WHITE PAPER – **MARCH 2018**

HARVESTING FREE VIRTUAL DESKTOP INFRASTRUCTURE COMPUTE CYCLES FOR HIGH PERFORMANCE COMPUTING WORKLOADS USING VMWARE vSPHERE®

vmware®

Contents

Introduction

High Performance Computing (HPC) workloads have traditionally been run only on bare-metal, unvirtualized hardware. However, performance for these highly parallel technical workloads has increased dramatically with the introduction of hardware support for virtualization, enabling organizations to begin embracing the numerous benefits that a virtualization platform can offer.

Jackson National Life Insurance Company (Jackson) is at the forefront of this trend because of its design and implementation of a novel approach that incorporates financial HPC workloads into Jackson's existing VMware virtualized environment. Specifically, harvesting unused computing capacity from its virtual desktop infrastructure (VDI) enables Jackson to deliver significant value to its organization. The approach is general and can also be applied to other types of HPC workloads as long as the appropriate licensing requirements are satisfied: 1) the HPC workloads are running on desktop operating system (OS) instances; 2) the VMware Horizon® agent is running on all such instances; and 3) these instances all count against purchased Horizon user licenses.

In this paper, we describe the broader trends toward adoption of virtualization for HPC, explain the Jackson deployment in detail, and provide performance data demonstrating the value of this approach.

Virtualizing HPC Workloads

The following are among the many benefits that virtualization offers to an enterprise that can often also add value in HPC environments:

- Supports a more diverse end-user population with differing software requirements – By using virtual machines (VMs), each user or group can run the OS and other software that are most effective for their needs. These various software environments can be freely mixed on the same hardware. Furthermore, that mix can be changed dynamically as user requirements change. This enables IT organizations to increase their overall agility and to help decrease time to solution for researchers, scientists, and engineers.
- Provides data security and compliance by isolating user workloads into separate VMs, including running within different virtual networks – This ensures that projects, such as studies involving clinical data, maintain control over their data and that the data is not shared inappropriately with other users. Moreover, data security can be achieved while enabling projects to share underlying hardware, increasing overall utilization of resources.
- Provides fault isolation, root access, and other capabilities not available in traditional HPC environments, via the use of VMs – The isolated nature of the VM enables root access to be granted to those users who require it. Because that privilege is granted only within the VM, it does not compromise the security of other users or their data. In addition, the VM abstraction isolates one user's jobs from issues caused by jobs of other users.

- Creates a more dynamic IT environment in which VMs and their encapsulated workloads can be live-migrated across the cluster for load balancing, maintenance, fault avoidance, and so on – This is a considerable advance over the traditional approach of statically scheduling jobs onto a bare-metal cluster without any ability to reassess those placement decisions after the fact.

HPC Workloads

In HPC, performance is of paramount importance. Delivering high performance in the VMware virtual environment has been a key part of the work required to make virtualization viable for these workloads.

HPC workloads can be broadly divided into two categories: throughput workloads and parallel distributed workloads. Throughput workloads often require that a large number of tasks be run to complete a job, with each task running independently and no communication between tasks. Rendering the frames of a digital movie is a good example of such a throughput workload. In this example, each frame can be computed independently and in parallel. The overall job is completed when all frames have been computed. Throughput workloads currently run on VMware vSphere® with very little degradation, typically a percentage point or two. In some circumstances, they can run slightly faster when virtualized. Figure 1 shows performance comparisons for a popular set of throughput benchmarks for life sciences, illustrating that virtual performance for this workload class can be very similar to unvirtualized performance.

