WHITE PAPER - **OCTOBER 2018**

RUNNING HPC AND MACHINE LEARNING WORKLOADS ON VMWARE VSPHERE

Best Practices Guide

October 2018

vmware®

Index

1. INTRODUCTION

High-performance computing (HPC) is the use of parallel-processing techniques for solving complex computational problems, including deep learning (DL) and other machine learning (ML) techniques. This white paper provides best practices for running these HPC workloads on VMware vSphere®.

By default, VMware vSphere ESXi is optimized for driving efficient CPU, memory, networking, and storage performance for a wide range of workloads. Performance Best Practices for VMware vSphere offers comprehensive performance-tuning guidance for the most performance-critical areas of VMware vSphere. However, HPC workloads often have much more demanding resource requirements than those workloads found in the typical enterprise. Special considerations are required to meet these extreme performance demands. This document is intended to provide supplementary best practices for IT administrators who manage VMware vSphere environments hosting HPC workloads.

The resulting benefits and outcomes of each configuration choice depend on the characteristics of the specific HPC application in use. VMware recommends experimenting with the available options prior to deployment.

For more information about HPC workload characteristics and the virtualization of HPC environments, see the companion paper Virtualizing High-Performance Computing (HPC) Environments Reference Architecture.

2. BIOS SETTINGS

When configuring the BIOS of a machine hosting HPC workloads, it's important to ensure the latest version of BIOS is being used and that virtualization support (e.g., Intel VT) and turbo (e.g., Intel Turbo Boost) are enabled. Additional considerations are discussed below.

2.1 Power Management

There are two levels of power management available when running VMware ESXi™:

1. BIOS-level power management, which determines whether power management should be handled by the BIOS, or whether that control should be ceded to ESXi
2. ESXi settings for power management, which determine what power policy to use when the BIOS has given control of power management to ESXi

Given the flexible host power management (HPM) capabilities of ESXi, configuring BIOS settings to allow ESXi to control power management is recommended. Steps for enabling ESXi power management control vary by server model.

For example, on Dell EMC servers, set the **system profile** to **performance per watt (OS)** to enable ESX HPM. On HP servers, set the **HP power profile** to **custom** and **HP power regulator** to **OS controlled**.

After setting the power management profile in the BIOS, ESXi can be customized to suit specific workload environments. ESXi has four power policies:

- **High Performance:** Uses no power management features
- **Balanced:** Reduces energy consumption with minimal negative impact to performance
- **Low Power:** Reduces energy consumption at the risk of negative impact to performance
- **Custom:** User-defined power management policy

For throughput workloads running on a partially loaded system (not all processor cores are busy all the time), the recommended ESXi HPM power policy is **balanced**. While throughput workloads are computationally intensive, using the **High-Performance** power policy would prevent the system from entering C/C1E states, reducing the benefits of Turbo boost.

In cases in which all of a host's cores are busy running user applications, **Balanced** and **High-Performance** settings should have the same performance effect, although there is a notable exception.

If the workload is extremely latency-sensitive (e.g., many MPI applications and also financial workloads requiring interconnect latencies ranging from microseconds to a few tens of microseconds), any form of power management adds latency that may be unacceptable. In this case, setting the host's BIOS power management to **maximum performance**, disabling all power management, is recommended over the use of ESXi power management. Users of Dell servers should set the **system profile** to **performance**. Users of HP servers should set the **HP power profile** to **maximum performance** and the **HP power regulator** to HP **static high-performance mode**.

Prior to deployment in a production environment, the ESXi esxtop utility can be used to measure statistics like CPU/memory utilization and turbo effects to determine through experimentation the best configuration for specific workloads.

2.2 Hyper-threading/Logical Processors

In non-virtualized environments, use of hyper-threading (i.e., logical processors) typically does not improve HPC performance. However, by enabling hyper-threading in the host BIOS when running ESXi, and configuring virtual machines (VMs) to use one physical core per virtual CPU (vCPU), the extra logical cores are available for use by ESXi hypervisor helper threads, delivering improved performance.

2.3 Single Root I/O Virtualization (SR-IOV)

If system support exists and SR-IOV capabilities are desired, SR-IOV should be enabled in the host BIOS. For a description of SR-IOV, please refer to section 8.2.

