS. However, only applications that support DAX can benefit from this additional performance. Microsoft SQL Server 2019 is one such application. Our tests focus on testing with this mode.

Both modes make it possible to use PMem as storage for data warehousing purposes. The resulting system benefits from low latency and very fast read and write speeds. Whenever possible, for best performance, DAX Mode should be used instead of Block Access Mode. On the other hand, Block Access Mode is ideal for application compatibility if additional storage performance is needed, but the specific application does not yet support DAX Mode. Even though the number of applications that directly support PMem with DAX Mode is growing, the Block Access Mode offers flexibility, ease of use, and performance gains compared to legacy solutions.

The example workload used in testing this reference architecture is based on two main test scenarios:

- The **Base scenario** was executed on a regular, entry-level Intel architecture platform that offers a good starting point for all workload types. However, this platform does not have PMem installed. We refer to this platform as the Base cluster.
- The **Plus scenario** was based on a cutting-edge Intel Xeon Platinum processor, with additional capacity drives and PMem modules using App Direct Mode. This platform illustrates the full potential of this reference architecture. We refer to this platform as the Plus cluster.

Note the following:

- The Plus cluster nodes have more PMem storage space available than the actual Microsoft SQL Server instances require (the utilization should be less than 75 percent in the case of a four-node cluster). This overprovisioning enables easy maintenance and VM migrations. Data stored in App Direct Mode is physically located on the same VMware ESXi node that hosts the particular Microsoft SQL Server VM. If that VM is to be migrated (such as for load balancing or maintenance reasons), the target ESXi node must have sufficient free space in its PMem to store all the App Direct data of the migrated VM.
- Data on PMem modules is not distributed like it is with vSAN. If the node goes down for whatever reason, the data will not be accessible until the node is rebooted.
- VMware does not apply any replication mechanisms for App Direct data. If the node hardware fails, the data will be lost. For that reason, enterprise production deployments must use replication or backup methods supported by Microsoft SQL Server to implement high availability.

As mentioned before, Direct Access Mode is the fastest mechanism and Microsoft SQL Server supports this feature. However, this is not true for all file types. The database should be split into multiple files. Database files are put on PMem configured in DAX Mode. Log files are put on PMem configured in Block Access Mode.

Hybrid Buffer Pool (see Figure 10) is a feature introduced in both the Windows and Linux versions of Microsoft SQL Server 2019. It enables reference to data pages on the SQL database files, which reside on PMem, instead of copying pages into a DRAM-based buffer pool. Usage of memory-mapped I/O by PMem also allows access to pages without using the I/O stack of the OS, so it provides performance benefits. To enable the Hybrid Buffer Pool feature for an instance of Microsoft SQL Server, follow the instructions provided by Microsoft.

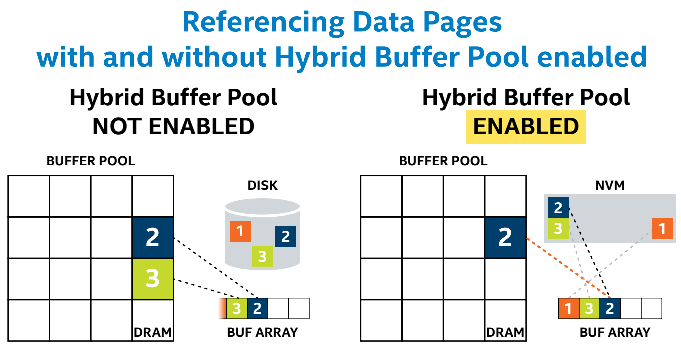


Figure 10. The Hybrid Buffer Pool feature improves I/O performance by changing the way you address data in memory.
Source: docs.microsoft.com/en-us/sql/database-engine/configure-windows/hybrid-buffer-pool?view=sql-server-ver15

Before you can start a Microsoft SQL Server benchmark, you must prepare the appropriate storage configuration. To achieve optimal results, we recommend distributing the Microsoft SQL Server databases across eight disks, as follows:

- Four disks[ii] for data files, 18 GB each with the data split into 16 files spread equally across the four disks (four data files per disk)
- Two disks[ii] for TempDB files, 200 MB each
- One disk[ii] for the TempLog, 100 MB
- One disk[ii] for the transaction log, 10 GB

The above proposed disk sizes are optimal for a test scenario involving 750 warehouses. To deploy PMem modules (NVDIMMs) for VMs, follow the recommendations contained in the "Preparing a VM for Intel Optane Persistent Memory" section, later in this document. After creation, each data disk was formatted with NTFS and DAX Mode support, and each log disk was formatted with NTFS with Block Access Mode support. For a detailed listing of the NVDIMM drives use, see Appendix B: Microsoft SQL on Windows Benchmark Configuration.

[ii] Note: In this context, "disk" refers to individual volumes of PMem.

Testing Methodology

Our testing benchmarks were performed on VMs using the configuration described above. For each Microsoft SQL Server VM, there was a separate VM with HammerDB installed, for load-generation purposes. We placed HammerDB on a different cluster so that it would not interfere with Microsoft SQL Server resource usage. Scaling was done by increasing the total number of such VMs in the cluster, thereby increasing the total number of data warehouses stored and stressed in the cluster.

For both clusters, we used VMs with eight virtual CPUs (vCPUs) and 44 GB of DRAM. For the Base cluster, we started with five VMs on a single ESXi node, and then increased this number by placing an additional five VMs on the second node of the cluster, iterating until we reached the total number of 20 VMs on all four nodes. The Distributed Resource Scheduler (DRS) was enabled for all non-Microsoft SQL Server VMs, while the Microsoft SQL Server VMs were bound to particular hosts. Our choice for the number of vCPUs, DRAM size, and number of VMs was intended to maximize the scalability and stability of the solution; DRAM and CPU resources are shared among other services within the cluster (including vSAN, NSX-T Edge VMs, and TGKi). Oversubscription of DRAM or CPU resources has a negative impact on vSAN performance, increasing the latency of reads and writes, which would negatively impact performance of all other services.

For the Plus configuration, the additional CPU resources and PMem allowed us to place additional instances of Microsoft SQL Server VMs on each ESXi node—eight instances each. The majority of the data was therefore placed locally in App Direct Mode on the ESXi host, resulting in a lesser load on vSAN. We had to decrease the DRAM size for each VM to keep the overall memory usage below 90 percent. This ensured that the OS, vSAN, and NSX-T services ran uninterrupted.

Base and Plus Cluster Benchmark Results

Even with the increased capacity, the Plus cluster delivered superior performance (up to 3.34X as many transactions per minute (TPM) when compared to the best result from the Base cluster (see Figure 11).[12]

We also explored the clusters' latency and their ability to meet service-level agreement (SLA) values. These values include CPU Time: Requests and CPU Time: Total. On both clusters, at least 98 percent of requests spent less than 5 ms in the CPU for all test scenarios (orange line in Figure 12).[13] More telling are the CPU Time: Total results. The Base cluster was unable to meet our SLA value of 80 percent of requests spending less than 20 ms total time in the CPU. However, for the Plus cluster, almost all the batches met this SLA value, with only a small drop during the most resource-consuming scenario.[14]

Conclusion for App Direct Mode

The results from the benchmark show that we were able to run not only more Microsoft SQL Server instances and data warehouses per each ESXi node—achieving 1.6X better density, but also obtain superior performance of the entire platform on the Plus cluster—up to 3.34X more TPM. At the same time, the Base cluster offers flexibility, high availability, and disaster recovery possibilities by fully utilizing VMware Cloud infrastructure. These capabilities could not be used with PMem modules at the time this reference architecture was published. This limitation made it impossible to perform live migration of VMs that were configured with App Direct Mode. Therefore, the high performance of the Plus cluster in a virtual environment with the use of PMem in App Direct Mode was possible only on-premises.

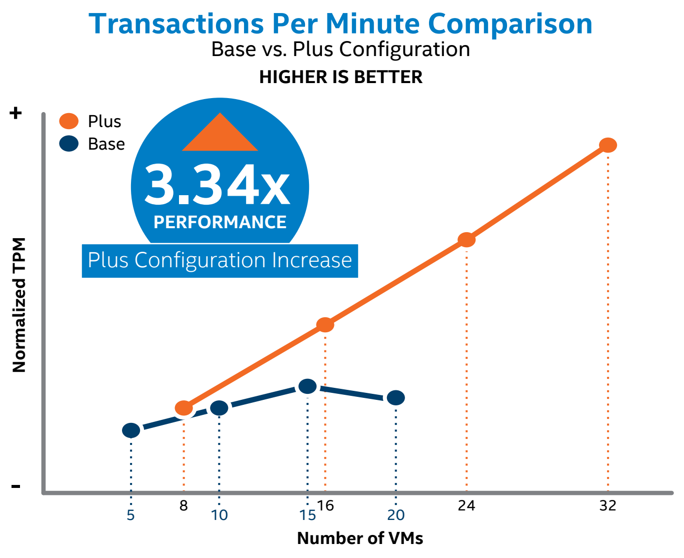


Figure 11. The Plus cluster achieved up to 3.34X more TPM compared to the Base cluster, and is highly scalable.

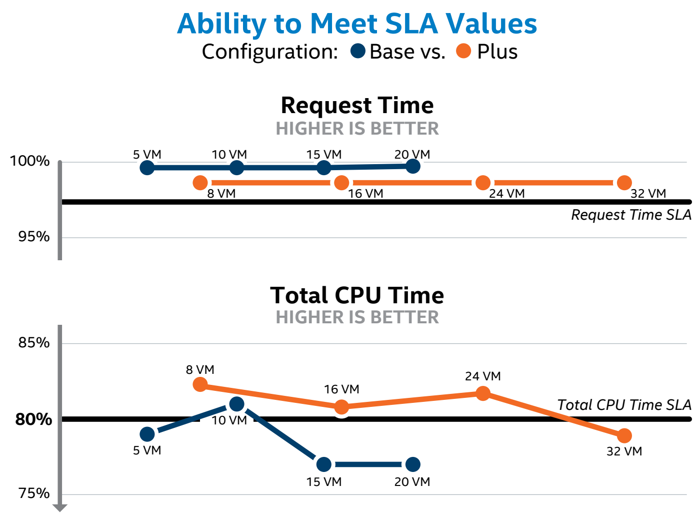


Figure 12. The Plus configuration consistently achieved its SLA as we added VMs, whereas the Base cluster did not.

Microsoft SQL Server with Memory Mode

The Internet of Things (IoT), machine-learning, and AI technologies often require fast access to large datasets. Size limitations of regular DRAM may be a bottleneck when scaling up in-memory database deployments. Shared storage with clustering is an alternative, but it can increase total costs, incurs some performance penalty, and contributes to additional management complexity. Intel Optane PMem in Memory Mode offers high memory densities that enable more efficient use of hardware through consolidation. This enables scaling up an existing environment instead of scaling out. With the additional memory available per single-server node, organizations can consolidate smaller DRAM-only Microsoft SQL Server VMs to fewer nodes equipped with DRAM and PMem and upgrade those servers with higher capacity memory as the databases grow. With up to 4.5 TB of PMem per socket, a dual-socket system with 512 GB PMem modules can be equipped with 6 TB of memory on a two-socket server. Increased memory density enables more data to be stored and accessed at near-DRAM speeds, making low-latency access to larger amounts of data a reality.

This reference architecture did not test Memory Mode benchmarking with Microsoft SQL Server because the App Direct Mode with DAX enabled provides a significant performance boost for Microsoft SQL Server. In contrast, Memory Mode provides capacity and density increases for large-scale workloads with near-DRAM performance.[15] Keep in mind however, that when using Memory Mode, you can build hardware configurations that are not possible to replicate with DRAM-only setups.

Database Warehousing Using Oracle Database with Intel Optane Persistent Memory

Oracle provides one of the industry-leading database platforms for the enterprise. Its databases are used by many large organizations as the basis for information management platforms. Customers have successfully deployed Oracle-based database warehousing by virtualizing extremely resource-intensive workloads using VMware ESXi and vSphere. This reference architecture provides a description of an example Oracle setup on VMware that illustrates the benefits of running Oracle on vSphere and managing large datasets more efficiently.

Ease of Manageability

A virtualized Oracle environment improves manageability by providing, among others, the following options:

- **Consolidation.** VMware technology allows multiple application servers to be consolidated onto one physical server, with little or no decrease in overall performance.
- **Ease of provisioning.** VMware virtualization encapsulates an application into an image that can be duplicated or moved, greatly reducing the cost of application provisioning and deployment. Furthermore, multiple instances of Oracle Database can be delivered (different versions of Oracle Database; different purposes such as development, QA, or production; or different end users).
- **Workload migration.** The vSphere vMotion feature enables customers to live-migrate workloads from source to destination ESXi hosts with minimal downtime, which simplifies common operations such as hardware maintenance.
- **Scaling up.** With vSphere, a VM can easily take advantage of multicore processors. If needed, vCPUs can be added to increase a VM's compute power.
- **High availability.** With Distributed Resource Scheduler, automatic failover and load balancing of VMs within a cluster are available out of the box. A more balanced cluster helps ensure better resource usage, and even the negative impact from complete node failure is reduced to a minimum as VMs are automatically moved from failed nodes and started on healthy systems.

Handling Large Datasets in Memory

As more and more data is stored in modern data warehouses, organizations need solutions that scale proportionally. Building clusters and distributing data across the system is one of the common approaches to scale, although it may not be desirable from either a performance or total cost point of view.

Oracle Database can be configured to use Intel Optane PMem in Memory Mode or App Direct Mode. In Memory Mode, Oracle Database can access 1.5 TB, 3 TB, or even 6 TB of memory for its in-memory operations, such as in-memory joins, in-memory aggregations, and in-memory column format data storage. In App Direct Mode, Oracle Database can use PMem as fast disk drives to store +DATA and +REDO files.

Platform-Verified Analytics/AI Workload Performance

The following sections discuss deep-learning inference and machine-learning model training workloads.

Deep-Learning Inference

Image classification is one of the most popular use cases for deep learning. Our tests benchmarked the ResNet50 v1.5 and Inception v3 topologies, using the TensorFlow distribution from the Deep Learning Reference Stack with Intel's Model Zoo pretrained models. For detailed instructions regarding the benchmark, refer to the "Running the TensorFlow Benchmark with a Deep Learning Reference Stack Container (Inference Benchmarks Reproduction)" section in Appendix A: Solution Features Validation and Benchmarking.