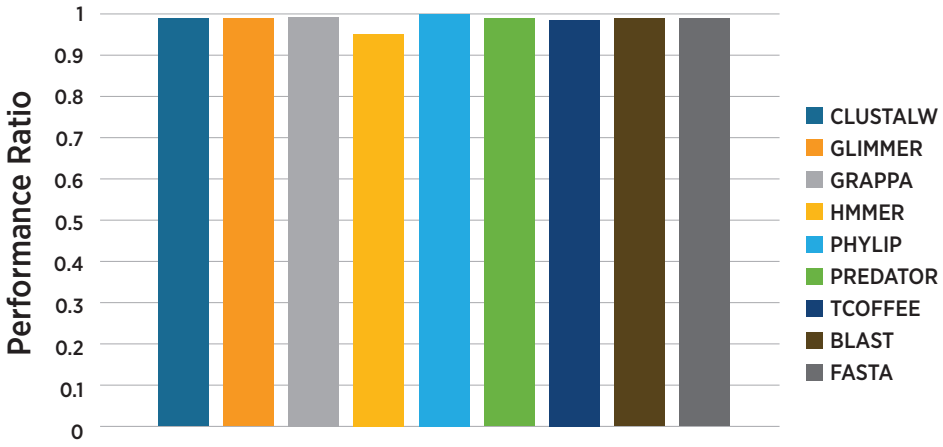

Figure 1. Performance Comparison of Typical HPC Throughput Applications – Virtualized Against Unvirtualized

NOTE: Higher is better.

Parallel distributed applications represent the second category of HPC workload. They are often referred to as MPI applications, referring to the most popular messaging library for building such applications. Unlike throughput workloads, these applications consist of many simultaneously running processes that communicate with each other, often with extremely high intensity. Because this communication is almost always in the critical performance path, the HPC community has adopted specialized hardware and software to achieve the lowest possible latency and highest bandwidth to support the efficient performance of these applications.

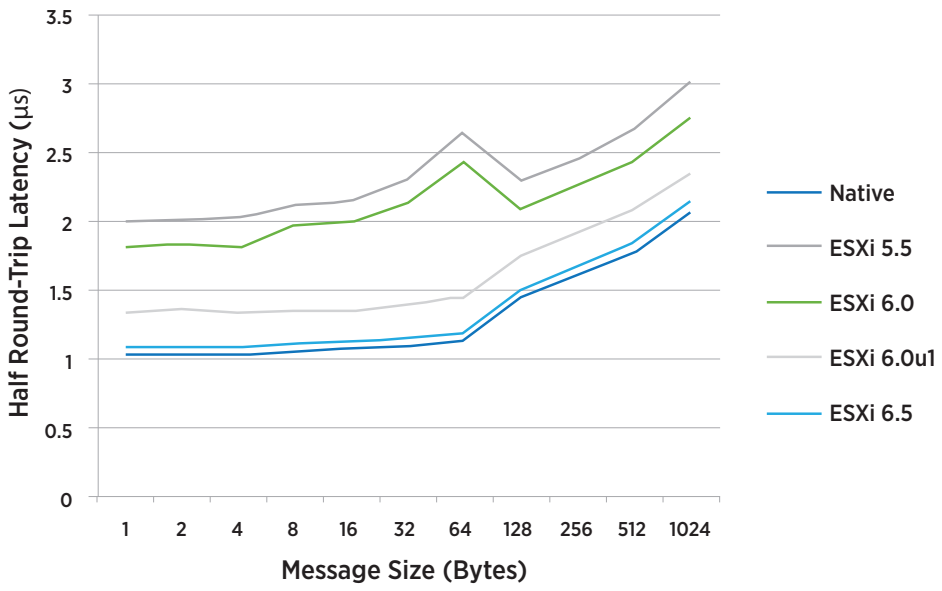

Figure 2. Improvement in InfiniBand Latency over Several Versions of the VMware ESXi Hypervisor

NOTE: Lower is better.

InfiniBand and RDMA are the two most widely used hardware and software approaches for HPC message passing. They can also be used in a virtual environment. Figure 2 shows how InfiniBand latencies under a vSphere platform using VMware vSphere DirectPath I/O™ have improved over the last several releases of VMware ESXi™, the vSphere hypervisor. Latencies now approach those achievable without virtualization. Figure 3 shows performance results for a variety of popular open-source and commercial MPI applications. The figure illustrates how degradations can be higher with this workload class than with throughput workloads. Because overheads depend on the specific application, the model being used, and the scale at which the application is run, a proof of concept (PoC) deployment is often recommended to determine achievable and acceptable performance.