2.4 Memory/Node Interleaving

Memory interleaving allows the entirety of a system's memory controllers to work in parallel, providing maximum memory bandwidth to an application. If bandwidth is the limiting performance factor for important applications, interleaving should be enabled. However, in HPC environments, memory latency is often the most critical performance factor. In this case, disabling node interleaving will increase application performance,

with the resulting non-uniform memory access (NUMA) capabilities exploitable by ESXi, allowing VMs to achieve the lowest possible memory latencies and highest performance levels.

2.5 Above 4G Mapping/Encoding

Some high-end PCIe devices—notably some server-class NVIDIA GPU cards—use large, multi-gigabyte memory-mapped I/O (MMIO) device memory regions to transfer data between the host and the device. For example, the NVIDIA Tesla P100 PCI MMIO space is slightly larger than 16GB. To enable use of such devices, Above-4G mapping/encoding in the host BIOS should be enabled. Steps to enable this depend on server manufacturer. Searching in BIOS for "above 4G decoding," "memory mapped I/O above 4GB," or "PCI 64-bit resource handling above 4G" will help locate the appropriate parameter to enable.

It's important to note that different MMIO limitations exist across vSphere versions. If a GPU card does not use large PCI MMIO regions, it is not necessary to configure special settings for BIOS or advanced VM configuration parameters described in section 4.3. For more details, please refer to VMware vSphere VMDirectPath I/O: Requirements for Platforms and Devices.

2.6 MMIOHBase (Supermicro only)

PCI memory addressing limits occur in versions of ESXi prior to 6.5u1 and, in some cases, the host BIOS maps PCI memory regions beyond memory addressing limits. For example, a Supermicro node maps PCI memory regions to a 56TB starting address by default. To resolve this issue, Supermicro BIOS can be set to start PCI regions at 16TB (using MMIOHBase). This avoids triggering the limitation imposed by pre-6.5u1 ESXi versions and allows successful passthrough of large-BAR PCIe devices. The limitation is lifted in ESXi 6.5u1 and later.

3. ESXI SETTINGS

3.1 General

As VMware continues to expand support for various HPC requirements and achieve improved performance of HPC applications on VMware vSphere, upgrading to the latest version of vSphere is always recommended. Newer versions of vSphere offer added and enhanced features that are useful for HPC.

3.2 CPU

In general, though HPC workloads are very computation-intensive, it's unnecessary to reserve CPUs for VMs to achieve high levels of performance. For extremely latency-sensitive workloads, however, reserving CPUs to grant exclusive assignment of CPU cores to a VM is recommended.

ESXi requires a small number of CPUs to run its services, making it impossible to reserve the entirety of a system's physical cores for VMs. As an example, running ESXi 6.5 on a 20-core host allows a maximum of 17 cores to be reserved. To determine the number of cores reserved by ESXi, users should run the command "sched-stats-t groups | less-s" on ESXi. The returned resvMHz value indicates the total CPU resources reserved for ESXi services. The same value, divided by the system's MHz-per-core, indicates the number of reserved cores.

3.3 Topology

In vSphere 5.0 and later, ESXi presents a virtual NUMA topology to VMs. NUMA information provided to the guest OS and applications improves overall application performance. To fully optimize performance, however, additional steps are needed. All vCPUs should be scheduled on the same NUMA node and, when possible, available VM memory should be allocated outside of the physical memory attached to that node.

If a VM is larger than a single NUMA node (versions prior to vSphere 6.5), vNUMA must be manually configured to present the correct vNUMA topology to the guest OS. To manually configure vNUMA, set the VM's **cpuid.coresPerSocket** value to match the host's physical topology. In vSphere 6.5 and later, added improvements enable automatic, intelligent sizing and configuration of the virtual NUMA topology for a VM, regardless of its size. For more information, please see Virtual Machine vCPU and vNUMA Rightsizing—Rules of Thumb. For Linux VMs, it is often useful to (install, if necessary) and run the lstopo utility to ensure that the NUMA topology reported by the guest operating system is correct and matches the underlying hardware.