We ran two experiments, first with a fat VM and second with a TKGi instance with multiple workers as smaller VMs. Both scenarios utilized the entire physical node available through VMware software. Fat VMs used 80 vCPUs for the Base configuration and 96 vCPUs for the Plus configuration. The Kubernetes clusters used up to six workers with 16 vCPUs for both the Base and Plus configurations.

Fat VM Results

For the fat VM, we compared throughput from the Deep Learning Reference Stack container against the throughput from the standard TensorFlow container. We achieved a 2.2X improvement for Base and 2.5X for Plus using a Deep Learning Reference Stack container with the ResNet 50 v1.5 topology at fp32 precision, relative to the performance on respective configurations using a standard Tensorflow container (see Figure 13).[16] For the Inception v3 topology at fp32 precision, we achieved a 2.4X improvement for the Base configuration and 3X for the Plus configuration using the Deep Learning Reference Stack, relative to standard Tensorflow performance on respective configurations (see Figure 14).[17] These results highlight the significant benefits that accrue from using software optimized for Intel architecture.

These figures show the throughput of the ResNet 50 v1.5 and Inception v3 topologies are much higher when using the Deep Learning Reference Stack container. This demonstrates the effectiveness of the Deep Learning Reference Stack container optimizations to maximize Intel processor utilization.

Kubernetes Results

For the Kubernetes cluster provisioned by TKGi, we measured throughput scaling for the Deep Learning Reference Stack container by running the benchmark on one to six worker VMs in parallel for the deep-learning workloads. We observed significant improvement in throughput with additional jobs running in parallel, demonstrating the effectiveness of ESXi scheduler. In a multi-node system, as the VMs running the workload are scaled, the overall throughput of the Inception v3 topology scales efficiently.[18] The efficiency of scaling is best seen on the Plus configuration shown in Figure 15.

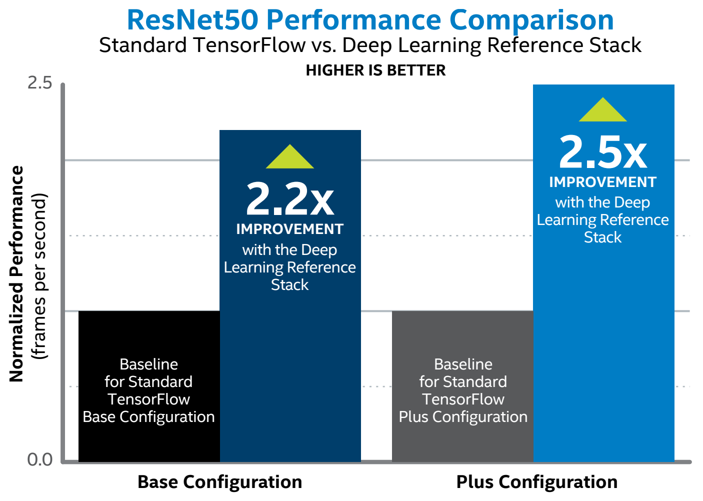


Figure 13. Using the Deep Learning Reference Stack version of TensorFlow more than doubled the performance of the ResNet50 v1.5 topology for both the Base and Plus configurations.

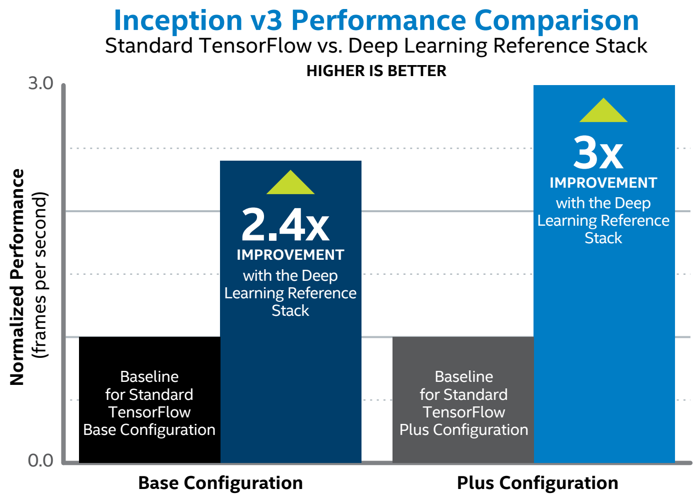


Figure 14. Using the Deep Learning Reference Stack version of TensorFlow more than doubled the performance of the Inception v3 topology for the Base configuration, and tripled the throughput of the Plus configuration.

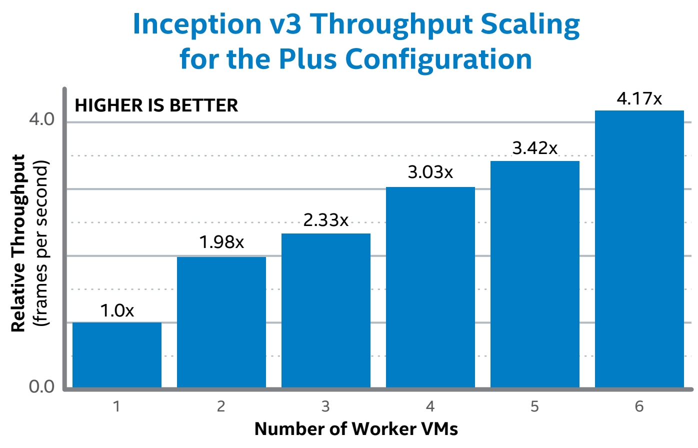


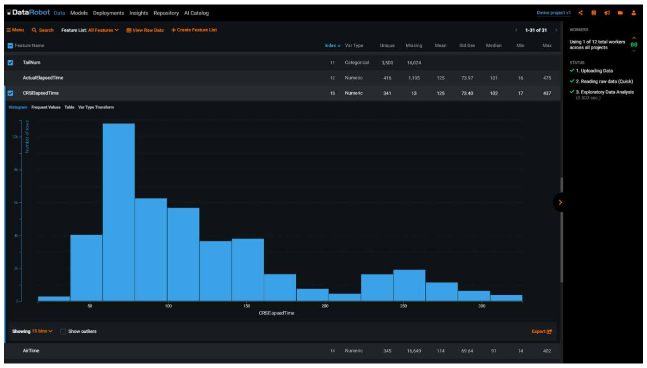
Figure 15. Inception v3 topology at fp32 precision throughput scaling using a Deep Learning Reference Stack container running from one to six workers in parallel for the Plus configuration.

Machine-Learning Model Training

The high-level goal of the workload example described here is to show how quick and easy it is to train a variety of simple models that allow prediction of important results. Advanced tools like DataRobot make the process more convenient than ever before.

1. **Data upload.** DataRobot allows choosing the JDBC source or uploading data from a URL, Apache Hadoop's Highly Distributed File System (HDFS), or locally stored files. The tool can handle `.csv`, `.tsv`, `.dsv`, `.xls`, `.xlsx`, `.sas7bdat`, `.parquet`, `.avro`, `.bz2`, `.gz`, `.zip`, `.tar`, and .tgz file formats. The choice is wide for a better user experience.

   The dataset chosen for demo purposes consists of flight data found here.
2. **Explore the AI catalog.** Uploaded datasets are available in the "AI catalog." Basic data analysis and dataset summaries can be found there, as well as basic information and a features list.
3. **Explore and visualize the data.** Convenient visualization is provided to better understand the data. This can help users choose important features for training. DataRobot automatically explores the dataset and identifies variable types and numerical statistics such as mean, median, and standard deviation. To see the details, click on the feature's name.

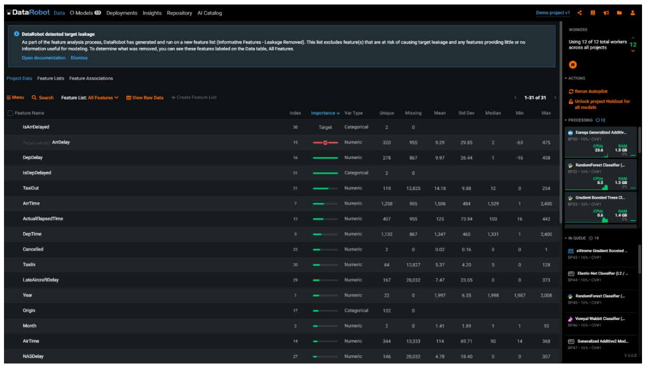


Feature Name	Index	Var Type	Unique	Missing	Mean	Std Dev	Median	Min	Max
TailNum	11	Categorical	3,500	16,024					
ActualElapsedTime	12	Numeric	416	1,195	125	73.97	101	16	475
CRSElapsedTime	13	Numeric	341	13	125	73.40	102	17	437
AirTime	14	Numeric	345	16,649	114	69.64	91	14	402

WORKERS
Using 1 of 12 total workers across all projects

STATUS
✓ 1. Uploading Data
✓ 2. Reading raw data (Quick)
✓ 3. Exploratory Data Analysis (0.803 sec.)

4. **Create a new project.** A new project can be created to start work on the data. DataRobot will automatically analyze the data and add suggestions. For example, DataRobot can create new fields based on existing features.
5. **Start the training process.** Choose the target feature by entering its name. The problem will be automatically identified based on the field type (for example, classification or regression). Click the Start button; the Autopilot lets DataRobot choose which training algorithms (blueprints) to check.
6. **Refine the model.** DataRobot's algorithms can again analyze data and check for redundancies or exclude one or more features that are at risk of causing target leakage and any features providing little or no information that is useful for modeling. The platform also decides which features are the most important for the result. The progress of these processes appears in the right column.



DataRobot detected target leakage
As part of the feature analysis process, DataRobot has generated and run on a new feature list (Informative Features - Leakage Removed). This list excludes feature(s) that are at risk of causing target leakage and any features providing little or no information useful for modeling. To determine what was removed, you can see these features labeled on the Data table, All Features.

Feature Name	Index	Importance	Var Type	Unique	Missing	Mean	Std Dev	Median	Min	Max
IsArrDelayed	30	Target	Categorical	2	0					
ArrDelay	15		Numeric	320	955	9.29	29.85	2	-63	475
DepDelay	16		Numeric	278	867	9.97	26.44	1	-16	458
IsDepDelayed	31		Categorical	2	0					
TaxiOut	21		Numeric	119	12,825	14.18	9.88	12	0	254
ArrTime	7		Numeric	1,258	955	1,506	484	1,529	1	2,400
ActualElapsedTime	12		Numeric	407	955	125	73.94	100	16	442
DepTime	5		Numeric	1,132	867	1,347	465	1,331	1	2,400
Cancelled	22		Numeric	2	0	0.02	0.16	0	0	1
TaxiIn	20		Numeric	64	12,827	5.37	4.20	5	0	128
LateAircraftDelay	29		Numeric	167	28,032	7.47	23.05	0	0	373
Year	1		Numeric	22	0	1,997	6.35	1,998	1,987	2,008
Origin	17		Categorical	132	0					
Month	2		Numeric	2	0	1.41	1.89	1	1	10
AirTime	14		Numeric	344	13,333	114	69.71	90	14	368
NASDelay	27		Numeric	146	28,032	4.78	18.40	0	0	307

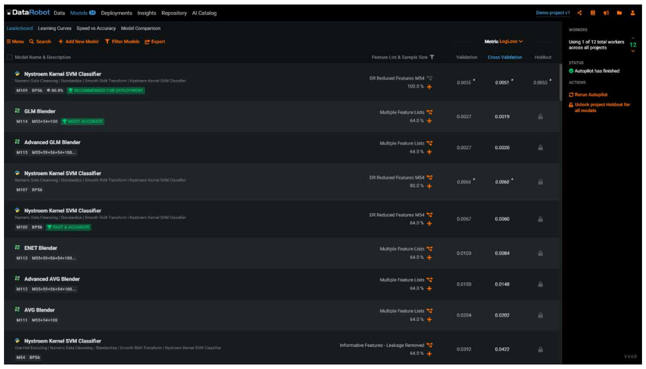
WORKERS
Using 12 of 12 total workers across all projects 12

ACTIONS
Rerun Autopilot
Unlock project Holdout for all models

PROCESSING 12
Eureqa Generalized Additiv...
BP30 • 16% • CV#1
CPUs 23.6 RAM 1.5 GB
RandomForest Classifier (...
BP32 • 16% • CV#1
CPUs 0.5 RAM 1.5 GB
Gradient Boosted Trees Cl...
BP33 • 16% • CV#1
CPUs 0.6 RAM 1.4 GB

IN QUEUE 19
eXtreme Gradient Boosted ...
BP43 • 16% • CV#1
Elastic-Net Classifier (L2 / ...
BP44 • 16% • CV#1
RandomForest Classifier (...
BP45 • 16% • CV#1
Vowpal Wabbit Classifier (...
BP46 • 16% • CV#1
Generalized Additive2 Mod...
BP47 • 16% • CV#1

7. **Compare algorithms.** DataRobot trains models using different algorithms and compares them for the best result. However, the platform does not train all models using the whole dataset. It can decide, based on results and performance for a part of the dataset, which algorithms are the most promising and proceed only with them. This approach saves time and resources. Users can compare algorithms on the leaderboard. Data is updated live during the training. When the process is completed, DataRobot shows its recommendations by labeling the best models, such as "Recommended for development," "Most accurate," and "Fast & Accurate" to help with algorithm choice.