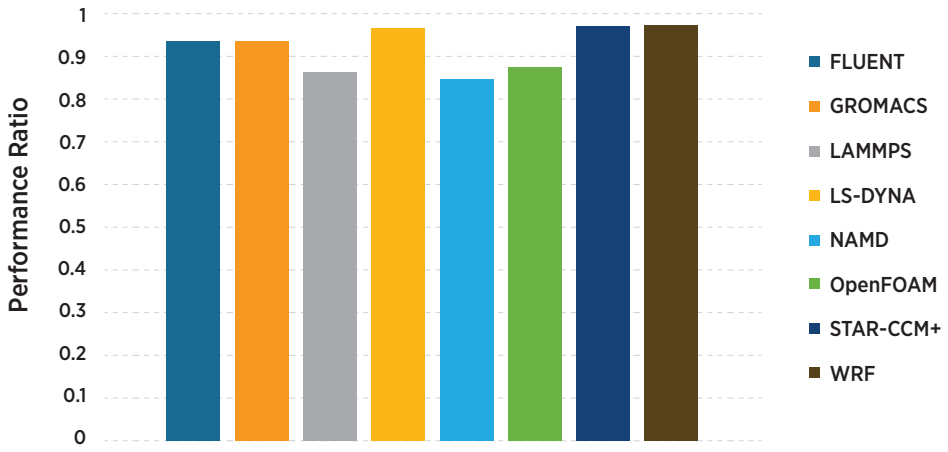

Figure 3. Performance of Various MPI Applications Running on a 16-Node Cluster Using 320 MPI Processes

NOTE: Results are shown as the ratios of unvirtualized and virtualized performance. Higher is better, meaning that a ratio of 1.0 indicates that virtual performance matches that of bare metal.

Cycle Harvesting

Cycle harvesting, the consumption of otherwise idle compute cycles to opportunistically perform useful work, is an old technique that dates back at least to the 1980s. Perhaps one of the best-known early examples of cycle harvesting is the SETI@home project, part of the Search for Extraterrestrial Intelligence (SETI) endeavor. The project harvests unused cycles from machines on which volunteers have allowed the SETI@home daemon to run, using those cycles to analyze radio telescope data for signals that might indicate the presence of extraterrestrial intelligence.

SETI@home is quite popular, with more than 100,000 active users contributing cycles from more than 150,000 machines. The project proved so popular that the underlying cycle-harvesting infrastructure created for SETI@home was generalized and released as the Berkeley Open Infrastructure for Network Computing (BOINC) in 2002. Today BOINC hosts more than 100 projects that span a wide array of scientific areas. There are more than 200,000 active users and more than 1,000,000 machines.

In the more traditional HPC realm, the HTCondor system has been used for cycle harvesting for many years. Originally called Condor, it has been available from the University of Wisconsin since the late 1980s.

Although cycle harvesting has traditionally been used to scavenge cycles from desktop machines when they become idle, the technique can also be applied to servers. Regardless of the hardware being used, the approach typically involves running an agent locally on each machine to coordinate implementation of the appropriate cycle-harvesting code and management of input and output data.

It was with this historical context that Jackson National Life Insurance Company contemplated how best to harvest the unused cycles on its Horizon VDI. A straightforward application of existing approaches would have dictated that some harvesting software be installed within each of Jackson's virtual desktop VMs. However, this injection of software into the user's environment and the resulting additional management complexity, data security questions, and reliability implications to user desktops was not compelling.

Instead, Jackson decided to explore whether the vSphere platform could somehow be used directly to support cycle harvesting without modifying any existing user environments. By combining built-in vSphere capabilities with good engineering and configuration management practices, the Jackson team was able to craft a simple, elegant, and extremely effective cycle-harvesting solution.

The remainder of this paper describes Jackson's solution in detail.

HPC Environment

From the perspective of its HPC users, the Jackson HPC computing infrastructure consists of two clusters controlled by the Microsoft HPC Job Manager. In Microsoft terminology, each job submitted to the HPC Job Manager comprises multiple tasks. These jobs are inserted via a head node into a queue. Then the HPC Job Scheduler load-balances and schedules execution of the jobs and their component tasks onto the nodes of the compute clusters.

The traditional HPC cluster at Jackson has been a bare-metal Microsoft Windows cluster running the HPC Job Manager. When the harvesting solution was brought online, it appeared to the HPC users that a second bare-metal cluster had been procured. In actuality, the "new" cluster comprised VMs provisioned within the Jackson VDI estate.

VDI Environment

The Jackson VDI environment consists of two identical data centers. Each data center consists of 13 enclosures of 16 blades each, plus 16 rack-mounted systems. The overall design focuses on user productivity, in the form of uptime and responsiveness, with the following imperatives:

- Use a shared-nothing architecture.
- Reduce fault domains to a minimum number of users.
- Spread departments and functions across data centers.
- Increase throughput via parallelization.
- Maintain a high core count for peak usage.
- Avoid memory overcommitment, even in a failover situation.
- Keep the average VDI hardware cost below USD $20 per month per VDI.