3.4 Memory

ESXi supports several different memory overcommit techniques, including transparent page-sharing, ballooning, memory compression, and swapping, designed to maximize the utilization of host memory. In HPC applications, however, memory overcommit often greatly reduces performance. Therefore, overcommitting memory is not recommended. Users should instead leave a portion of memory available to ESXi, separate from memory made available to VMs.

There are two major components of ESXi memory consumption:
1. A system-wide overhead for the hypervisor and various host agents
2. An additional overhead for each VM

The amount of ESXi-reserved memory-per-VM depends on a variety of factors, including the configured number of vCPUs, the configured memory size, and which platform features are enabled for the VM. The following table summarizes the amount of memory that can be reserved for VMs on ESXI 6.5, in general terms, without negatively impacting performance.

Host-level memory state can be monitored using memory statistics mode of the ESXi esxtop utility, indicating the amount of memory stress imposed on the host. Within ESXi, five memory states are associated with one or more memory reclamation techniques. A "high" memory state indicates enough free memory is available for VMs to run without memory pressure, enabling optimal performance for HPC applications.

For more information regarding ESXi memory management, please see vSphere Resource Management.

Table 1. Maximum VM memory guidance for HPC applications on ESXi 6.5

PHYSICAL CORES	HOST MEMORY	MAX GB OF MEMORY FOR VMS WITHOUT NEGATIVE IMPACT ON PERFORMANCE
16	128GB	118
24	256GB	240
32	512GB	486
64	1TB	976
96	2TB	1960

4. VIRTUAL MACHINE (VM) SETTINGS

4.1 VM Sizing and Placement

Generally, VMs should be sized according to the type of workload they will be hosting.

For message passing interface (MPI) workloads:
MPI workloads are defined as CPU-heavy, multi-host applications capable of employing all available cores on each host. Frequently called upon to perform a wide range of simulation-based tasks, these applications are very common in some HPC environments. MPI is also used in Machine Learning: Uber's Horovod distributed training framework, for example, is written as an MPI application.

As is the case in non-virtualized HPC environments, entire hosts are typically dedicated to running MPI applications. Consequently, one large VM per host is most often utilized in such cases. Due to the computationally intensive nature of these applications, CPU or memory over-commitment can greatly impact performance if the amount of actively used resources exceeds the host's physical limits.

When an MPI application is to be deployed with a high-speed interconnect enabled via DirectPath I/O, the VM's memory must be reserved. The same is true when using SR-IOV technology for high-speed Ethernet networking cards or RDMA interconnects.

For throughput workloads:
Throughput workloads scale horizontally and benefit from the allocation of more than one VM per host. As described earlier, strict adherence to NUMA boundaries for VM sizing enables optimal performance for throughput workloads.

In virtualized HPC (vHPC) environments, CPU oversubscription (unlike memory oversubscription) can be leveraged to achieve higher throughput relative to bare-metal environments. For a detailed performance study, please see Virtualizing HPC Throughput Computing Environments.

Other sizing guidelines
Sizing guidelines for VMs are identical for both MPI and throughput workloads.

Allocate only the amount of virtual hardware required for the type of workload to be run in a VM. Note that while CPU overprovisioning can be helpful, actual overuse of available resources can negatively impact performance of the VM and the system.

For HPC or ML workloads that require accelerators configured either by Direct Path I/O mode, SR-IOV mode, or NVIDIA vGPU, reserved memory is required. Optimal data-transfer performance between CPU and GPU is achieved by accessing local, socket-attached memory and employing the CPU socket to which the required PCI devices are attached. Using processor affinity for vCPUs (numa.nodeAffinity) ensures the VM with accelerator was scheduled on the NUMA node to which the accelerator is attached and the VM's memory is appropriately allocated from the NUMA node's local memory, thus optimizing application performance. While this degree of optimization cannot typically be achieved in a production environment, it can be useful as part of a specialized deployment of workloads requiring the absolute highest performance.