Model Name & Description	Feature List & Sample Size	Validation	Cross Validation	Holdout
Nystroem Kernel SVM Classifier Numeric Data Cleansing | Standardize | Smooth Ridit Transform | Nystroem Kernel SVM Classifier M109 BP56 80.0% RECOMMENDED FOR DEPLOYMENT	DR Reduced Features M54 100.0 %	0.0055	0.0051	0.0055
GLM Blender M114 M55+54+100 MOST ACCURATE	Multiple Feature Lists 64.0 %	0.0027	0.0019	
Advanced GLM Blender M115 M55+59+56+54+100...	Multiple Feature Lists 64.0 %	0.0027	0.0020	
Nystroem Kernel SVM Classifier Numeric Data Cleansing | Standardize | Smooth Ridit Transform | Nystroem Kernel SVM Classifier M107 BP56	DR Reduced Features M54 80.0 %	0.0066	0.0060	
Nystroem Kernel SVM Classifier Numeric Data Cleansing | Standardize | Smooth Ridit Transform | Nystroem Kernel SVM Classifier M100 BP56 FAST & ACCURATE	DR Reduced Features M54 64.0 %	0.0067	0.0060	
ENET Blender M113 M55+59+56+54+100...	Multiple Feature Lists 64.0 %	0.0103	0.0084	
Advanced AVG Blender M112 M55+59+56+54+100...	Multiple Feature Lists 64.0 %	0.0150	0.0148	
AVG Blender M111 M55+54+100	Multiple Feature Lists 64.0 %	0.0204	0.0202	
Nystroem Kernel SVM Classifier One-Hot Encoding | Numeric Data Cleansing | Standardize | Smooth Ridit Transform | Nystroem Kernel SVM Classifier M54 BP56	Informative Features - Leakage Removed 64.0 %	0.0392	0.0422	

WORKERS
Using 1 of 12 total workers across all projects 12

STATUS
Autopilot has finished

ACTIONS
Rerun Autopilot
Unlock project Holdout for all models

8. **Deploy the model.** All models are available for download and manual deployment. The user can also deploy them automatically, which is a way to quickly start the prediction process. All data necessary to use the model after deployment can be found in the user interface.
9. **Start the prediction process.** DataRobot provides the REST API that can be used to make an inference. This allows users to use the model with a variety of programming languages. Also, instead of calling the API directly, users can take advantage of an auto-generated Python script. This code can be found in the user interface and can be adjusted as needed. It can be copied and become a part of a bigger system or simply remain as-is.
10. **Monitor the model's health.** Users can observe the model behavior in the user interface. The users can track the number of predictions made, requests, data error rate, and mode. Data drift tracking is also available.

Build Once, Run Anywhere

The machine-learning use case described in this document consists of the following elements:

- Model training using DataRobot
- Publication of the model (inference engine, the scorer) in an internal application catalog
- Deployment of the scorer

The model-training process is described in the previous section, "Machine-Learning Model Training." After finishing this step, the model binaries (scorer) either for Python or Java runtimes can be obtained.

A Docker image with those binaries is automatically built and published to a container registry—this example uses Harbor. As a Cloud Native Computing Foundation (incubating) project, Harbor delivers compliance, performance, and interoperability to help enterprises consistently and securely manage images across cloud-native compute platforms like Kubernetes and Docker. With the scorer (or with any other app) accessible in the registry, it can be deployed on the environment of choice, whether it's an on-premises cluster, private cloud, or public cloud.

User experience (both for business users and the operators of the platform) can be improved by publishing the scorer to an application catalog such as Kubeapps, so it can be deployed easily and quickly from a web user interface using just a few mouse clicks. Furthermore, the catalog can integrate with the container registry and use its webhooks to automatically detect if a new version of the application appears in the registry. In that case, the catalog can seamlessly redeploy the application, replacing the current version with the new one.

The Prediction Server feature in DataRobot v6.0.0 simplifies prediction model deployment. This feature enables the user to deploy a prediction service from the user interface (when the training is finished, the user can choose to deploy the selected model). Then the prediction can be obtained either by using an HTTP call or using a Python[19] or R[20] client.

Bill of Materials

Hardware

The reference architecture described here can scale from a single rack with just eight servers up to 15 domains (one management domain and 14 workload domains) with up to 400 ESXi servers total. This reference architecture uses 12 Intel® servers, two top-of-rack Arista switches, and a single Quanta Cloud Technology management switch in a single rack. Additional racks can be added when needed.

The initial software imaging requires an additional server or laptop running virtualization software and a privately managed switch. These components are not part of the reference architecture and are not needed after completing the VMware Cloud Foundation imaging and bring-up process.

To demonstrate support for heterogeneous hardware configurations, this reference architecture uses two types of servers differing in the CPU model, memory size, and number of drives. Enterprises can modify the base vSAN ReadyNode configuration to some extent, adding more memory or drives or replacing the CPU with a higher core-count or better clock-speed model. The general rules are described here.

In this reference architecture, each rack consists of a set of two top-of-rack switches and a single out-of-band management switch. In multirack deployments, an additional set of spine switches is recommended (usually installed in the second rack). With the introduction of VMware Cloud Foundation 3.0 and Bring You Own Network, VMware no longer certifies switch compatibility with VMware Cloud Foundation. For the networking components in this reference architecture, two models of network switches were used: two Arista DCS-7060CX2-32S-R for the data plane, and one Arista DCS-7010T-48-R for the management plane. Any network hardware with similar performance may be used instead.

Table 1 lists the hardware components for this reference architecture.

Table 1. Reference Architecture Hardware Components

	MANAGEMENT CLUSTER 4 NODES		BASE WORKLOAD DOMAIN 4 NODES		PLUS WORKLOAD DOMAIN 4 NODES	
	Part Description	**Qty** (per Node)	**Part Description**	**Qty** (per Node)	**Part Description**	**Qty** (per Node)
Base SKU	Intel® Server System VRN2208WFAF82R	1	Intel Server System VRN2208WFAF82R	1	Intel® Server System VRN2208WFAF83R	1
Mainboard	Intel® Server Board S2600WF0R	1	Intel Server Board S2600WF0R	1	Intel Server Board S2600WF0R	1
CPU	Intel® Xeon® Gold 6230 processor (20 cores, 2.1 GHz)	2	Intel® Xeon® Gold 6248 processor (20 cores, 2.5 GHz)	2	Intel® Xeon® Platinum 8268 processor (24 cores, 2.9 GHz)	2
Memory	32 GB RDIMM DDR4-2933	12	32 GB RDIMM DDR4-2933	12	32 GB RDIMM DDR4-2666	12
					128 GB Intel® Optane™ persistent memory module	12
Caching Tier	375 GB Intel® Optane™ SSD DC P4800X Series (PCIe x4 U.2)	2	375 GB Intel Optane SSD DC P4800X Series (PCIe x4 U.2)	2	375 GB Intel Optane SSD DC P4800X Series (PCIe x4 U.2)	2
Capacity Tier	2 TB Intel® SSD DC P4510 Series (2.5" NVMe U.2)	4	2 TB Intel SSD DC P4510 Series (2.5" NVMe U.2)	4	2 TB Intel SSD DC P4510 Series (2.5" NVMe U.2)	6
Boot Device	480 GB Intel® SSD DC S4510 Series (M.2, 80 mm)	1	480 GB Intel SSD DC S4510 Series (M.2, 80 mm)	1	480 GB Intel SSD DC S4510 Series (M.2, 80 mm)	1
NIC	Intel® Ethernet Converged Network Adapter X710-DA2	1	Intel Ethernet Converged Network Adapter X710-DA2	1	Intel Ethernet Converged Network Adapter X710-DA2	1

Software

This reference architecture consists of two main software component suites: VMware Cloud Foundation and TKGi. The latter requires multiple components and supporting infrastructure. In addition, several networking services like an enterprise Network Time Protocol (NTP) server and a Domain Name System (DNS) server are needed for seamless integration with the external networks and global time synchronization. Tables 2 and 3 provide software component information. For a complete list of requirements and prerequisites, refer to the official VMware documentation.

Table 2. VMware Cloud Foundation Products and Services

COMPONENT	VERSION	BUILD
VMware Cloud Foundation bundle	3.9	14866160
Cloud Builder VM	2.2.0.0	14866160
VMware ESXi hypervisor	ESXi 6.7 Update 3	15160138
VMware vSAN	6.7 Update 3	14263135
VMware NSX Data Center for vSphere	6.4.5	13282012
VMware NSX-T Data Center	2.5	14663974
VMware vCenter Server Appliance	vCenter Server 6.7 Update 3	14367737
VMware SDDC Manager	3.9	14866160
VMware vRealize Suite Lifecycle Manager	2.1 Patch 2	14062628
Tanzu Kubernetes Grid Integrated (TKGi)	1.5	14878150

Table 3. Other Software Components

COMPONENT	VERSION
Kubeapps	1.10.0
TensorFlow image	tensorflow/tensorflow:1.15.0-py3
Deep Learning Reference Stack distribution of TensorFlow image	clearlinux/stacks-dlrs-mkl:v0.5.0
Oracle Database	19.3
DataRobot	6.0.0
Microsoft SQL Server 2019	15.0.2070.41
Microsoft Windows Server 2019 Datacenter	17763.rs4.180914-1434
Windows HammerDB	3.3

Notes:

- The build version of the ESXi hypervisor is higher than in the official documentation. Additional security patches were applied to the ESXi systems before VMware Cloud Foundation deployment.
- Only the main components are listed in Table 2. For information on other components, see the VMware Cloud Foundation release notes.

BIOS and Firmware Components

From the hardware perspective, Table 4 lists the firmware and driver versions that were used in this solution.

Table 4. BIOS and Firmware Components

FIRMWARE/DRIVER NAME	VERSION
BIOS	02.01.0010
BMC	2.37
ME	04.01.04.339
SDR	1.98
NIC firmware	6.80 0x8003d05 1.2007.0
NIC version	1.9.5
Intel® Optane™ SSD DC P4800X	E2010435
Intel® SSD DC P4510	VDV10170
Intel® Optane™ persistent memory firmware	01.02.00.5417
CPU microcode	Base: 0x0500002c Plus: 0x0500002c Management: 0x0400002c

Deployment Blocks

The goal of using solutions like VMware Cloud Foundation, NSX-T, TKGi, and vSAN is to transform the legacy data center into an SDDC, where administrators can define, deploy, and manage clusters and resources based on actual demand from end users. Each of the mentioned components is a standalone product and may be used independently.

VMware Cloud Foundation

VMware Cloud Foundation is an integrated software platform that automates the deployment and lifecycle management of a complete SDDC on a standardized hyperconverged architecture. VMware Cloud Foundation consists of several core components (see Figure 16):

- VMware for compute virtualization
- VMware NSX for vSphere and NSX-T Data Center for network virtualization
- VMware vSAN for storage virtualization
- VMware vRealize Suite for cloud monitoring

VMware Cloud Foundation allows organizations to build enterprise-ready cloud infrastructure for the private and public cloud.

The standard architecture model for VMware Cloud Foundation includes a dedicated management domain (one per instance) for all management components and up to 14 virtual infrastructure workload domains created by the end user.

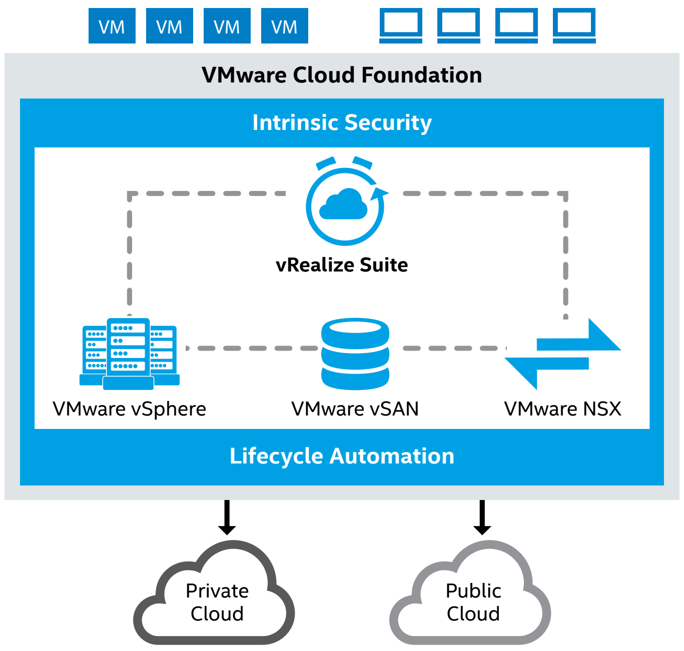


Figure 16. VMware Cloud Foundation environment.
Source: docs.vmware.com/en/VMware-Cloud-Foundation/3.9/com.vmware.vcf.ovdeploy.doc_39/GUID-7EBCC024-9604-4064-90A1-4851A78F7641.html

Management Domain

The management domain is a special-purpose workload domain that is used to host the infrastructure components needed to instantiate, manage, and monitor the VMware Cloud Foundation infrastructure. It is automatically created using the Cloud Builder on the first rack in the VMware Cloud Foundation system during bring-up. It contains management components such as SDDC Manager, vCenter Server, NSX-T Management Cluster, and vRealize Log Insight. The management domain uses vSAN as primary storage and requires a minimum of four nodes to work properly. When more racks are added to the system, the management domain automatically integrates those additional components.

Workload Domains

Workload domains are a logical grouping of private cloud capacity; they are provisioned automatically by SDDC Manager. Each workload domain is administered and patched independently and has its own compute, storage, and network resources to consume. All the tasks related to the workload domains are performed using the SDDC Manager web interface. This includes the creation, expansion, and deletion of workload domains, along with physical-infrastructure monitoring and management. A useful general FAQ for VMware Cloud Foundation is available here.

VMware vSAN

VMware vSAN is storage virtualization software—fully integrated with VMware vSphere—that joins all storage devices across a vSphere cluster into a shared data pool (see Figure 17). The use of vSAN eliminates the need for external shared storage.

Two vSAN cluster configurations are possible:

- **A hybrid vSAN cluster** uses two types of storage devices: flash devices for the cache tier and magnetic drives for the capacity tier.
- **An all-flash vSAN cluster** uses flash devices for both the cache and capacity tiers.

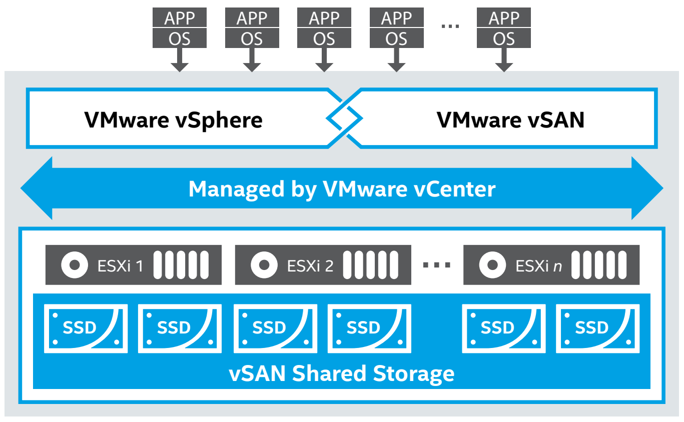


Figure 17. VMware vSAN overview.