Host Architecture

A typical physical host has two CPUs, each with 8 to 14 cores. Hyperthreading is enabled on all systems. The hosts run three types of VM.

- Virtual desktop machines – To support multiple service levels for end users, these VMs are of various sizes regarding number of virtual CPUs (vCPUs), amount of virtual memory, and virtual disk. Although there can be as much as a 5:1 CPU oversubscription with this class of VM, the memory allocated to these VMs will not exceed 50 percent of physical memory to allow for failover of VDI workloads from the other data center.
- One NFS server per host – This VM has two vCPUs, one pinned to a core in each physical CPU. This VM serves an NFS file share to the end-users' virtual desktops.
- Two cycle-harvesting VMs – Each harvester is typically sized to have as many vCPUs as there are cores per physical CPU socket, with 7GB of virtual RAM (vRAM) allocated per vCPU. For systems with less memory, on which the allocation of 7GB of vRAM per harvester vCPU would consume more than 50 percent of system memory, the harvester vCPU counts are reduced slightly.

Figure 4 shows the number of CPUs for all the VMs that run on a typical host. The VDI workload is configured to fit within 50 percent of a host's physical memory, and the harvesters together are sized similarly. Using this approach, harvester VMs can be terminated and enough free memory guaranteed on the system to support the additional VDI workloads that would be hosted in the event the other data center fails.

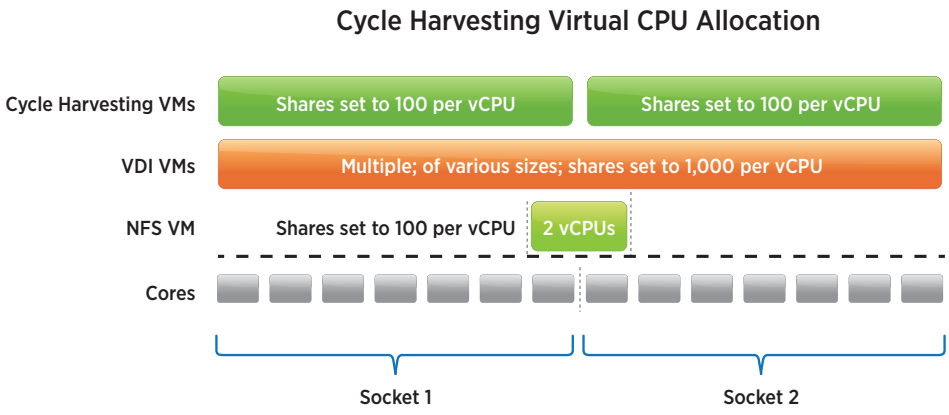

Figure 4. CPU Design for VMs

NOTE: Although the VDI VMs typically are CPU oversubscribed by about 5:1, the harvester VMs generally are sized to exactly match the number of cores per physical CPU socket.

Each host has two SSD drives for storage, each with at least 1.6TB capacity and formatted as single VMware vSphere VMFS datastores. On the first SSD's datastore, there is one VMDK that is used to back the NFS server VM previously described. The second SSD's datastore contains a replica of the NFS VMDK from the corresponding blade at the mirror data center. The VMDK from the first datastore is similarly replicated to the corresponding blade at the mirror data center. For example, the NFS VMDK from "vdihost6001a" in the first data center is replicated onto "vdihost6001c" in the second data center; the NFS VMDK from "vdihost6001c" in the second data center is replicated onto "vdihost6001a" in the first data center. All files are synchronized using VMware vSphere Replication™. Therefore, the two corresponding blades in the two data centers each have one local NFS VMDK and one replica VMDK, each on a separate SSD. With this design, a failure of one blade is protected by the replica copy of the SSD at the mirror data center.

The cycle-harvesting VMs have a virtual disk for scratch that is backed by a VMDK file on the SSD datastore, one for each harvester. The size of the scratch disk is equal to 30GB times the number of cores. This scratch disk is not replicated because there is never any critical data on it. Figure 5 illustrates the storage architecture of the hosts.