4.2 VM Networking and Latency Sensitivity

For extremely latency-sensitive workloads like financial-trading systems, vSphere provides configuration options for reducing latency and jitter. Important vSphere per-VM tuning options include the following:

- Set **latency sensitivity** to **high** (default is **normal**)
  This setting enables a series of networking optimizations that improve latency and reduce jitter. Within ESXi 6.7, setting latency sensitivity to high requires full CPU and memory reservation.

- Fully reserve CPUs and memory
  When latency sensitivity set to high and CPUs are fully reserved, exclusive pCPU access is enabled for the specific VM. In this case, each vCPU owns a specific pCPU and no other vCPUs are allowed to run there. With exclusive pCPU access, the ESXi CPU scheduling layer is bypassed for with regard to the specific VM, reducing VMkernel and VMM context switches and therefore reducing network latency and jitter.

  In instances where latency sensitivity is not set to high, fully reserving CPU and memory can still help reduce latency and jitter.

- Use DirectPath I/O or SR-IOV
  Virtual network adapters like vmxnet3 incur an overhead in virtualized environments. When reduced latency is desirable and core virtualization management features like vMotion are not needed, DirectPath I/O or SR-IOV can be employed to enable direct VM-to-network device access. For more information, please see the High-Speed Interconnects section of this white paper.

To learn about networking-specific tuning for extremely latency-sensitive workloads, please refer to Best Practices for Performance Tuning of Latency-Sensitive Workloads in vSphere VMs and Deploying Extremely Latency-Sensitive Applications in VMware vSphere 5.5.

4.3 Advanced VM Settings

As previously discussed, a number of high-end PCIe devices utilize a large MMIO space, requiring special BIOS settings to enable their use with ESXi. In addition to these BIOS changes, a VM deployed with such a device must be configured (prior to installation of the guest OS) to use UEFI for booting rather than the classic master boot-record approach.

Two custom settings must also be added to the VM's configuration:

1. **pciPassthru.use64bitMMIO = "TRUE"** enables 64-bit MMIO support for the use of large PCI MMIO space
2. **pciPassthru.64bitMMIOSizeGB**, set to the size of the desired MMIO region as a power-of-two number of GB. To calculate the value, sum the GPU memory sizes of all GPUs allocated to the VM and then round up to the next power of two. For example, to use two NVIDIA 32GB Tesla V100 GPUs in passthrough mode in a single VM, set pciPassthru.64bitMMIOSizeGB = "128", where 128 is calculated as 32 + 32 = 64, which is then rounded up to 128.

5. GUEST OPERATING SYSTEMS

All guest OS tuning is similar to that applied in bare-metal HPC environments.

5.1 Choice of Guest Operating Systems

Linux, including RHEL, CentOS, Ubuntu, SUSE, and others, is the common guest OS choice for running HPC applications, though some workloads such as electronic design-automation heavily utilize Windows. For ML workloads, including those employing Caffe2 and TensorFlow frameworks, Ubuntu is the most commonly used guest OS.

5.2 OS Power Management

As a general rule, it's unnecessary to apply power-management settings within the guest OS as these are managed by ESXi (as described in section 2.1).

5.3 OS Networking

Guest firewall rules (e.g., Linux iptables) typically increase networking I/O latency. If a given VM's security policy allows it, users should disable the firewall in the guest OS.

For latency-sensitive workloads, a **latency performance** profile is recommended, assuming the guest OS supports such profiles. For high-networking, bandwidth-oriented workloads, a **bandwidth performance** profile is recommended. To set a latency-performance profile, run **tuned-adm profile latency-performance** in the guest OS. **tuned-adm** is a Linux command-line utility that enables the OS switching between tuned profiles to improve performance. For detailed information about using tuned-adm, see its main page.

6. STORAGE CONSIDERATIONS

Two types of storage design exist in a vHPC environment: storage for the VM's virtual disks (VMDKs), including its boot disk and operating system files, and storage used for HPC application data.

6.1 Local or Shared Storage for VMDK

In an vHPC environment, VMDKs are most typically used only for boot disks, though some HPC applications may additionally benefit from their use as local scratch disks. For ML training, in which large volumes of data are continually transferred from storage into a compute accelerator's memory, performance can be improved by staging the training data within VMDKs in close physical proximity to the server.

Depending on the specifics of the HPC deployment, either local or shared storage can be an appropriate approach for hosting virtual disks.