VMware NSX

VMware NSX is a network virtualization solution that allows the creation of software-defined networks in virtualized data centers. Just as VMs are abstracted from physical server hardware, virtual networks (including switches, ports, routers, firewalls, and so on) are constructed in the virtual space. Virtual networks are provisioned and managed independent of the underlying hardware.

VMware offers two NSX platform types: NSX-V and NSX-T. NSX-V is designed for vSphere deployments only. It is the original NSX platform, has been available for several years, and is always tied to a single VMware vCenter Server instance. NSX-T is designed for many virtualization platforms and multihypervisor environments (examples include KVM, Docker, Kubernetes, and OpenStack, as well as AWS-native workloads). It can be deployed without a vCenter Server and is adapted for heterogeneous compute systems. It includes the NSX-T Container Networking interface (CNI) plug-in that allows configuration of network connectivity for container applications.

NSX Components

The main components of VMware NSX are NSX Manager, NSX Controllers, and NSX Edge gateways:

- **NSX Manager** is a centralized component of NSX that is used for network management. This virtual appliance provides the graphical user interface (GUI) and the RESTful APIs for creating, configuring, orchestrating, and monitoring NSX-T Data Center components. NSX Manager is the management plane for the NSX-T Data Center ecosystem.
- **NSX Controllers** are a distributed state-management system used to overlay transport tunnels and control virtual networks, which can be deployed as VMs on VMware ESXi or KVM hypervisors. The NSX Controller manages all logical switches within the network, and it handles information about VMs, hosts, switches, and Virtual Extensible LANs (VxLANs). Having three controller nodes ensures data redundancy in case of one NSX Controller node failure.
- **NSX Edge** is a gateway service that provides access to physical and virtual networks for VMs. It can be installed as a distributed virtual router or as a services gateway. The following services can be provided: dynamic routing, firewalls, network address translation (NAT), DHCP, virtual private networks (VPNs), load balancing, and high availability. NSX Edge can connect to two transport zones—one for overlay and the other for north-south peering with external devices (see Figure 18).

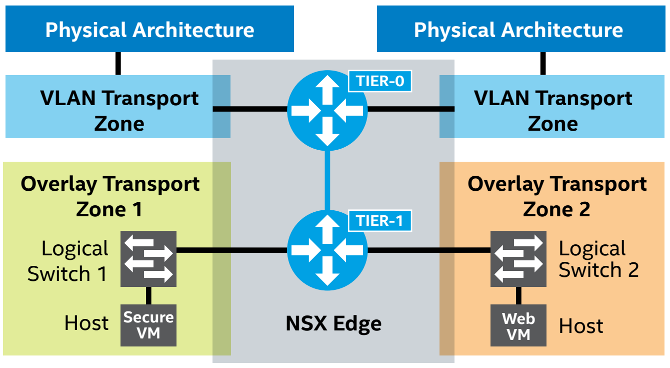


Figure 18. VLAN and Overlay Transport Zones using VMware NSX.
Source: docs.vmware.com/en/VMware-NSX-T-Data-Center/2.3/com.vmware.nsxt.install.doc/GUID-F47989B2-2B9D-4214-B3BA-5DDF66A1B0E6.html

These two transport zones define the limits of logical network distribution on the NSX Edge.

- **Overlay Transport Zone.** Any traffic that originates from a VM participating in an NSX-T Data Center domain might require reachability to external devices or networks. This is typically described as external north-south traffic. The NSX Edge node is responsible for decapsulating the overlay traffic received from compute nodes as well as encapsulating the traffic sent to compute nodes.
- **VLAN Transport Zone.** In addition to the encapsulate or decapsulate traffic function, NSX Edge nodes also need a VLAN transport zone to provide uplink connectivity to the physical infrastructure.

NSX-V requires the use of a vSphere VDS, as usual in vSphere. Standard virtual switches cannot be used for NSX-V. NSX-T presumes that you have deployed an NSX-T virtual distributed switch (N-VDS). Open vSwitches (OVS) are used for KVM hosts, while VMware vSwitches are used for ESXi hosts.

N-VDS is a software NSX component on the transport node that performs traffic transmission. N-VDS is the primary component of the transport node's data plane, which forwards traffic and owns at least one physical NIC. Each N-VDS of the different transport nodes is independent, but they can be grouped by assigning the same names for centralized management.

Transport zones are available for both NSX-V and NSX-T. Transport zones define the limits of logical network distribution. Each transport zone is linked to its N-VDS. Transport zones for NSX-T are not linked to clusters. There are two types of transport zones for VMware NSX-T due to GENEVE encapsulation: Overlay and VLAN. As for VMware NSX-V, a transport zone defines the distribution limits of VxLAN only.

Tanzu Kubernetes Grid Integrated

Tanzu Kubernetes Grid Integrated (TKGi) is a solution to operationalize Kubernetes for multicloud enterprises and service providers (see Figure 19). VMware Cloud Foundation enables automated deployment of TKGi on NSX-T workload domains. TKGi simplifies Kubernetes cluster deployment with Day 1 and Day 2 operations support and manages container deployment from the application layer all the way to the infrastructure layer. The deployed Kubernetes cluster is available in its native form—there are no add-ons or proprietary extensions. You can provision entire Kubernetes clusters using the command-line interface and can run container-based workloads on the clusters with Native Kubernetes CLI (kubectl). More in-depth documentation on TKGi can be found here.

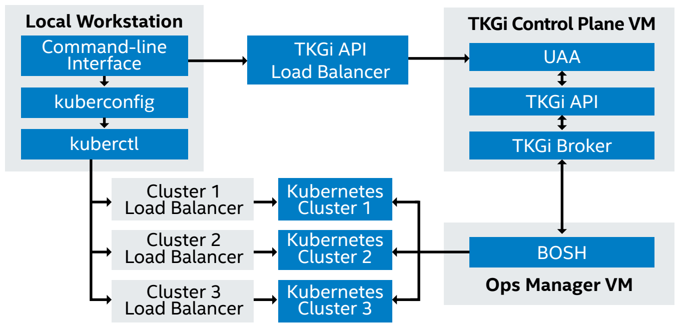


Figure 19. TKGi control plane.
Source: docs.pivotal.io/pks/1-6/control-plane.html

Integration of TKGi, VMware Cloud Foundation, NSX-T, and vSAN

VMware Cloud Foundation combines compute, storage, networking, security, and cloud management services—creating an ideal platform for running enterprise workloads and containerized applications. vSAN provides the flexibility to define policies on demand and delivers ease of management of storage for containers. Developers can consume storage as code by abstracting the complexity of the underlying storage infrastructure. With the help of NSX-T, the end users no longer have to know about the underlying network architecture. NSX-T can automatically create load balancers, routers, and switches to be used by TKGi (see Figure 20).

With the close integration of TKGi, NSX-T, and vSAN based on the VMware Cloud Foundation network for containers in Kubernetes clusters (see Figure 21), managing ephemeral and persistent storage as well as having vSAN's availability and data service features is available with ease. Additionally, vSphere High Availability and VMware vSphere Fault Tolerance can protect VMs from physical server failure. The combination of these technologies makes TKGi on VMware Cloud Foundation a complete solution, perfect for Kubernetes administrators and developers.

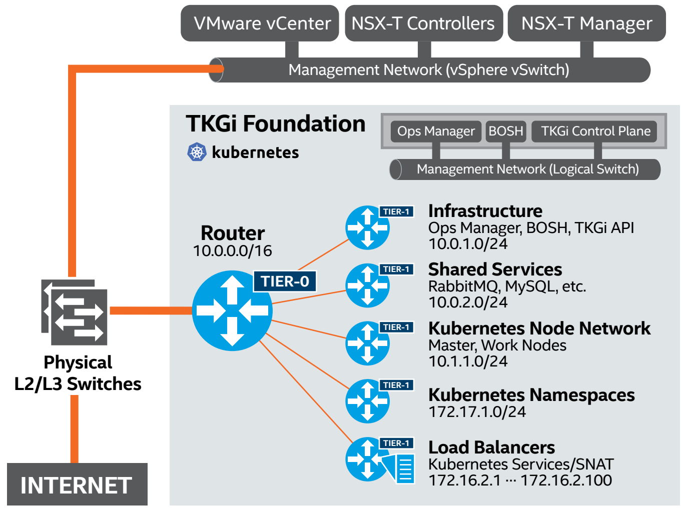


Figure 20. TKGi using NSX-T virtual network routers.
Source: docs.vmware.com/en/VMware-Enterprise-PKS/1.6/vmware-enterprise-pks-16/GUID-nsxt-multi-pks.html

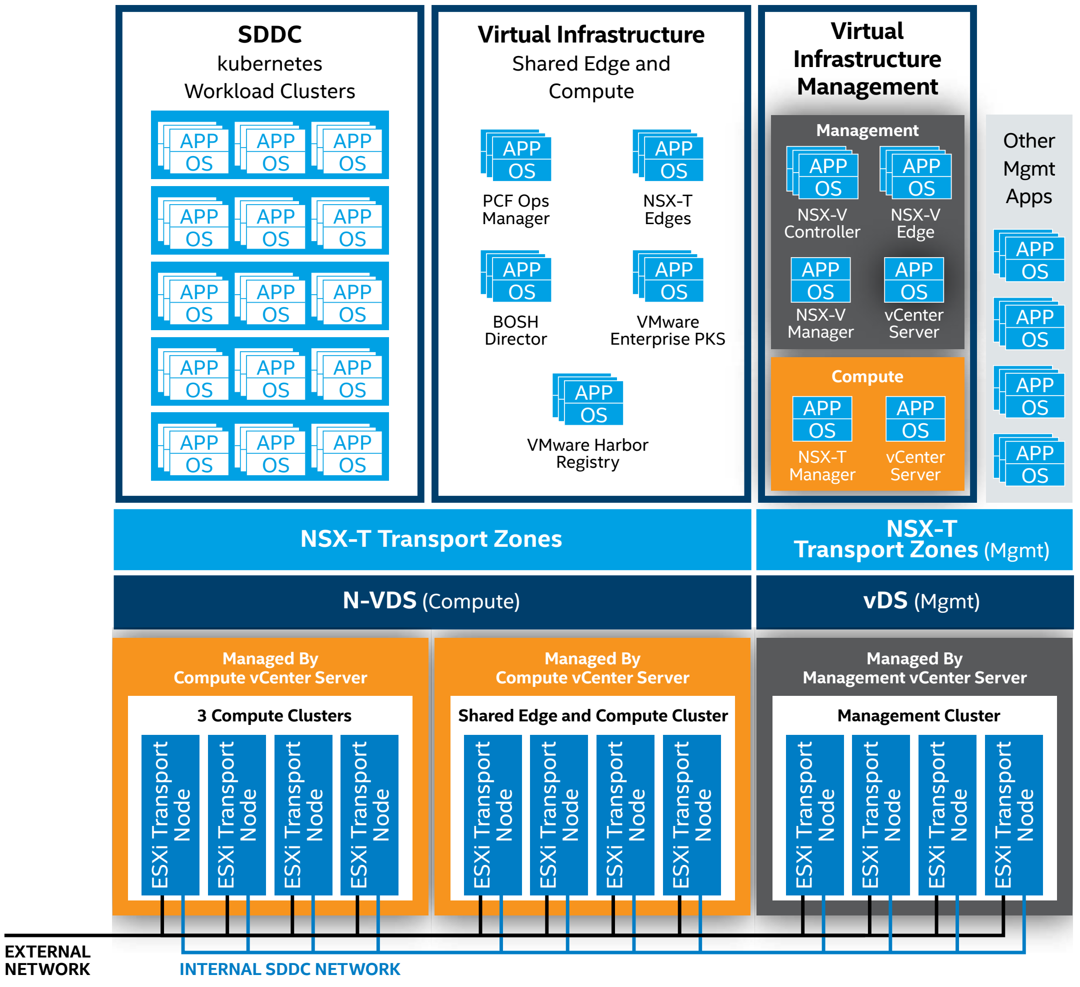


Figure 21. The ecosystem of VMware Cloud Foundation with TKGi with the use of VMware NSX-T.
Source: docs.vmware.com/en/VMware-Validated-Design/5.1/sddc-architecture-and-design-for-vmware-enterprise-pks-with-vmware-nsx-t-workload-domains/GUID-8F4D6F40-8126-4C41-952D-192A45AF5AF3.html

VMware Cloud on AWS

VMware Cloud on AWS is a complete VMware solution in the public cloud (see Figure 22). It is sold and operated as a service and consists of the same components as the on-premises environment—vSphere, vSAN, NSX-T, and vCenter Server—allowing rapid extension, migration, and protection of a regular VMware environment directly to the AWS public cloud, along with seamless integration for deployment of Kubernetes. With additional tools and add-ons (such as HCX and Hybrid Linked Mode), it provides methods of VM migration to and from the cloud. The service itself has two distinctive preconfigured regions: one for management and one for the customer. VMware is responsible for the management portion and customers control the operational portion. Users have very limited access to the management resources and settings but can manage workloads from the compute resource pool.

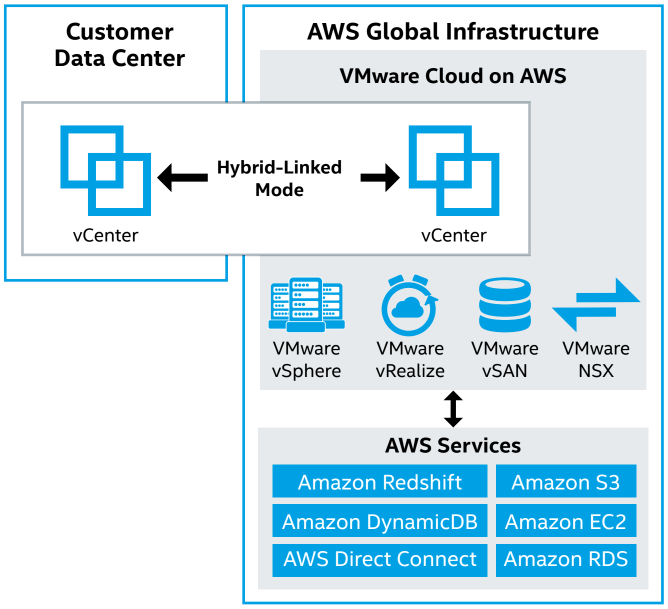


Figure 22. Components of VMware Cloud on AWS.
Source: youtube.com/watch?v=Ol7rNfkZT2c

On the network level, two gateways provide connectivity to VMware Cloud Foundation. The Management Gateway (MGW) enables users to connect to the management layer (vCenter, ESXi hosts, NSX, SRM, and HCX Managers), which uses a dedicated management subnet and restricted ports. The Compute Gateway (CGW) enables ingress and egress of workload VM network traffic traversing in and out of VMware Cloud. The Distributed Firewall feature allows for filtering of traffic between VMs on different segments within VMware Cloud. There are no restrictions on the CGW or Distributed Firewall, and users can configure firewall rules as they choose. The Management and Compute Gateways use separate VMware NSX Edges. During the deployment phase, the user can choose from two different Amazon EC2 Intel architecture-powered instances (i3.metal or r5.metal), integration into the AWS native environment, and connectivity options. Detailed instance information can be found here.