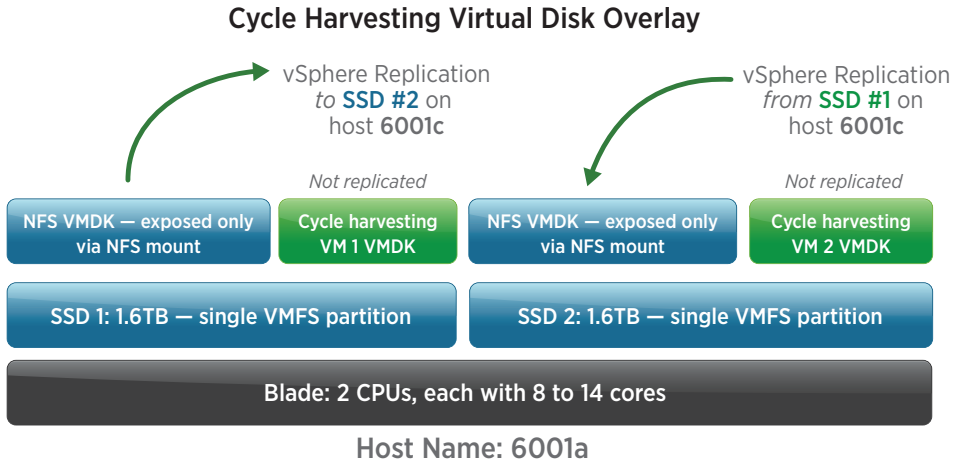

Figure 5. Storage Design for Hosts

By configuring the cycle harvesters with as many vCPUs as there are cores on a socket, and by scaling the vRAM and scratch disk by the number of cores, the harvesters can perform as much work as the physical host can handle in situations where the VDI VMs become completely idle. The job scheduler allocates jobs on a per-core basis, thereby ensuring that these VMs are never oversubscribed.

Table 1 shows a summary of the physical infrastructure.

ENVIRONMENTAL SUMMARY	
444 hosts being cycle harvested	Total capacity of 71.6 million I/Os per second (4K RW 50%)
10,468 cores	Total throughput of 4.022TB/second to disk
71TB of RAM being used by cycle harvesters in 155TB physical RAM	6,768 VDI desktops
280TB of SSD being used out of 2007TB physical local SSD	888 cycle-harvesting desktop VMs

Table 1. Summary of Physical Infrastructure

Resource Sharing

Because the primary purpose of the environment is to serve VDI desktops, vSphere resource shares for CPU and memory are used to ensure best performance. Resource shares in vSphere systems provide a relative resource guarantee to VMs during times of contention. However, unlike a reservation, they enable idle resources to be consumed by other VMs beyond their allocated shares when there is no contention. The VDI VMs are configured with 1,000 shares per vCPU; the NFS server VM is also configured with 1,000 shares per vCPU. The cycle-harvesting VMs, in contrast, are configured with only 100 shares per vCPU.

VM TYPE	SHARE SETTING (PER VCPU)
VDI Desktop	1,000
NFS File Share	1,000
Cycle Harvester	100

Table 2. Share Setting by VM Type

Although this seems quite simple, the management and enforcement of resource shares by the ESXi hypervisor is in fact **the linchpin** that enables this cycle-harvesting approach to succeed where other, more complex approaches had failed. Jackson personnel carefully performed tests that confirmed that this relative resource-share setting is sufficient to ensure that the VDI desktops always experience best performance without any drops in responsiveness. It also enables the cycle harvesters to employ enough cycles to perform a meaningful amount of work. The following section discusses the system performance in detail.

Performance

As part of our performance analysis, to get a sense of the size of the opportunity for harvesting cycles, we first looked at the CPU utilization of a set of VDI VMs on a randomly selected host over a 24-hour period. This host had 28 cores and 384GB of memory, so its harvesters were configured with 12 vCPUs each to avoid the memory overcommitment issue mentioned in the "Host Architecture" section.

One might expect that VDI CPU utilization is relatively constant during business hours and idle during off-hours. Figure 6 shows that this is not the case. The usage pattern varies widely between virtual desktops as well as between hosts. Some show very little activity, some show a more classic workday pattern, and others show significant CPU usage in off-hours.