Local Storage

When Direct Path I/O (section 8.1) is employed for hardware accelerators, or for direct access to high-speed interconnects, the associated VMs cannot be migrated with vMotion. Since the VMs will not be moved, it is not important from a mobility perspective to locate VMDKs on shared storage, making local disks a suitable choice for VMDK storage.

Shared Storage

VMDKs created in shared datastores offer better disk utilization of the underlying storage capacity and, most important, support efficient live migration of VMs for load-balancing, maintenance events, and the like. Shared storage is the most common approach used in virtual environments and is likewise recommended for vHPC environments. Choosing between vSAN, SAN, and NFS involves the same considerations taken into account for non-HPC environments.

See the recommendations in Performance Best Practices for VMware vSphere for optimal performance of vSAN and NFS for VMDK storage.

6.2 Shared File Systems for HPC Application Data

HPC environments invariably use some type of globally accessible, network-attached storage to provide uniform access to application data from all nodes in the HPC cluster. This in-guest storage access usually involves one or two approaches.

NFS can be used as a shared files system for HPC application data. To ensure appropriate NFS performance, the network connectivity to the NFS server from the VMs should be high performing. Follow the recommendations in Best Practices for Performance Tuning of Latency-Sensitive Workloads in vSphere VMs to achieve the best networking performance for extremely latency sensitive workloads, which include I/O workloads involving many small random reads and writes between the NFS server and its clients.

Some HPC deployments—often larger installations with higher performance requirements—take advantage of Lustre, IBM Spectrum Scale, BeeGFS, or other high-end parallel file system solutions. These file systems support simultaneous access from multiple compute nodes and transfers of massive amounts of data. The architecture of parallel file systems makes it easy to scale in capability and performance. These file systems can be accessed by applications running within VMs by mounting them in the standard way within the guest OS.

7. NETWORK CONSIDERATIONS

Ethernet-based networking is a common requirement of traditional HPC for management purposes. High-speed Ethernet (10GbE and higher) represents the de-facto standard for virtualization.

Virtual networking based on Ethernet is used for all administrative traffic, NFS traffic, login sessions, and the like. It is occasionally used for MPI when applications do not have high-latency sensitivity. In advanced security and networking scenarios, VMware NSX® can be employed. VMware NSX Data Center is the network virtualization platform for the software-defined data center (SDDC), delivering networking and security entirely in software and abstracted from the underlying physical infrastructure. With optimized networking and security policies, NSX can provide secure, multi-tenant virtualized HPC environments that allow organizations to increase overall hardware utilization. NSX achieves this by simultaneously centralizing resources and supporting the data privacy and compliance needs of multiple teams.

8. COMPUTE ACCELERATORS

In ML, and in DL in particular, larger data sets and models lead to better accuracy. This also leads to significantly increased computation, however. As a result, DL with deep neural networks necessarily relies on accelerators to increase performance. Many traditional HPC applications have been rewritten to take advantage of these accelerators.

GPUs are the most common type of compute accelerator used for HPC and ML/DL workloads. These are discussed in greater detail in the following sections.

8.1 Full and Multi-GPUs vs. Fractional GPUs

Traditional, GPU-enabled HPC applications typically require an entire GPU or multiple GPUs. For ML/DL, three major workflows exist—development, training, and inference—each with differing requirements.

During initial development, data scientists often use laptops or desktop systems to perform exploratory data analytics, which often include building models based on reduced data sets. In cases where this data is confidential, virtual desktops provide server-class resources while ensuring that the relevant data is retained within the data center. Granting access to fractional GPUs may be appropriate, as it allows multiple data scientists to share an expensive underlying resource and enable higher utilization of the hardware.

Deep neural network architectures, including convolutional neural network (CNN) and recurrent neural network (RNN), require compute-intensive training involving repeated forward and backward propagation of parameters and repeated processing of the entire input training set. When transitioning from development to training, employment of full GPUs or multiple GPUs is recommended in order to train larger neural network models, or train multiple neural network models in parallel, using either data parallelism or model parallelism.