For more details regarding VMware Cloud on AWS, view the VMware Cloud on AWS Technical Deep Dive presentation.

Environment Configuration and Deployment

Installation and Configuration of Intel Optane Persistent Memory Module on VMware ESXi

To use Intel Optane persistent memory modules on an ESXi system, you must install and configure the modules on each server. The first step is installing them in the DIMM slots on the motherboard. Consult the platform manual for detailed placing rules and guidelines because several restrictions exist on what configurations are possible. One restriction is that the installed DIMM type and population configured to CPU1 must match CPU2. Find a detailed diagram for all possible population configurations in your platform manual.

Once the Intel Optane persistent memory modules are installed, choose between Memory Mode or App Direct Mode. Read Intel Optane Persistent Memory – BIOS settings for details. In the case of ESXi, the only requirement is to create a goal configuration for the region. If you intend to use App Direct Mode, set the Memory Mode [%] option to 0 (you still need to create the region). To benefit from additional memory with Memory Mode, change the Memory Mode [%] option to 100. You need to reboot the server each time you create a new goal configuration. Manual creation of namespaces is not necessary; ESXi will automatically create namespaces during boot if needed. After that, the system will be ready.

Important: If you want to change the mode from Memory Mode to App Direct Mode (or vice versa), you must delete the namespaces created by ESXi after making the change in the BIOS. Follow the instructions in Delete a Namespace in the VMware Host Client.

Preparing a VM for Intel Optane Persistent Memory

If you plan to use App Direct Mode, you need to perform additional steps when configuring any VM. The steps are similar for any guest OS on a VM; read the Enhance VMware VMs with Intel Optane Persistent Memory guide for a general understanding of what should be done. On Linux machines you will use the `ndctl` tool, while Windows Server 2019 has a built-in set of commands to achieve the same state (to configure DAX on Windows Server, see Understand and deploy persistent memory).

Important: The guest OS on the VM needs to support persistent memory if you want to use DAX mode.

If the guest OS does not support DAX, you may still use App Direct Mode on the VM, but only as a regular file system; however, you will still see performance improvements.

If you use Memory Mode, there are no additional steps required for VM configuration. The VMs will see the Intel Optane persistent memory as regular DRAM modules.

Environment Provisioning

As mentioned in the Solution Overview section, the complete environment consists of three main products: VMware Cloud Foundation, VMware NSX-T, and TKGi. The following sections describe how to provision these components.

Hardware and Software Requirements

Multiple requirements must be met before deploying the VMware Cloud Foundation platform. The complete details are listed in the VMware Cloud Foundation Planning and Preparation Guide.

There are specific requirements for the network (Jumbo Frames and 802.1Q tagging), network pools, VLANs and IP pools, hostnames, and IP addresses. You should be familiar with the aforementioned planning and preparation guide before taking the next steps. The entire setup relies heavily on multiple VLANs. Depending on your needs, each domain may have its own set of isolated VLANs for management, VMware vMotion, vSAN, VxLAN (NSX VTEP), and uplinks. Read VMware's VLAN IDs and IP Subnets documentation for more details.

Important: The maximum transmission unit (MTU) value for Jumbo Frames must be at least 1,700 bytes when using GENEVE encapsulation. This is because GENEVE headers have an additional metadata field of variable length. Without fulfilling this requirement, the overlay network will not work properly.

VMware Cloud Foundation Management Domain Deployment

The VMware Cloud Foundation deployment process includes several steps after you obtain all necessary hardware components, install them in a rack, and provide the necessary power, cooling, and uplink connection to the data center infrastructure. First, you deploy Cloud Builder VM, which is used to manage the initial Management Domain configuration process. It may also be used to install the ESXi OS on the nodes if you don't already have it installed; however, this step is optional. Then you need to download and complete the Deployment Parameter Sheet, and finally initiate the VMware Cloud Foundation start-up process.

The entire deployment of VMware Cloud Foundation is described in the VMware Cloud Foundation Architecture and Deployment Guide – VMware Cloud Foundation 3.9 document; see the Deploying Cloud Foundation chapter. Below are overviews of these four steps.

Step 1: Deploy Cloud Builder VM

Cloud Builder VM is used to deploy VMware Cloud Foundation; it also includes the VMware Imaging Appliance (VIA), which can be used for imaging the ESXi servers. You may also skip using VIA and install ESXi manually on all nodes. For details, read the Software Requirements chapter of the VMware Cloud Foundation Planning and Preparation Guide.

The detailed deployment procedure of Cloud Builder VM is available in VMware's Deploy Cloud Builder VM documentation.

Step 2: Install ESXi Software on VMware Cloud Foundation Servers

Imaging ESXi servers using Cloud Builder VM (which is done with VIA) is optional. If you already have servers with a supported version of ESXi, you do not need to use VIA. You may also install ESXi manually on each machine. Using VIA has advantages because it not only installs ESXi but also deploys any additional VIBs and configures standard passwords across all machines.

Detailed information on how to prepare hosts and set up VIA for ESXi installation is available in the Installing ESXi Software on Cloud Foundation Servers chapter of the Cloud Foundation Guide.

Be sure to also install any required or custom VIBs that your servers need. In most cases, those will be specific drivers for NICs or SSDs. If you decided to use VIA, you can include any required VIBs in the **Bundle ➔ Modify VIBs** menu so they will be automatically pre-installed with the ESXi. If you don't use VIA, you will need to manually install the VIBs. For this reference architecture, we added the following VIBs:

- **NIC driver and update tool for Intel® Network Adapter.** VMware ESXi 6.7 i40en 1.9.5 NIC Driver for Ethernet Controllers X710, XL710, XXV710, and X722 family available here.
- **Intel VMD driver for NVMe.** VMware ESXi 6.7 intel-nvme-vmd 1.8.0.1001 NVMe Driver for Intel available here.
- **Intel® SSD Data Center Tool for drive management.** Intel_SSD_DCT_3.0.24_ESXi available here.

Step 3: Download and Complete the Deployment Parameter Sheet for VMware Cloud Foundation

The Parameter Sheet is a spreadsheet file that is used to gather all the information regarding the environment. This separate file needs to be downloaded from vmware.com. After completing all the required fields (including a list of VLANs, network addresses, uplinks, passwords, and licenses) you import this file during the VMware Cloud Foundation bring-up process.

Full documentation for the Deployment Parameter Sheet is available in the VMware Cloud Foundation Architecture and Deployment Guide; see Download and Complete Deployment Parameter Sheet. The documentation describes in detail all the Parameter Sheet's tabs and fields.

Important: Be sure to provide passwords that match the password complexity criteria described in the Parameter Sheet. Failure to follow this rule will result in a failed deployment.

Step 4: VMware Cloud Foundation Bring-up

When you have ESXi installed on all management nodes, added all needed custom VIBs, and completed the Deployment Parameter Sheet, you can proceed to bring up VMware Cloud Foundation.

The bring-up process deploys SDDC Manager, vCenter Server for the management domain, NSX Managers and Controllers, Platform Services Controller (PSC), vSAN, and vRealize Log Insight (that is, the complete management domain of VMware Cloud Foundation). The process takes about two hours.

After the bring-up process is complete, you should see a notification with a link to the new SDDC Manager web interface. The SDDC Manager interface is accessible through a standard web browser.

The complete description of the VMware Cloud Foundation bring-up process is included in VMware's Initiate the Cloud Foundation Bring-Up Process documentation.

The management workload domain is now created and contains all the components necessary to manage the infrastructure. You should not deploy any user applications on this management cluster. Instead, create one or more workload domains that include one or more separate vSphere clusters with vSAN and NSX pre-installed and configured, along with an additional, dedicated instance of vCenter Server for each such domain. Instances of vCenter Server (one per each workload domain) and the NSX-T management cluster will be deployed on the management domain during subsequent steps.

Important: After you deploy the SDDC Manager and your management domain, you must configure dual authentication. This is required to perform particular tasks using SDDC Manager, such as password management for the entire platform and configuring NSX Manager backups. For details, read the VMware documentation, Configure Dual Authentication.

VMware Cloud Foundation Workload Domain Deployment

Step 1: Prerequisites for Workload Domain with NSX-T

Because this reference architecture uses NSX-T (not NSX-V) for networking, the NSX-T bundle must be downloaded to the SDDC Manager:

1. To browse and download bundles to the SDDC Manager, provide VMware Account credentials in the **Administration ➔ Repository Settings** section.
2. Search for the NSX-T Manager bundle (for this solution the exact bundle name is `NSX_T_MANAGER 2.5.0.0.0-14663974`) and download it using the **Repository ➔ Bundles** section.

   The NSX-T Manager bundle on the SDDC Manager is required, because when you create the first workload domain that is using NSX-T, the SDDC Manager will deploy a number of additional VMs on the management cluster, including a set of three NSX-T Manager VMs. All subsequent NSX-T workload domains will share this NSX-T Manager cluster. This step is necessary only when no other NSX-T workload domain exists.

You must also prepare a dedicated VLAN for VxLAN (Host VTEP) with DHCP enabled. Also, an MTU of at least 1,700 (9,000 is recommended) needs to be enabled for all VLANs and interfaces. All of the prerequisites are described in detail here.

For further edge VM deployment, you will also have to prepare additional VLAN for Edge VTEPs, as well as two more VLANs for uplinks. Routing must be possible between Host VTEP and Edge VTEP VLANs.

Step 2: Creating a Workload Domain

Creating a new workload domain on VMware Cloud Foundation is controlled and orchestrated by SDDC Manager, which installs and configures all the needed components (including vCenter Server, vSAN, and NSX-T). To deploy the new workload domain, the end user must use the VI Configuration wizard. If this will be the first workload domain in the environment that uses NSX-T, the wizard will require additional information (such as the DNS names for the NSX-T manager VMs), because it will create an NSX-T backplane in addition to the regular workload domain.

The complete procedure for creating a workload domain is available in the VMware Cloud Foundation Operations and Administration Guide document; see the Start the VI Configuration Wizard chapter.

The time needed to create a workload domain depends on the server configuration and the requested infrastructure size. In the example environment, the process took about one hour and 40 minutes to provision the complete infrastructure of a workload domain with four servers. This means that, depending on system and network configuration, the environment can be provisioned in as little as two hours instead of several days (which was how long it took before the automation provided by SDDC Manager in VMware Cloud Foundation was available).

Moreover, because the whole process is automated, there is a much lower risk of a misconfiguration that can often occur during manual installation. Such configuration errors could lead to serious issues or additional delays in infrastructure provisioning.

The end result of running the VI Configuration wizard will be a new cluster with a network fabric with the following configuration:

- Two transport zones (one for VLAN and one for overlay) for each host
- Network I/O Control (NIOC) and host uplink profiles
- Four logical switches (management, VSAN, vMotion, and overlay)
- Some empty VDS and portgroups

Important: Do not edit or delete any of these configurations.

Adding the NSX-T Edge Host to the Newly Created Workload Domain

When the workload domain deployment is finished, it's time to create and configure NSX-T Edge VMs. NSX Edge hosts are necessary to enable overlay virtual infrastructure and public networks for north-south traffic. NSX Edge hosts are not deployed automatically during workload domain creation.

Read the document Deploy and Configure NSX Edges in Cloud Foundation to deploy NSX Edge VMs in the workload domain. You must perform this procedure for each workload domain that you create. When executing the procedure, you may notice that some of the objects (like vlan-tz and overlay transport zones) have been created automatically during the first workload domain creation. Do not edit, delete, or re-create those objects. Create only the objects that are missing (such as Traditional Uplink Transport Zones, missing segments, IP pools, and so on).

Important: Aside from the VMware Cloud Foundation 3.9 documentation that we refer to here, the VMware Validated Design Documentation from VMware also provides guidance on how to set up NSX in a VMware environment. According to this latter document, there is an additional network segment that must be created for the NSX-T Edge VMs to work; however, this network segment is missing from the VMware Cloud Foundation 3.9 documentation (see full list of segments here). When you reach the "Create NSX-T Segments for System, Uplink and Overlay Traffic" section of the document, use the list of segments provided in Table 5 below instead of the table from the Cloud Foundation document.

Table 5. Complete List of Necessary Network Segments

SEGMENT NAME	TRANSPORT ZONE	VLAN	TYPE
nvds01-uplink01	sfo01-esxi-vlan	0-4094	Trunk
nvds01-uplink02	sfo01-esxi-vlan	0-4094	Trunk
sfo01-w-uplink01	sfo01-w-uplink01	1647	Access
sfo01-w-uplink02	sfo01-w-uplink02	1648	Access
sfo01-w-overlay	**sfo01-esxi-vlan**	**0-4094**	**Trunk**

The last row in Table 5 (highlighted in yellow) is missing from the VMware Cloud Foundation 3.9 documentation, but is then used in the same documentation in the next step (one of the Edge Appliance interfaces should be connected to this segment). Without this row, you cannot follow the documentation so make sure you add it. Similar to the missing segment, you must also create the overlay profile (it is also missing from the VMware Cloud Foundation 3.9 document). Consult the Create Uplink Profiles section of the Validated Design document (see the table in step 4 of this section; look for `sfo01-w-overlay-profile`) for the necessary overlay profile values.