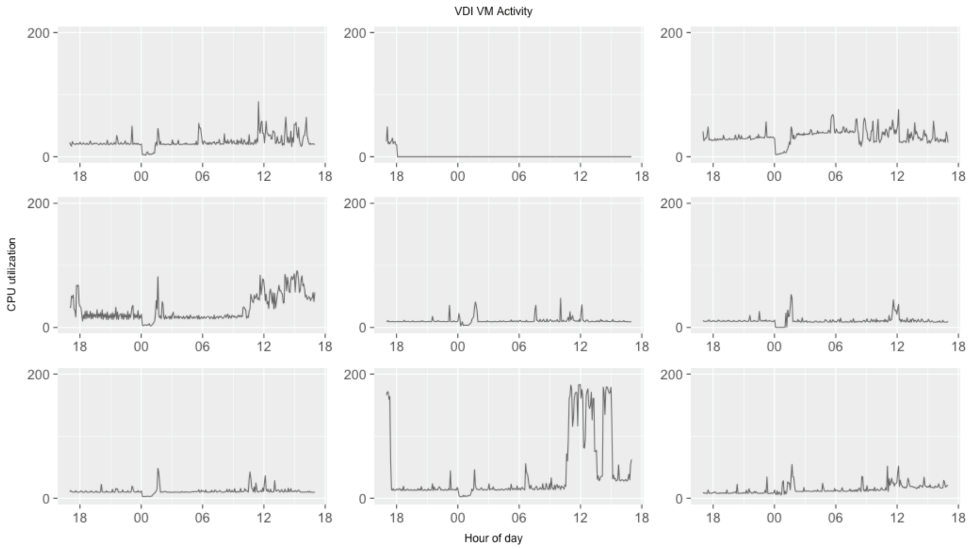

Figure 6. CPU Utilization for a Set of VDI VMs on One Jackson Host Shown from 5 PM to 5 PM and Gathered at 5-Minute Intervals Using the esxtop Utility

NOTE: The charts show that usage is not restricted to normal business hours and that business-hour usage is irregular.

We then summed utilization of all VDI VMs running on the host to determine the total load placed on the system by VDI workloads. Figure 7 plots the results and shows CPU utilization of the host's two harvester VMs. In aggregate, VDI activity on this lightly loaded host generally conforms to a pattern of higher use during the business day, with between four and six CPUs consumed. But usage does not fall to zero in off-hours. As with the plots of individual VDI VM utilization, there is significant high-frequency fluctuation in aggregate VDI utilization, due to the interactivity of desktop sessions.

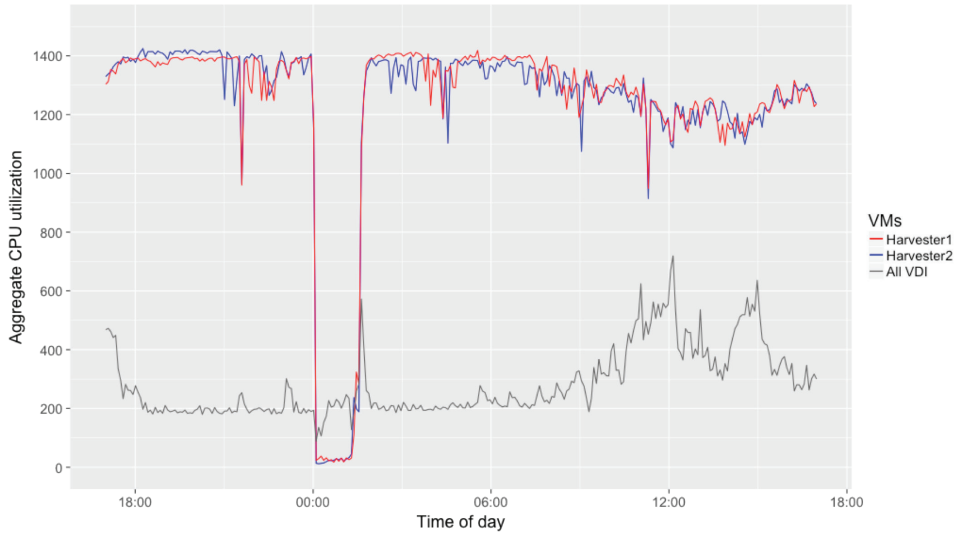

Figure 7. Aggregate CPU Utilization

NOTE: Total VDI CPU utilization on a single host over a 24-hour period is shown in black. The utilization of the two cycle-harvesting VMs is shown in red and blue. Significant harvesting occurs during normal business hours. The harvester dropouts at midnight occurred when the Microsoft HPC Job Scheduler temporarily ran out of jobs to schedule.