Data parallelism trains multiple copies of a model on multiple hosts in parallel with different subsets of input data. Model parallelism decomposes the model or neural network layers into pieces and maps the pieces onto separate hosts. Both types of parallelism can scale to multiple GPUs across multiple hosts, in which case interconnect latency and bandwidth may limit performance.

The solution is to use high-performing interconnects, such as the RDMA-based approaches covered in High-Speed Interconnects in this guide. For example, running the Horovod distributed training framework can use multiple GPUs across hosts, and with RDMA interconnects it can achieve higher scaling efficiency than Ethernet.

Inference only needs to perform forward propagation on one or a small number of input examples to generate predictive results and is thus less compute-intensive than training. When deploying trained neural network models into production and performing real-time inference, fractional GPUs can be leveraged for increased sharing and better resource utilization.

To enable full GPUs or multi-GPUs, vSphere supports DirectPath I/O.

To enable fractional GPUs, vSphere supports two solutions: NVIDIA GRID vGPUs and BitFusion, a partner offering that supports the ability to provide remote access to GPUs to VMs anywhere in the data center.

8.2 Advanced Management Features

Beginning with vSphere 6.7, vSphere provides the capability to suspend and resume NVIDIA vGPU-accelerated VMs. It enables sharing GPU resources across mixed workloads with minimal disruption. For example, a user may run ML training at night, suspend VMs in the morning and run interactive jobs during daytime.

vSphere 6.7u1 supports vMotion for NVIDIA vGPU-accelerated VMs. This enables maximized data-center utilization, improving productivity and reducing costs. For example, virtual desktop infrastructure (VDI) VMs become idle at night. These can be consolidated by live migrating to a different host, allowing the original host to be repurposed for HPC/ML workloads.

To benefit from these features, regularly upgrading to the latest version of vSphere is recommended.

8.3 GPU Direct

NVIDIA GPUDirect is a family of NVIDIA technologies that enables direct data exchange between multiple GPUs, third-party network adapters, and other devices. GPUDirect P2P enables data buffers to be directly exchanged between the memories of two GPUs within a host. GPUDirect RDMA enables host channel adapters (HCAs) to write-and-read GPU memory data buffers without the need to copy data to host memory, offloading the burden of CPU. Both have demonstrated significant performance improvements for GPU-accelerated HPC and ML/DL applications.

To take advantage of GPUDirect between GPUs within a host, the host must support NVIDIA NVLINK technology and GPUs must be NVLINK-connected.

To take advantage of GPUDirect RDMA between GPUs across hosts, GPUs and RDMAs must in DirectPath I/O mode and requires vSphere 6.7 or beyond.

9. HIGH-SPEED INTERCONNECTS

For extremely demanding latency-sensitive or high-throughput workloads, HPC environments often rely on use of dedicated remote direct memory access (RDMA) interconnects between compute and storage nodes to deliver optimal application performance. RDMA allows direct-memory access from the memory of one computer to the memory of another, without involving the OS or host CPU, as the transfer of memory is offloaded to the RDMA-capable HCAs. MPI implementations leverage RDMA semantics between compute nodes while parallel file systems employ RDMA transfers with large bandwidth and reduced CPU usage. RDMA interconnects can be configured in three ways in vSphere: DirectPath I/O, SR-IOV. and PVRDMA.

9.1 DirectPath I/O

In vSphere, PCI devices can be configured in DirectPath I/O (passthrough) mode, allowing a guest OS to directly access the device and essentially bypass the hypervisor. Because of the shortened access path, performance of applications accessing the device in this way are often very close to that of bare-metal systems. With DirectPath I/O, configuration of one or multiple interconnects into a single VM is possible.

VMware supports DirectPath I/O as a vSphere feature, however support varies by hardware OEM. Also, some features are unavailable for VMs configured with DirectPath I/O, including vMotion, hot-adding and removal of virtual devices, taking snapshots, and distributed resource scheduler (DRS) and high availability (HA).

9.2 SR-IOV

Single-root I/O virtualization (SR-IOV) is a specification that allows a single PCIe physical device to appear as multiple separate physical devices. SR-IOV uses physical functions (PFs) and virtual functions (VFs) to manage global functions for the SR-IOV devices. PFs are full PCIe functions capable of configuring and managing the SR-IOV functionality. SR-IOV can be used to share a single InfiniBand or RDMA over Converged Ethernet (RoCE) connection across multiple VMs by assigning a VF to each VM.