You can deploy the Edge Appliances in one of two ways: Either follow the documentation and deploy using Open Virtualization Format (OVF) template, or deploy from the NSX-T Manager web user interface. When using the web user interface, you won't be able to attach the correct network segment to VM interfaces using the wizard—only networks from the management domain will be visible. Leave the default layout as it is and wait for the VM to appear in vCenter. Then change the networks by editing the VM settings to make all the necessary network segments visible.

Tanzu Kubernetes Grid Integrated Deployment

Now that you have a workload domain backed by NSX-T and NSX-T Edge hosts, you can install Tanzu Kubernetes Grid Integrated (TKGi).

The deployment is mostly automated. Follow the steps in the Deploy an Enterprise PKS Solution document to check the prerequisites and configure the additional subnets, hostnames, and IP addresses. Note that you must have existing DNS entries for all of the TKGi components (BOSH, Ops Manager, Harbor, and so on). Verify that those entries match the IP addresses that TKGi assigns to its VMs during the deployment. If they do not match, the deployment will fail.

You can deploy one TKGi instance per each workload domain. Having a TKGi deployment within a workload domain does not limit it in any way; it can still be used as a normal workload domain as long as the resources on the cluster are available.

VMware Cloud on AWS Configuration

This section describes the components and steps needed to bring up VMware Cloud on AWS and connect it to the on-premises environment.

Creation of SDDC Manager

The first step to bring up the cloud environment is to deploy an SDDC from the VMware Cloud console. This process is simple and requires selecting an AWS region where the service should be located and choosing deployment options (one or more hosts within the SDDC with the option of a stretched cluster), host type, and name of the SDDC. You also must connect the newly created SDDC to an AWS account (within 14 days). The entire process is explained in detail in the Deploy an SDDC from the VMC Console document. After finishing the entire process, you will have a complete vCenter environment with a compute cluster ready. Be sure to add the necessary firewall rules in the VMware Cloud Console SDDC settings after deployment; by default, all traffic is blocked and you won't be able to use your SDDC environment without changing these rules.

VPN Configuration

A dedicated connection is needed to access an SDDC securely from the on-premises services. Achieve this by establishing a VPN connection or by using AWS Direct Connect (DX). In this reference architecture, we configured a site-to-site IPsec VPN. The process is relatively easy to configure from the SDDC side, but detailed planning and pre-configuration are usually needed on the on-premises side. Each environment is different and will require additional configuration to prepare the tunnel endpoint, routing, and firewall rules. The type of the on-premises tunnel endpoint defines the exact settings that need to be set for the tunnel to be established, and both ends of the tunnel must match. In our case, we used a policy-based VPN, but depending on your personal needs and environment, you can use a route-based VPN and Border Gateway Protocol (BGP) instead. End users can also connect to VMware Cloud on AWS without a VPN, but it is less secure than having a VPN or DX in place as a prerequisite to using some of the more advanced features that come with hybrid cloud.

For step-by-step VPN configuration information, read the VMware Cloud on AWS: Connecting with VPN article and the Configure a VPN Connection Between Your SDDC and On-Premises Data Center Procedure on VMware Docs.

HCX Deployment

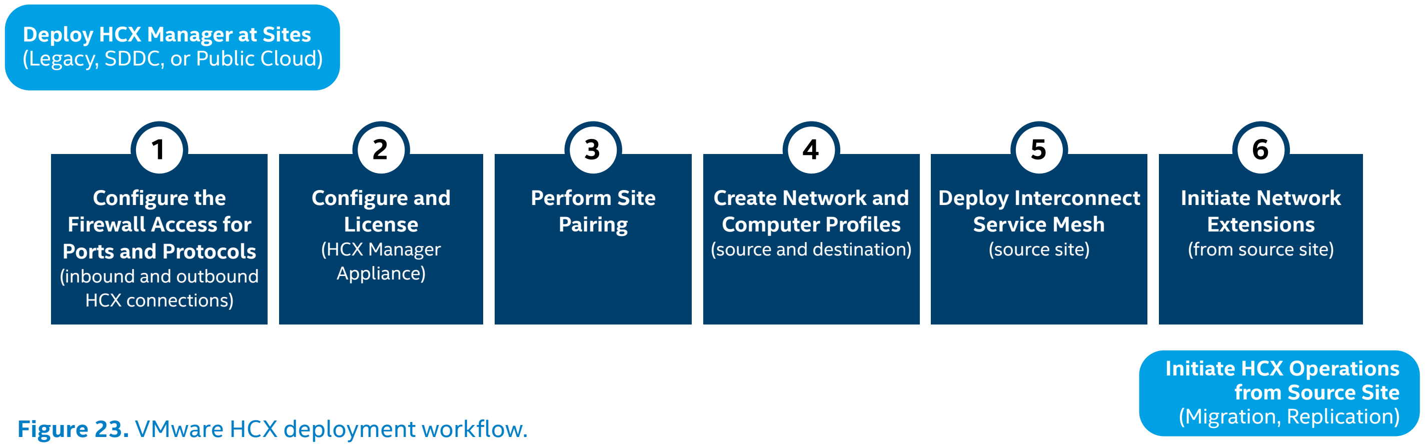
VMware HCX provides additional features and functionalities to the hybrid cloud environment. It provides site interconnection, enables vMotion and bulk migration of VMs, network extension (Layer 2 adjacent segments), business continuity and protection with disaster recovery, VM replication, and WAN optimization. It is a site-to-site architecture with a distinction of the source and destination environment (there is a specific installer for each). Typically, the HCX Cloud is deployed in the public cloud by the cloud provider and the HCX Connector is installed on-premises. Regardless of the type, the HCX is always deployed in the management zone, next to vCenter Server. The resulting HCX Manager is then used for administering VMware HCX (it is added as a new icon and menu option in the vCenter Server).

HCX consists of a virtual management machine at the source and destination sites with up to five types of interconnect service appliances (depending on the features selected and the HCX license). The services are enabled and deployed as virtual appliances at the source site with their peer appliances located at the destination site. The usual workflow of the deployment is presented in Figure 23. For a detailed description of deploying HCX along with Interconnect Service Mesh, consult the VMware HCX User Guide.

Hybrid Linked Mode Enablement

For ease of manageability, the Hybrid Linked Mode can be configured. It enables administration of both on-premises and cloud vCenter with single sign-on. It also centralizes the management into one place: an additional on-premises VM appliance from which the entire infrastructure is visible, as if connected to a single platform service controller. Detailed deployment information is available in Configuring Hybrid Linked Mode on VMware Docs.

Important: You must configure the VPN before the Hybrid Linked Mode because a requirement is that the cloud vCenter is reachable using its internal (non-public) cloud IP address, which is possible only when the VPN is configured.



Deploy HCX Manager at Sites (Legacy, SDDC, or Public Cloud)
1 Configure the Firewall Access for Ports and Protocols (inbound and outbound HCX connections)
2 Configure and License (HCX Manager Appliance)
3 Perform Site Pairing
4 Create Network and Computer Profiles (source and destination)
5 Deploy Interconnect Service Mesh (source site)
6 Initiate Network Extensions (from source site)
Initiate HCX Operations from Source Site (Migration, Replication)

Figure 23. VMware HCX deployment workflow.
Source: docs.vmware.com/en/VMware-HCX/services/hcx-user-guide.pdf

Summary

With the need for high-performance data analytics and AI on the rise in an increasingly digital world, enterprises seek flexible solutions that can run traditional data analytics and AI applications on-premises or in the public cloud. The VMware hybrid cloud platform combines the best of Intel hardware—including Intel Optane persistent memory—and VMware virtualization software. With this end-to-end solution that is ready to deploy, enterprises are poised to run both their traditional data analytics workloads and the AI and machine-learning workloads of the future.

At the same time, you must be aware that data warehousing workloads are latency- and I/O-sensitive, while analytics workloads are CPU- and memory-intensive. This reference architecture has been validated to meet expected key performance indicators in demanding customer workload scenarios. Our results include the following:

Data Warehousing

- The Plus configuration can deliver up to 3.34 times as many transactions per minute than the Base configuration.
- At least 98 percent of Microsoft SQL Server requests spend less than 5 ms in the CPU for both the Base and Plus configurations.
- The Plus configuration consistently meets an SLA value of 80 percent of requests spending less than 20 ms total time in the CPU, except in the most demanding scenarios. Even then, the drop is slight.

Analytics and AI

- We observed linear improvement in throughput with additional resources for AI inference workloads—proving excellent scalability.
- The use of the Deep Learning Reference Stack's optimized version of TensorFlow delivers between 2.2 to 3 times as many frames per second—on either the Base or Plus configuration—as the standard version of TensorFlow.

Find the solution that is right for your organization. Contact your Intel representative or visit Intel® Select Solutions for VMware vSAN.

Learn More

You may find the following resources helpful:

- Intel® Data Center Blocks for Cloud – vSAN ReadyNodes
- Intel® Optane™ persistent memory home page
- Intel Optane Persistent Memory – Virtualized Performance
- Deep Learning Reference Stack
- 2nd Generation Intel® Xeon® Scalable processors
- Intel® Select Solutions for VMware vSAN ReadyNodes
- Intel® Optane™ Solid State Drives
- Intel® Solid State Drives Data Center Family
- Intel® Distribution of OpenVINO™ toolkit
- Intel® Deep Learning Boost
- Intel® Framework Optimizations
- VMware vSAN
- VMware Cloud Foundation
- VMware Cloud on AWS

Appendix A: Solution Features Validation and Benchmarking

This section contains information on how to reproduce feature validation experiments, including installation of some required components and software procedures, configuration instructions, and benchmark execution scripts.

Configuring Oracle 19c to Use Intel® Optane™ Persistent Memory in Memory Mode

To use Intel® Optane™ persistent memory in Memory Mode with the Oracle Database, you need to configure PMem on VMware physical servers to use Memory Mode. Then when you create a VM for Oracle Database, you can choose a larger amount of memory available for the VM than was possible previously. You can assign 1.5 TB, 3 TB, or even up to 6 TB of memory (depending on the CPU model) for the VM.

Configuring Oracle 19c to Use Intel® Optane™ Persistent Memory in App Direct Mode[iii]

To use Intel Optane PMem in App Direct Mode with the Oracle Database, you must configure PMem on VMware physical servers to use App Direct Mode according to this documentation. This will configure PMem with the DAX support file system. Then you need to assign these disks as dedicated for Oracle Grid configuration. These disks will be used for +DATA and +REDO. Follow these steps:

1. Based on the following code, prepare the disks on your system.

```
[root@oracledb ~]# parted /dev/pmem0
GNU Parted 3.2
Using /dev/pmem0
Welcome to GNU Parted! Type ‘help’ to view a list of commands.
(parted) print
Error: /dev/pmem0: unrecognised disk label
Model: NVDIMM Device (pmem)
Disk /dev/pmem0: 889GB
Sector size (logical/physical): 512B/4096B
Partition Table: unknown
Disk Flags:
(parted) mklabel gpt
(parted) mkpart primary 2048s 100%
(parted) align-check opt 1
1 aligned
(parted) print
Model: NVDIMM Device (pmem)
Disk /dev/pmem0: 889GB
Sector size (logical/physical): 512B/4096B
Partition Table: gpt
Disk Flags:

Number Start End Size File system Name Flags
 1 1049kB 889GB 889GB primary

(parted) quit
Information: You may need to update /etc/fstab.
```

```
[root@oracledb ~]# parted /dev/pmem1
GNU Parted 3.2
Using /dev/pmem1
Welcome to GNU Parted! Type ‘help’ to view a list of commands.
(parted) print
Error: /dev/pmem1: unrecognised disk label
Model: NVDIMM Device (pmem)
Disk /dev/pmem1: 889GB
Sector size (logical/physical): 512B/4096B
Partition Table: unknown
Disk Flags:
(parted) mklabel gpt
(parted) mkpart primary 2048s 100%
(parted) align-check opt 1
1 aligned
(parted) print
Model: NVDIMM Device (pmem)
Disk /dev/pmem1: 889GB
Sector size (logical/physical): 512B/4096B
Partition Table: gpt
Disk Flags:

Number Start End Size File system Name Flags
 1 1049kB 889GB 889GB primary

(parted) quit
Information: You may need to update /etc/fstab.
```

2. After preparing the disks, change their ownership to your Oracle account and group with the following commands:

```
chown oracle:dba /dev/pmem0
chown oracle:dba /dev/pmem1
```

3. Now download Oracle archives with software. Use the following commands to download archives (change the URL to the one you obtain from the previous site):

```
curl -o /tmp/LINUX.X64_193000_grid_home.zip https://download.oracle.com/otn/linux/oracle19c/190000/LINUX.X64_193000_grid_home.zip
curl -o /tmp/LINUX.X64_193000_db_home.zip https://download.oracle.com/otn/linux/oracle19c/190000/LINUX.X64_193000_db_home.zip
```

4. Use the following commands to install Oracle Grid to use Automatic Storage Management (ASM) to manage the hard disk:

```
su - grid
cd /u01/app/grid/product/19.3/grid/
unzip -q /tmp/LINUX.X64_193000_grid_home.zip

su - root
cd /u01/app/grid/product/19.3/grid/cv/rpm
CVUQDISK_GRP=dba; export CVUQDISK_GRP
rpm -iv cvuqdisk-1.0.10-1.rpm

su - grid
cd /u01/app/grid/product/19.3/grid/
./gridSetup.sh
```

[iii] Note: All current versions of the Oracle Database as of this publication are susceptible to corruption in the configurations as outlined in the "My Oracle Support Note: File Systems and Devices on PMem in Database Servers May Cause Database Corruption" (Document ID 2608116.1). However, the issues described in this Support Note do not apply to PMem operating in Memory Mode. Click here for more information.