The surprise illustrated in Figure 7 is the amount of cycle harvesting achieved with this approach. Although the degree of harvesting dips in midday to about 12 cores per harvester, each harvests about 14 CPUs for HPC use during VDI off-peak hours. The increase beyond 12 cores for these 12-vCPU VMs is due to Intel Turbo Boost Technology.

Figure 8 sums the three curves from Figure 7 to show overall CPU utilization on the host. On this 28-core system, the maximum CPU utilization percentage is (28 cores) * (100%) * (1.25 hyperthreading increase) = 3,500. Because the two harvesters on this small-memory system are each configured with 12 rather than 14 vCPUs, the total utilization achieved is about 85 percent of overall system capacity. For systems on which the harvesters are configured to match the number of cores per CPU socket, overall utilization is very close to 100 percent. In either case, the systems are now being driven at consistently high utilization 24 hours per day.

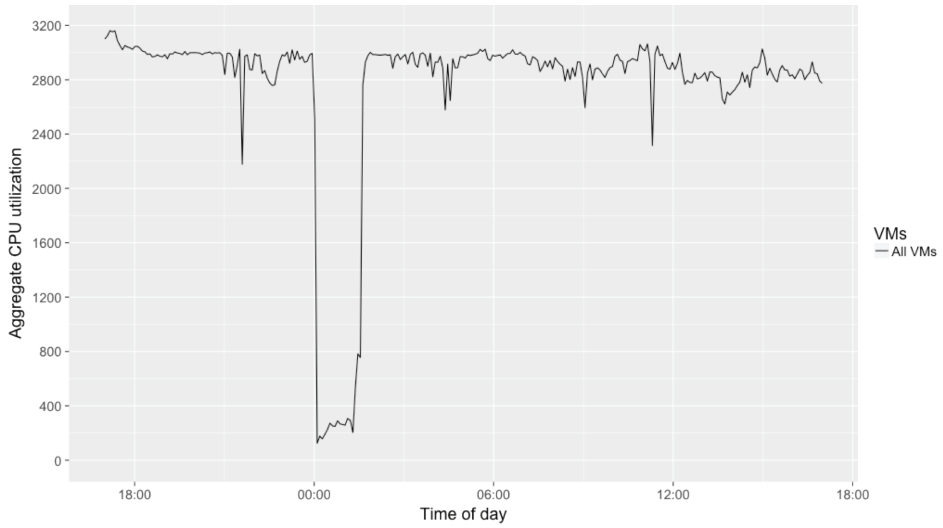

Figure 8. Total CPU Utilization on the Host, Including All VDI VMs and Harvesters

NOTE: The system is now being run at consistently high levels of utilization 24 hours per day when HPC jobs are available to keep the harvesters busy. The drop at midnight is due to temporarily running out of HPC jobs to schedule.

Figure 9 shows utilization data collected at 2-second intervals for about 10 minutes. This zoomed view shows that the harvester utilization closely tracks the VDI utilization in a complementary way: As VDI utilization decreases, the harvesters immediately consume the unused resources; when VDI workload increases again, harvester consumption is reduced to compensate.

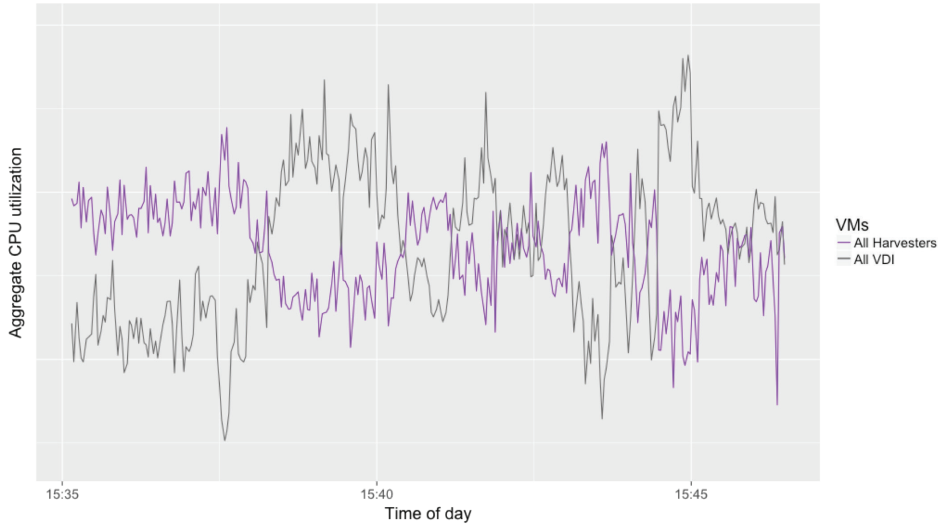

Figure 9. Fine-Grain Comparison of VDI Against Harvester Utilization at 2-Second Intervals