Appropriate BIOS, hardware, and hypervisor or guest driver support are required for use of SR-IOV. As with DirectPath I/O, SR-IOV is not compatible with certain core vSphere features.

9.3 PVRDMA

In HPC environments, RDMA devices improve performance in the form of low latency and high bandwidth. While this performance improvement is attractive, RDMA devices accessed from within the guest OS using passthrough or SR-IOV mode cannot take advantage of VMware functions.

Released in vSphere 6.5, paravirtual RDMA (PVRDMA) enables RDMA in the VMware virtual environment and maintains features like vMotion, distributed resource scheduler, and taking snapshots that are unavailable in DirectPath I/O or SR-IOV mode. To use PVRDMA, a supported RoCE card and compatible ESXi release are required. RoCE (i.e., RDMA over Converged Ethernet) ensures low-latency, lightweight, and high-throughput RDMA communication over an Ethernet network.

10. SUMMARY

Virtualization has matured significantly in recent years, offering the functionality and high-performance foundation ideal for deployment of virtualized HPC/ML environments. The best practices outlined in this paper facilitate the optimal deployment of virtualized HPC/ML clusters on vSphere.

GLOSSARY

BAR: PCI base area register
CNN: convolutional neural network
DRS: distributed resource scheduler
UEFI: unified extensible firmware interface
FPGA: field-programmable grid array
GPGPU: general purpose graphics processing unit
HA: high availability
HCA: host channel adapter
HPC: high-performance computing
HPM: host power management
LRO: large receive offload
ML: machine learning
MMIO: memory-mapped I/O
MPI: message passing interface
NFS: network file system
NUMA: non-uniform memory access
PCIe: peripheral component interconnect express
PF: physical function
PVRDMA: paravirtual remote direct memory access
RDMA: remote direct memory access

RNN: recurrent neural network
RoCE: remote direct memory access over converged ethernet
SDDC: software-defined data center
SR-IOV: single root I/O virtualization
vCPU: virtual CPU
VF: virtual function
vGPU: virtualized graphics processing unit
vHPC: virtual high-performance computing
VM: virtual machine
VMFS: virtual machine file system
VMM: virtual machine monitor
vNUMA: virtual non-uniform memory access

AUTHORS

Na Zhang is senior member of technical staff working on High-Performance Computing (HPC) within VMware's Office of the CTO. She has been working on various vHPC topics, including performance tuning for throughput workloads, MPI workloads, and financial services workloads in virtual environment, design, and implementation of vHPC tools, accelerator solutions, and integration of HPC middleware with VMware products. She received her Ph.D. in Applied Mathematics from Stony Brook University in 2015. Her research primarily focused on design and analysis of parallel algorithms for large- and multi-scale simulations running on world-class supercomputers. She has served on the Technical Program Committee for more than 10 international HPC conferences and workshops, including SC (The International Conference for High-Performance Computing, Networking, Storage, and Analysis, vHPC, HPC&S), vHPC (Workshop on Virtualization in High-Performance Cloud Computing), and HPC&S (The International Conference on High Performance Computing & Simulation).

Josh Simons is the Chief Technologist for HPC. With more than 20 years of experience in high-performance computing, he currently leads an effort within VMware's Office of the CTO to bring the value of virtualization to HPC. Previously, Josh was a Distinguished Engineer at Sun Microsystems with broad responsibilities for HPC direction and strategy. He joined Sun in 1996 from Thinking Machines Corporation, a pioneering company in the area of Massively Parallel Processors (MPPs), where he held a variety of technical positions. Josh has worked on developer tools for distributed parallel computing, including language and compiler design, scalable parallel debugger design and development, and MPI. He has also worked in the areas of 3D graphics, image processing, and real-time device control. Josh has an undergraduate degree in Engineering from Harvard College and a Masters in Computer Science from Harvard University. He has served as a member of the OpenMP ARB Board of Directors since 2002.

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com