5. In the installation window of Oracle Grid, select **Configure Oracle Grid Infrastructure for Standalone Server** then click **Next** and follow these steps:
   a. In the next window, select one of your prepared hard disks, and change **Redundancy** to **External**. This disables redundancy but improves the efficiency of data writing. Note that this approach provides low security and should only be used for testing—it is not generally recommended.
   b. Click **Next** and then provide the password for the account in the next window. Choose **Oracle ASM Operator Group** for the DBA user group. The Installation Location and Create Inventory steps will read the ENVs.
   c. Click **Next**, follow the guidelines in the configuration wizard, and click **Install**.
   d. After the Oracle Grid installation finishes successfully, use the following command to check the configuration (you should see DATA disk group):

```
[grid@server ~ ]$ asmcmd
ASMCMD> ls
DATA/
ASMCMD> quit
```

6. Create a REDO disk group;run the following command to start the configuration wizard:

```
[grid@hammer-server ~]$ asmca
```

7. In the configuration wizard:
   a. Double-click **Disk Groups** to expand the list of disks groups. At the bottom of the window, click **Create** to add a REDO disk group.
   b. Select your prepared disk and set **Redundancy** to **External**. This disables redundancy, but improves the efficiency of data writing. Note that this approach provides low security and should only be used for testing—it is not generally recommended.
   c. Click **OK** to accept changes. You should see two disks groups: DATA and REDO.

The next step is to install the Oracle Database. Follow the instructions in the Oracle documentation.

Running the TensorFlow Benchmark with a Deep Learning Reference Stack Container (Inference Benchmarks Reproduction)

You first need to use Secure Shell (SSH) to connect to a node or VM in your VMware environment. Then follow these steps:

1. Install Docker and install Helm 3.
2. When Docker is ready, prepare a Dockerfile (`dlrs.Dockerfile`) with the following content:

```
FROM clearlinux/stacks-dlrs-mkl:v0.5.0

RUN mkdir /tf && cd /tf
RUN swupd clean && swupd bundle-add wget git devpkg-gperftools sysadmin-basic
RUN git clone --depth 1 https://github.com/tensorflow/models.git /tf/tf_models
RUN git clone -b v1.5.0 --depth 1 https://github.com/IntelAI/models.git /tf/intel-models
RUN wget -q -P /tf https://storage.googleapis.com/intel-optimized-tensorflow/models/v1_5/resnet50v1_5_int8_pretrained_model.pb
RUN wget -q -P /tf https://zenodo.org/record/2535873/files/resnet50_v1.pb
RUN wget -q -P /tf https://storage.googleapis.com/intel-optimized-tensorflow/models/v1_5/inceptionv3_int8_pretrained_model.pb
RUN wget -q -P /tf https://storage.googleapis.com/intel-optimized-tensorflow/models/v1_5/inceptionv3_fp32_pretrained_model.pb

WORKDIR /tf
CMD [“/bin/bash”]
```

3. Build the Docker image and push it to your Docker registry with the following commands (adjust the values to your environment):

```
docker build -f dlrs.Dockerfile -t stacks-dlrs-mkl:v0.5.0 .
docker login -u “${DOCKER_REPO_USER}” -p “${DOCKER_REPO_PASS}” “${DOCKER_REPO}”
docker tag stacks-dlrs-mkl:v0.5.0 ${DOCKER_REPO_PROJECT}/stacks-dlrs-mkl:v0.5.0
docker push ${DOCKER_REPO_PROJECT/stacks-dlrs-mkl:v0.5.0
```

4. When Helm 3 is ready, run the `helm create dlrs-benchmark` command to create a Helm 3 Chart for deployment of the TensorFlow benchmark with a Deep Learning Reference Stack container.

5. Inside the `dlrs-benchmark` directory, edit the following files and copy/paste the content:

   values.yaml:

```
image:
  repository: ${DOCKER_REPO_PROJECT/stacks-dlrs-mkl
  tag: v0.5.0

# How many jobs run in parallel on K8s
jobs: 5

# How many resources apply to pod in Guaranteed QoS class (requests==limits)
resources:
  cpu: «16»
  memory: «8Gi»
```

   Chart.yaml:

```
apiVersion: v2
name: dlrs-benchmark
description: A Helm chart for Kubernetes
type: application
version: 0.1.0
appVersion: 1.0.0
```

   templates/job.yaml:

```
apiVersion: batch/v1
kind: Job
metadata:
  labels:
    role: dlrs-benchmark
  name: dlrs-benchmark
spec:
  completions: {{ .Values.jobs }}
  parallelism: {{ .Values.jobs }}
  template:
    metadata:
      labels:
        role: dlrs-benchmark
      name: dlrs-benchmark
    spec:
      containers:
      - name: dlrs-benchmark
        image: {{ .Values.image.repository }}:{{ .Values.image.tag }}
        imagePullPolicy: Always
        resources:
          requests:
            cpu: {{ .Values.resources.cpu }}
            memory: {{ .Values.resources.memory }}
          limits:
            cpu: {{ .Values.resources.cpu }}
            memory: {{ .Values.resources.memory }}
        command:
        - «bash»
        args:
        - «/usr/local/bin/job.sh»
        volumeMounts:
        - name: usr-local-bin
          mountPath: «/usr/local/bin/job.sh»
          subPath: job.sh
      restartPolicy: Never
      volumes:
      - name: usr-local-bin
        configMap:
          name: dlrs-benchmark-job
```

templates/configmap.yaml:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: dlrs-benchmark-job
data:
  job.sh: |
    #!/bin/bash

    ########################################
    # Run benchmark.
    # Arguments:
    #   $1 Filepath to model
    #   $2 Model name
    #   $3 Precision of the model
    ########################################
    function run_benchmark() {
    for batch_size in 1 16 32 64 128
    do
        benchmark_app_output=»$(python3 /tf/intel-models/benchmarks/launch_benchmark.py --in-graph $1 --model-
name $2 --framework tensorflow --precision $3 --mode inference --batch-size ${batch_size} --benchmark-only 2>&1)»
        latency=»$(echo $benchmark_app_output | sed -n ‹s/.*Latency: \([0-9]*\.[0-9]*\) ms.*/\1/p›)»
        if [[ -z «${latency}» ]]; then
            latency=›-1›
        fi
        fps=»$(echo $benchmark_app_output | sed -n ‹s/.*[T,t]hroughput.*: \([0-9]*\.[0-9]*\) images\/sec.*/\1/p›)»
        if [[ -z «${fps}» ]]; then
            fps=›-1›
        fi
        echo «$2_$3,${batch_size},${fps},${latency}»
    done
    }

    echo «model_name,batch_size,fps,latency_ms»

    run_benchmark «/tf/resnet50v1_5_int8_pretrained_model.pb» «resnet50v1_5» «int8»
    run_benchmark «/tf/resnet50_v1.pb» «resnet50v1_5» «fp32»

    run_benchmark «/tf/inceptionv3_int8_pretrained_model.pb» «inceptionv3» «int8»
    run_benchmark «/tf/inceptionv3_fp32_pretrained_model.pb» «inceptionv3» «fp32»
```

6. When you have prepared the dlrs-benchmark Helm chart, run it on the Kubernetes cluster with the following command:

```
helm install --namespace default --name tf-benchmark tf-benchmark
```

7. After this command, the Kubernetes job is spawned on the cluster. To obtain the benchmark output, run:

```
kubectl get pods -l job-name=dlrs-benchmark -o name | xargs -n 1 kubectl logs
```

DataRobot Configuration in a VMware Environment

To install DataRobot on your environment, please contact DataRobot support for the best experience. They will provide all required files and instruction for a deployment best suited to your needs. In our case, we used VMs with the specifications in Table A1.

Table A1. Reference Architecture DataRobot Specifications

VM NAME	COUNT	vCPU CORES	RAM	STORAGE
Application & Data	1	8	64 GB	500 GB
Modeling Service	2	48	384 GB	500 GB
Prediction Server	1	4	32 GB	500 GB
Model Management	1	4	16 GB	500 GB
AI Catalog	1	4	32 GB	50 0GB

Kubeapps Usage

To use Kubeapps, you must have Helm 3 installed according to the installation documentation on your environment. Once it's installed, follow these steps:

1. Install Kubeapps according to their documentation on your Kubernetes cluster.
2. Create your Helm chart by running the `helm create [NAME_OF_YOUR_HELM_CHART]` command. Prepare all needed files for your chart. More information about creating Helm charts is available in the official Helm documentation.
3. With the Helm chart prepared, you can package it by running the `helm package [CHART_PATH]` command. This will create a .tgz file.
4. Log on to the Harbor registry and upload your Helm chart to the desired project. More information about managing Helm charts in Harbor is in this documentation.
5. Now you can add your Harbor Helm repository to Kubeapps:
   a. To add your private repository, go back to the Kubeapps user interface, navigate to **Configuration ➔ App Repositories,** and click on **Add App Repository**.
   b. Provide your Harbor registry address in the proper field and accept changes.
   c. Once you create the repository, click on the link of the repository you want to use, and deploy your own applications using Kubeapps.

Appendix B: Microsoft SQL on Windows Benchmark Configuration

Microsoft SQL VM on the Base Configuration

vCPU: 8
RAM: 44 GB
NIC: VMXNET 3
VMs: maximum 5 VMs per each ESXi node, 20 in total
Storage for VMs: vSAN

Microsoft SQL VM on the Plus Configuration

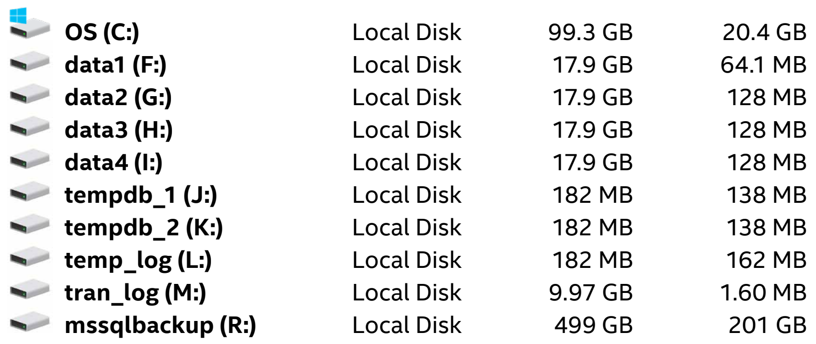
vCPU: 8
RAM: 44 GB
NIC: VMXNET 3
VMs: maximum 8 VMs per each ESXi node, 32 in total
Storage for VMs: vSAN
Storage for database: Intel® Optane™ persistent memory

Disk Layout

Figure B1 shows the disk layout for Microsoft SQL:

- 4x disks for data files (18 GB each)
- 2x disks for TempDB files (200 MB each)
- 1x disk for TempLog (100 MB)
- 1x disk for Transaction Log (10 GB).

Data disks on Intel Optane persistent memory should be formatted in NTFS and DAX (direct access). Logs on Intel Optane persistent memory should be formatted in NTFS and block access.



OS (C:)	Local Disk	99.3 GB	20.4 GB
data1 (F:)	Local Disk	17.9 GB	64.1 MB
data2 (G:)	Local Disk	17.9 GB	128 MB
data3 (H:)	Local Disk	17.9 GB	128 MB
data4 (I:)	Local Disk	17.9 GB	128 MB
tempdb_1 (J:)	Local Disk	182 MB	138 MB
tempdb_2 (K:)	Local Disk	182 MB	138 MB
temp_log (L:)	Local Disk	182 MB	162 MB
tran_log (M:)	Local Disk	9.97 GB	1.60 MB
mssqlbackup (R:)	Local Disk	499 GB	201 GB

Figure B1. An example of disk naming, layout, and size for the Microsoft SQL VM test machines.

Microsoft SQL Database Configuration

The database data is split into 16 files, placed on four disks (four files per disk).

Other options from T-SQL (Transact-SQL):

```
ALTER DATABASE [benchmarkedDB] SET DISABLE_BROKER WITH ROLLBACK
IMMEDIATE
 GO
 IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
 begin
EXEC [benchmarkedDB].[dbo].[sp_fulltext_database] @action =
'enable'
end
GO
ALTER DATABASE [benchmarkedDB] SET ANSI_NULL_DEFAULT OFF
GO
ALTER DATABASE [benchmarkedDB] SET ANSI_NULLS OFF
GO
ALTER DATABASE [benchmarkedDB] SET ANSI_PADDING OFF
GO
ALTER DATABASE [benchmarkedDB] SET ANSI_WARNINGS OFF
GO
ALTER DATABASE [benchmarkedDB] SET ARITHABORT OFF
GO
ALTER DATABASE [benchmarkedDB] SET AUTO_CLOSE OFF
GO
ALTER DATABASE [benchmarkedDB] SET AUTO_SHRINK OFF
GO
ALTER DATABASE [benchmarkedDB] SET AUTO_UPDATE_STATISTICS ON
GO
ALTER DATABASE [benchmarkedDB] SET CURSOR_CLOSE_ON_COMMIT OFF
GO
ALTER DATABASE [benchmarkedDB] SET CURSOR_DEFAULT GLOBAL
GO
ALTER DATABASE [benchmarkedDB] SET CONCAT_NULL_YIELDS_NULL OFF
GO
ALTER DATABASE [benchmarkedDB] SET NUMERIC_ROUNDABORT OFF
GO
ALTER DATABASE [benchmarkedDB] SET QUOTED_IDENTIFIER OFF
GO
ALTER DATABASE [benchmarkedDB] SET RECURSIVE_TRIGGERS OFF
GO
ALTER DATABASE [benchmarkedDB] SET AUTO_UPDATE_STATISTICS_ASYNC
OFF
GO
ALTER DATABASE [benchmarkedDB] SET DATE_CORRELATION_OPTIMIZATION
OFF
GO
ALTER DATABASE [benchmarkedDB] SET TRUSTWORTHY OFF
GO
ALTER DATABASE [benchmarkedDB] SET ALLOW_SNAPSHOT_ISOLATION OFF
GO
ALTER DATABASE [benchmarkedDB] SET PARAMETERIZATION SIMPLE
GO
ALTER DATABASE [benchmarkedDB] SET READ_COMMITTED_SNAPSHOT OFF
GO
ALTER DATABASE [benchmarkedDB] SET HONOR_BROKER_PRIORITY OFF
GO
ALTER DATABASE [benchmarkedDB] SET RECOVERY FULL
GO
ALTER DATABASE [benchmarkedDB] SET MULTI_USER
GO
ALTER DATABASE [benchmarkedDB] SET PAGE_VERIFY CHECKSUM
GO
ALTER DATABASE [benchmarkedDB] SET DB_CHAINING OFF
GO
ALTER DATABASE [benchmarkedDB] SET FILESTREAM( NON_TRANSACTED_
ACCESS = OFF )
GO
```