NOTE: The means of each curve have been aligned to make it easier to see how quickly and closely the harvester VMs can absorb unused VDI cycles as the VDI workloads fluctuate.

Because the cycle harvesting is activated even with transient drops in VDI utilization, the Jackson team was concerned that its VDI users' interactivity would be degraded due to this context switching. When the team ran a test and harvesting was enabled, users reported an improvement in responsiveness rather than a degradation. The explanation was simple: When the Jackson team realized that they would be driving hosts at utilization levels when harvesting was enabled, they disabled power management and put their systems into high-performance mode. This prevented the systems from entering CPU idle and sleep states, which typically add latency and can degrade responsiveness for interactive workloads.

Summary

We have described a novel approach that adds significant business value for Jackson National Life Insurance Company through the use of cycle-harvesting techniques that deliver what would otherwise be unused computing capacity for processing HPC workloads. The approach leverages built-in VMware vSphere capabilities to perform very fine-grain sharing of compute resources between VDI and HPC workloads with no degradation for VDI users.

Employing unused on-premises compute capacity has enabled Jackson to avoid the use of expensive public cloud resources for its HPC workloads. Because administering the solution relies solely on built-in vSphere functionality, no new IT skills are required. In addition, the cycle harvesting is transparent to both VDI and HPC users. VDI users reported improved desktop responsiveness when harvesting was implemented.

Perhaps the most transformative outcome of the implementation is that HPC users have been enabled to move beyond a traditional batch-processing mentality by which jobs are submitted one day and results are reviewed the following day. Because cycle harvesting liberates so much computing power, even during the day, users can now start to view their HPC resources as an engine for returning results in real time.

In addition, because the number of compute cycles being harvested significantly eclipses the number available with Jackson's original bare-metal cluster, and because the overall usage experience has been improved, the bare-metal cluster is being retired and the VDI estate will then serve as Jackson's primary HPC infrastructure.

With this success, Jackson National Life Insurance Company is now exploring ways to expand its approach to include cycle harvesting from its thin desktop systems and how best to similarly leverage the vast number of cycles locked within its GPU infrastructure.

Contributors

Wayne Longcore is Vice President of IT Client Services for Jackson National Life Insurance Company, the U.S. subsidiary of London-based Prudential plc. In this role, Wayne oversees all areas of engineering, strategy, and support of the company's end-user computing environment, including thousands of virtual desktops, laptops, printers, and software applications and their associated support servers across Jackson and its affiliates. Before joining Jackson, Wayne worked on multiple international assignments for various organizations, including IBM Research and Development, Consumers Energy, and the European headquarters of SAP. He has more than 30 years of experience in technology and management. Wayne graduated from Michigan State University with a bachelor's degree in computer science and minors in economics and electrical engineering.

Josh Simons is Chief Technologist for High Performance Computing in the Office of the CTO at VMware. He currently leads the effort to bring the value of virtualization to HPC. Previously, he was a Distinguished Engineer at Sun Microsystems, focusing on HPC, and worked at Thinking Machines Corporation. Josh has a bachelor's degree in engineering from Harvard College and a master's degree in computer science from Harvard University. He currently serves on the OpenMP board of directors.

Charu Chaubal is Director of Technical Marketing in the Cloud Platform business unit at VMware. He has been with VMware for more than 11 years and has managed teams responsible for customer education and sales enablement for various technologies such as hypervisor security, virtual networking, hyper-converged storage, and big data. Previously, he worked at Sun Microsystems, where he had more than 7 years of experience in architecting distributed resource management and in grid infrastructure software solutions. Charu holds several patents in the fields of data center automation and numerical price optimization. He received a bachelor's degree in engineering from the University of Pennsylvania and a PhD from the University of California at Santa Barbara, where he studied theoretical models of complex fluids.

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

3/18