```
ALTER DATABASE [benchmarkedDB] SET TARGET_RECOVERY_TIME = 60
SECONDS
GO
ALTER DATABASE [benchmarkedDB] SET DELAYED_DURABILITY = DISABLED
GO
ALTER DATABASE [benchmarkedDB] SET QUERY_STORE = OFF
GO
USE [benchmarkedDB]
GO
ALTER DATABASE SCOPED CONFIGURATION SET MAXDOP = 0;
GO
ALTER DATABASE SCOPED CONFIGURATION FOR SECONDARY SET MAXDOP =
PRIMARY;
GO
ALTER DATABASE SCOPED CONFIGURATION SET LEGACY_CARDINALITY_
ESTIMATION = OFF;
GO
ALTER DATABASE SCOPED CONFIGURATION FOR SECONDARY SET LEGACY_
CARDINALITY_ESTIMATION = PRIMARY;
GO
ALTER DATABASE SCOPED CONFIGURATION SET PARAMETER_SNIFFING = ON;
GO
ALTER DATABASE SCOPED CONFIGURATION FOR SECONDARY SET PARAMETER_
SNIFFING = PRIMARY;
GO
ALTER DATABASE SCOPED CONFIGURATION SET QUERY_OPTIMIZER_HOTFIXES =
OFF;
GO
ALTER DATABASE SCOPED CONFIGURATION FOR SECONDARY SET QUERY_
OPTIMIZER_HOTFIXES = PRIMARY;
GO
ALTER DATABASE [benchmarkedDB] SET READ_WRITE
GO
USE [benchmarkedDB]
GO
IF NOT EXISTS (SELECT name FROM sys.filegroups WHERE is_default=1
AND name = N'SECONDARY') ALTER DATABASE [benchmarkedDB] MODIFY
FILEGROUP [SECONDARY] DEFAULT
GO
USE [benchmarkedDB]
GO
declare @autogrow bit
SELECT @autogrow=convert(bit, is_autogrow_all_files) FROM sys.
filegroups WHERE name=N'SECONDARY'
if(@autogrow=0)
 ALTER DATABASE [benchmarkedDB] MODIFY FILEGROUP [SECONDARY]
AUTOGROW_ALL_FILES
GO
USE [master]
GO
declare @autogrow bit
SELECT @autogrow=convert(bit, is_autogrow_all_files) FROM sys.
filegroups WHERE name=N'SECONDARY'
if(@autogrow=0)
 ALTER DATABASE [benchmarkedDB] MODIFY FILEGROUP [SECONDARY]
AUTOGROW_ALL_FILES
GO
-- disable autogrow on tranlog
ALTER DATABASE [benchmarkedDB]
MODIFY FILE (NAME = N'benchmarkedDB_log', FILENAME = 'M:\vdap_log.
ldf', SIZE = 10000MB, FILEGROWTH = 0 );
GO
```

Microsoft SQL Configuration

As TSQL:

```
sp_configure ‘show advanced options’, 1;
GO
RECONFIGURE WITH OVERRIDE;
GO
sp_configure ‘max server memory’, 50000;
GO
RECONFIGURE WITH OVERRIDE;
GO
sp_configure ‘min server memory’, 40000;
GO
RECONFIGURE WITH OVERRIDE;
GO
sp_configure ‘max worker threads’, 3000;
GO
RECONFIGURE WITH OVERRIDE;
GO
sp_configure ‘recovery interval’, 32767;
GO
RECONFIGURE WITH OVERRIDE;
GO
sp_configure ‘lightweight pooling’, 1;
GO
RECONFIGURE WITH OVERRIDE;
GO
sp_configure ‘priority boost’, 1;
GO
RECONFIGURE WITH OVERRIDE;
GO
EXEC sys.sp_configure N’network packet size (B)’, N’8192’
GO
RECONFIGURE WITH OVERRIDE
GO
ALTER SERVER CONFIGURATION SET MEMORY_OPTIMIZED HYBRID_BUFFER_POOL = ON;
GO
ALTER DATABASE [benchmarkedDB] SET MEMORY_OPTIMIZED = ON;
GO
```

HammerDB Configuration for Load Generation

- Windows SQL Server 2019 15.0.2070.41 with patches up to 2020-03-01
- Windows Server 2019 Datacenter 17763.rs5_release.180914-1434 with patches up to 2020-03-01
- NIC: VMXNET3
- Storage: vSAN
- Network: VLAN-based (no NSX-T overlay segments)
- Software used for benchmark: HammerDB 3.3 Windows
- SQL Server ODBC Driver: ODBC Driver 17 for SQL Server
- TPC-C driver script: Timed Driver Script
- Total transactions per user: 1,000,000
- Minutes of ramp-up time: 10
- Minutes for test duration: 20
- Use all warehouses: Yes
- 750 warehouses and 75 users per VM
- 1 HammerDB VM per 1 Microsoft SQL VM
- Data collected: transactions per minute (TPM) summed for all HammerDBs (all Microsoft SQL instances), CPU Time: Requests (5 ms), CPU Time: Total (20 ms) from all Microsoft SQL Instances

Endnotes

[1] Testing by Intel, June 15-23, 2020. Each cluster (Base, Plus, and the Management cluster that was used for workload generation) consists of four machines. All machines are identical within the given cluster. On top of those clusters the VMs are placed: five VMs per node for Base and eight VMs per node for Plus. We had four scenarios for each environment (Base and Plus). The first scenario used only one node in the cluster, the second scenario used two, and so on, up until all four nodes were used. Each scenario was repeated three times; the results given here are the average values for these three runs for each scenario.

Base 4-node cluster configuration: Intel® Xeon® 6248 processor (2.5 GHz, 20 cores); 384 GB DRAM (12x 32 GB 2933MHz); Intel® Hyper-Threading Technology ON; Intel® Turbo Boost Technology ON; Storage: VMware vSAN, disk group: 1x Intel® Optane™ SSD DC P4800X 375 GB and 2x Intel® SSD DC P4510 2 TB, two disk groups per node; BIOS = 02.01.0010; microcode = 0x0500002c; OS = VMware ESXi 6.7.0 15160138; 1x Intel® Ethernet Adapter XXV710-DA2.

Base VM configuration: 8 vCPUs; 44 GB vMemory; vDisks: 100 GB – OS drive (LSI controller), 4x 18 GB – Data DB (1 SCSI controller), 4 files per disk, 2x 200 MB – Temp DB (1 SCSI controller), 200 MB – Temp log (same SCSI controller as User transaction log), 10 GB – User transaction log (1 SCSI controller); vNetwork = VMXNET3; OS = Windows Server 2019 Datacenter 17763.rs5.180914-1434; Database = MSSQL Server 2019 15.0.2070.41.

Plus 4-node cluster configuration: Intel® Xeon® 8268 processor (2.9 GHz, 24 cores); 384 GB DRAM (12x 32 GB 2933MHz) plus 1,536 GB Intel® Optane™ persistent memory (12x 128 GB); Intel Hyper-Threading Technology ON; Intel Turbo Boost Technology ON; Storage: VMware vSAN, disk group: 1x Intel Optane SSD DC P4800X 375 GB and 2x Intel SSD DC P4510 2 TB, two disk groups per node; BIOS = 02.01.0010; microcode = 0x0500002c; OS = VMware ESXi 6.7.0 15160138; 1x Intel Ethernet Adapter XXV710-DA2; PMem used a 2-2-2 configuration in App Direct Mode; PMem firmware version = 01.02.00.5417.

Plus VM configuration: 8 vCPUs; 44 GB vMemory; vDisks: 100 GB – OS drive (LSI controller), 4x 18 GB Data DB (NVDIMM), 4 files per disk, 2x 200 MB Temp DB (NVDIMM non-DAX volume), 200 MB Temp log (NVDIMM non-DAX volume), 10 GB User Transaction log (NVDIMM non-DAX volume); vNetwork = VMXNET3; OS = Windows Server 2019 Datacenter 17763.rs5.180914-1434; Database = MSSQL Server 2019 15.0.2070.41.

[2] This result was reported by an independent test/benchmark published at glesys.se/blogg/benchmarking-intel-optane-dc-persistent-memory (Table 3, measured latency in microseconds with 512-byte block size).

[3] Intel. "Product Brief: Intel Optane SSD DC P4800X/P4801X Series," intel.com/content/www/us/en/products/docs/memory-storage/solid-state-drives/data-center-ssds/optane-ssd-dc-p4800x-p4801x-brief.html.

[4] Intel. "Product Brief: Intel Optane SSD DC P4800X Series," intel.com/content/www/us/en/solid-state-drives/optane-ssd-dc-p4800x-brief.html. Based on internal Intel testing.

[5] Intel® Ethernet 700 Series includes extensively tested network adapters, accessories (optics and cables), hardware, and software along with broad operating system support. A full list of the product portfolio's solutions is available at intel.com/ethernet. Hardware and software are thoroughly validated across Intel® Xeon® Scalable processors and the networking ecosystem. The products are optimized for Intel® architecture and a broad operating system ecosystem: Windows, Linux kernel, FreeBSD, Red Hat Enterprise Linux (RHEL), SUSE, Ubuntu, Oracle Solaris, VMware ESXi.

[6] Intel Ethernet 700 Series network adapters are backed with global support infrastructure for customers pre- and post-sales.

[7] Intel Ethernet 700 Series network adapters' supported connections and media types are direct-attach copper and fiber SR/LR (QSFP+, SFP+, SFP28, XLPPI/CR4, 25G-CA/25G-SR/25G-LR), twisted-pair copper (1000BASE-T/10GBASE-T), and backplane (XLAUI/XAUI/SFI/KR/KR4/KX/SGMII). Note that Intel is the only vendor offering the QSFP+ media type.

[8] Intel Ethernet 700 Series network adapters' supported speeds include 1 GbE, 10 GbE, 25 GbE, and 40 GbE.

[9] docs.vmware.com/en/VMware-Cloud-on-AWS/services/com.vmware.vmc-aws.getting-started/GUID-EF198D55-03E3-44D1-AC48-6E2ABA31FF02.html. Intel deployed VMware Cloud on AWS four times and can confirm that deployment took about two hours each time, based on anecdotal experience.

[10] software.intel.com/content/www/us/en/develop/tools/distribution-for-python/benchmarks.html.

[11] To learn more, see builders.intel.com/ai/blog/1-billion-models-datarobot-intel.

[12] See endnote 1.

[13] Testing by Intel, June 15-23, 2020. See endnote 1 for physical infrastructure configuration information.

[14] Testing by Intel, June 15-23, 2020. See endnote 1 for physical infrastructure configuration information.

[15] Intel. "Technology Brief: Intel Optane Technology: Memory or Storage? Both," intel.com/content/dam/www/public/us/en/documents/technology-briefs/what-is-optane-technology-brief.pdf.

[16] Testing by Intel, June 15-23, 2020. Workload: ResNet 50 v1.5 topology at fp32 precision. Default TensorFlow tensorflow/tensorflow:1.15.0-py3; optimized TensorFlow: clearlinux/stacks-dlrs-mkl:v0.5.0.

Base 4-node cluster configuration: Intel Xeon 6248 processor (2.5 GHz, 20 cores); 384 GB DRAM (12x 32 GB 2933MHz); Intel Hyper-Threading Technology ON; Intel Turbo Boost Technology ON; Storage: VMware vSAN, disk group: 1x Intel Optane SSD DC P4800X 375 GB and 2x Intel SSD DC P4510 2 TB, two disk groups per node; BIOS = 02.01.0010; microcode = 0x0500002c; OS = VMware ESXi 6.7.0 15160138; 1x Intel Ethernet Adapter XXV710-DA2.

Base Fat VM configuration: 80 vCPUs; OS = CentOS Linux release 8.1.1911; kernel = 4.18.0-147.5.1.el8_1.x86_64; other SW: VMware vSphere 6.7, VMware Cloud Foundation 3.9, VMware ESXi hypervisor 6.7 update 3.

Normalized Base Results: 2.2X more throughput using the Deep Learning Reference Stack version of TensorFlow, compared to the default TensorFlow performance.

Plus 4-node cluster configuration: Intel Xeon 8268 processor (2.9 GHz, 24 cores); 384 GB DRAM (12x 32 GB 2933MHz) plus 1,536 GB Intel Optane persistent memory (12x 128 GB); Intel Hyper-Threading Technology ON; Intel Turbo Boost Technology ON; Storage: VMware vSAN, disk group: 1x Intel Optane SSD DC P4800X 375 GB and 2x Intel SSD DC P4510 2 TB, two disk groups per node; BIOS = 02.01.0010; microcode = 0x0500002c; OS = VMware ESXi 6.7.0 15160138; 1x Intel Ethernet Adapter XXV710-DA2; PMem used a 2-2-2 configuration in App Direct Mode; PMem firmware version = 01.02.00.5417.

Plus Fat VM configuration: 96 vCPUs; otherwise, identical to the Base Fat VM configuration.

Normalized Plus Results: 2.5X more throughput using the Deep Learning Reference Stack version of TensorFlow, compared to the default TensorFlow performance.

[17] Testing by Intel, June 15-23, 2020. Workload: Inception V3 topology at fp32 precision. See endnote 16 for physical infrastructure configuration and fat VM configurations.

Normalized Base Results: 2.4X more throughput using the Deep Learning Reference Stack version of TensorFlow, compared to the default TensorFlow performance.

Normalized Plus Results: 3X more throughput using the Deep Learning Reference Stack version of TensorFlow, compared to the default TensorFlow performance.

[18] Testing by Intel, June 15-23, 2020. Workload: Inception V3 topology at fp32 precision. 6 worker VMs with 16 vCPU each. See endnote 16 for physical infrastructure configuration. Default TensorFlow tensorflow/tensorflow:1.15.0-py3; optimized TensorFlow: clearlinux/stacks-dlrs-mkl:v0.5.0.

Worker Node configuration: OS = Ubuntu 16.04.6 LTS; kernel = 4.15.0-55-generic; other SW: VMware vSphere 6.7, VMware Cloud Foundation 3.9, VMware ESXi hypervisor 6.7 update 3.

Normalized Results:

NUMBER OF WORKER VMS	THROUGHPUT IMPROVEMENT
1	1X
2	1.98X
3	2.33X
4	3.03X
5	3.42X
6	4.17X

[19] To learn more, see https://pypi.org/project/datarobot.

[20] To learn more, see https://cran.r-project.org/web/packages/datarobot/index.html.

Solution Provided By: