GPU as a Service

Abstract

Many teams and individuals at VMware require access to GPUs (Graphics Processing Units). For the VMware Machine Learning and Artificial Intelligence (ML/AI) community, GPUs are needed for training and inference of ML/AI models. For some other teams, GPUs are needed to test and develop VMware products that work with GPUs.

There are primarily two ways in which these teams can get access to GPUs –

- By buying dedicated hardware or by leasing servers with GPUs from HaaS (Hardware as a Service).
- By using GPUs in the Public Cloud.

GPUs are specialized hardware resources and require custom setups – specialized servers, higher power requirements and custom networking setups to name a few. Overall, they are expensive to be individually owned. While Public Clouds like AWS, Google Cloud and Azure provide easy access to GPUs, there are significantly higher costs that come with this flexibility, amongst some other challenges.

In this paper, we describe how we can overcome these limitations by pooling the already procured GPUs and sharing them across Business Groups by leveraging the vGPU features in VMware vSphere. We show how GPU-as-a-Service is more than 14x cheaper than AWS by comparing the machine costs for GPU-as-a-Service with VMware's actual spend in AWS for equivalent GPU machines during a 60-day period. And by sharing GPUs across users, teams and use cases to maintain higher average utilization levels, we can make owning GPUs a more economical and viable option for VMware.

Introduction

There are real challenges for a Data Scientist and a Software Engineer at VMware today to get access to a GPU. Unless there is an actual business justification, it's extremely cumbersome to get approvals to procure servers with GPUs or to use GPUs in AWS, Google Cloud or Azure.

A little over a decade ago, Software Engineers at VMware faced similar challenges with getting access to commodity servers or Virtual Machines (VM), until Nimbus (VMware's Private VM Cloud), made it extremely easy to get unfettered access to VMs, without any justifications. More recently, in 2015, there were similar challenges with getting access to a container that environment that were removed by DECC (Distributed Elastic Container Cloud), which is the largest and longest running Kubernetes service at VMware today, by providing easy access to Kubernetes Namespaces. Both Nimbus and DECC also made business sense because they keep the overall costs low by drastically increasing the utilization of the underlying hardware resources and leverage VMware technology and products to do so.

The vision of GPU-as-a-Service is to do for GPUs, what Nimbus did for VMs and what DECC did for containers. We achieve this by pooling the existing, unutilized GPU machines and making them available to individuals and teams through services like DECC and Nimbus. GPU-as-a-Service leverages the vGPU concept of vSphere that uses NVIDIA GRID technology to provide virtualized GPUs to VMs and containers.

Table of Contents

1. Background

VMware's investments in the ML/AI space have been increasing to align with the trends seen in the technology industry, and with a vision of the "Self-driving Datacenter". The ML/AI community at VMware is constantly working to discover different areas of the business that could benefit by applying this breakthrough technology. And to encourage innovation in this area, it's important that their roadblocks are cleared so they can focus on the actual problem at hand.

For some other teams, GPUs are needed to test and develop VMware products that work with GPUs.

There are primarily two ways to get access to GPUs at VMware –

- By buying dedicated hardware or by leasing servers from Hardware-as-a-Service (HaaS).
- By using GPUs in the Public Cloud.

2. Available options and their limitations

In this section, we discuss some of the existing and potential options along with their advantages and disadvantages.

2.1 Public Clouds

Public Clouds like AWS, Google Cloud and Azure are arguably the most popular choice. Because the GPUs are already configured, it takes less time to get started with the task at hand. Additionally, for most practical purposes, there are unlimited hardware resources available.

However, this flexibility does come at a cost. It's cheaper to get started, but as the project matures, and there is on-going need to use GPUs, the costs mount up. As is the case with Public Cloud spend in general, over time, there is a need and desire to consume other unrelated services in the same eco-system which drive the bills even higher.

The other important drawback with using GPUs in the Public Cloud is the associated latency. At VMware, in most cases, the data that is needed for ML/AI is on-prem and maintained in services like SuperCollider. This significantly impacts the performance of workloads.

2.2 Dedicated hardware

Teams at VMware can get access to dedicated hardware either by procuring the GPUs and the servers that must be installed in or by leasing the available GPUs from HaaS, which is a service provided by the Datacenter Operations team in VES (VMware Engineering Services). Procuring hardware requires budget, a strong business justification, and a long list of approvals. Once it's procured, it takes weeks, if not months, to have the hardware configured and ready for use.

While leasing saves procurement time, it does require approvals and time to configure the machines. Machines are also leased for weeks or months at a time and not just for a few hours. The appropriate type of GPUs must also be present in the inventory and not be reserved already.

In both these cases, the hardware is dedicated to a single user or team. And if it's not getting used by them, the servers and GPUs are idling, while they could have been used by someone else in the company.

2.3 BitFusion

BitFusion is a VMware product that is meant to enable shared access to GPUs. It creates a pool of GPU machines (BitFusion cluster) and runs computes tasks on these GPUs by intercepting NVIDIA CUDA calls from the user's workload. This requires the BitFusion FlexDirect packages to be installed on the machine where GPU access is needed. However, BitFusion has some limitations that make it a non-viable option for majority of the users and infrastructure providers.

There is no concept of quota so a single user of a BitFusion cluster can potentially use up all the available GPU resources. This is an infrastructure providers nightmare.

Since the compute tasks and the data to be processed are being sent to the BitFusion cluster from the FlexDirect client over the network, there are high latency costs associated with it. There are significant performance impacts of accessing remote GPUs as against native GPUs, where both data and compute are collocated, which in turn hampers developer productivity.[17]

BitFusion is also not popular or well-adopted in the ML & Data Science community that heavily relies on OSS tools for their workflows. It also requires specialized container images or custom VM setup by the users which is a major roadblock for the Data Scientists that have very limited DevOps skills.

2.4 VMware Tanzu

With the release of vSphere 7 with Tanzu, users can create Kubernetes clusters called Tanzu Kubernetes Grid (TKG) Clusters using VMs with vGPUs. Amongst the options presented here, this is the most viable alternative to GPU-as-a-Service in terms of functionality and "consume-ability" but it still has some limitations when it comes to sharing. [1]

TKG clusters are great for providing dedicated Kubernetes clusters, but most users just need Kubernetes Namespaces and not the complete Kubernetes cluster, with a dedicated Kubernetes control plane. Since Kubernetes is a container orchestration platform, it's easy to share GPU resources across multiple containers, running in different Namespaces. If a vGPU is assigned to a dedicated TKG cluster, that vGPU is no longer available in the pool for another container to use. This impacts overall GPU utilization, like dedicated hardware.

With GPU-as-a-Service, supporting TKG cluster with GPUs would be possible, it just wouldn't make sense to use this approach as the primary mode of consumption for containers.

3. Goals

GPU-as-a-Service aims to:

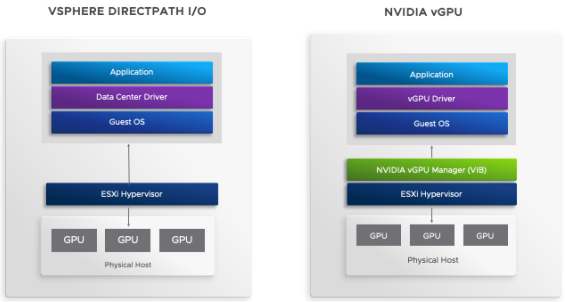
1. Enable unfettered access to GPUs
2. Support GPU access via Kubernetes for Containers
3. Support GPU access via VMs on vSphere
4. Drive hardware utilization higher to keep costs low
5. Use VMware products to support VoV (VMware on VMware)
6. Enable seamless integration for Open-source and ISV (Independent Software Vendor) software

4. Design

4.1 Overview

NVIDIA GRID vGPU technology allows partitioning a single GPU card into multiple smaller vGPUs on VMware vSphere [2]. The performance impact of virtualization as compared to GPUs in Passthrough (DirectPath I/O) mode is negligible. However, vGPUs enable features like VMware vmotion for these GPU based workloads that are not possible in passthrough or on bare metal.

Figure 1. Directpath I/O vs. NVIDIA vGPU [2]



We leverage the vGPU support in vSphere and use that as a building block to provide managed Kubernetes clusters with vGPU nodes and as an environment to run VMs with vGPUs using either Nimbus or Calatrava.

The Kubernetes clusters are created and setup similar to the other managed DECC Tanzu based Kubernetes clusters and by using the same code base. TKG clusters from Calatrava can also be supported going forward once more GPUs are added to the service.

To begin with, a SDDC (Software Defined Datacenter) is created using the physical servers with GPUs. This includes setting up the core SDDC components -- ESXi servers, vSphere, networking with NSX-t, storage with vSAN. The NVIDIA AI Enterprise software (formerly known as NVIDIA vGPU Manager or NVIDIA GRID Drivers) is installed on the GPU servers running ESXi. At this point, the SDDC is ready to be used to create VMs with vGPUs.

NVIDIA GRID requires setting up a license server system called the NVIDIA License System (NLS) that manages and allocated licenses for the vGPUs. NVIDIA licenses can be added to the NLS and it will allocate the appropriate licenses to the VMs as they come up. This is a one time setup.

When the VMs are created, the NVIDIA GRID Linux drivers are installed and the "nvidia-gridd" Linux service is configurated to point to the NLS for licenses as needed. Like setting up any hardware device, installing and configuring the GPU drivers is a required step to be able to use the hardware on the machine. In this case, the only additional step is for licensing.

We use the above approach to make GPUs available in services like DECC and Nimbus.

4.2 DECC Kubernetes clusters (Containers)

DECC is the first existing service at VMware to integrate with GPU-as-a-Service.

In its current form, due to only a limited number of physical GPUs being available in GPU-as-a-Service, they are all being consumed via DECC. More than 200 users have accessed GPUs via GPU-as-a-Service with DECC and it continues to provide GPU access to many VMware employees and teams every single day.

4.2.1 Setup

- Use DECC Ansible automation which leverages Kubeadm to create VM nodes with vGPUs, install NVIDIA GPU drivers, setup Kubernetes and the usual DECC addons and services. Node deployment is currently being enhanced to use Kubernetes Cluster API for vSphere along with Kubernetes Cluster Autoscaler, to support adding a Kubernetes node with vGPU when there is an actual container requesting a NVIDIA GPU and that's waiting to get scheduled due to unavailability of resources.
- Setup the GPU specific components that enable functionality on Kubernetes like GPU discovery, GPU scheduling, metrics collection for GPUs, etc. also using the Ansible automation.
- Kubernetes supports quota for GPUs but it only does so for the number of GPUs and not based on the type or size of GPU. For this reason, we add a Kubernetes admission control policy using Gatekeeper or OPA Admission Controller. This is used to enforce restrictions like what Kubernetes Pods in a given namespace can use which kind of GPUs. For example, a user namespace that has been restricted to use 1x 8GB GPU should not be able to use 1x 32GB GPU. Without this additional Admission Control policy, and with only Kubernetes ResourceQuotas in place, if a user's Namespace had quota for 1 GPU, they could use a GPU of any memory size, not just 8GB.

4.2.2 Usage

- Similar to other DECC clusters, users are able to request a Kubernetes Namespace on the DECC cluster and apart from the usual Compute and Storage requirements they also provide information on how many GPU resources they would like access to and of what kind. Today this is done via the DevHub UI and the DECC micro-frontend [5] has been enhanced to support the GPU resource type, apart from Memory, CPU and Storage resource types.
- The Kubernetes ResourceQuota that is used for Compute is also used for enforcing quota on GPUs. GPU resources can be over allocated (that is, more quota can be allocated than there are vGPUs available in the cluster) but cannot be over-provisioned (at any given time, no more than the total available vGPUs can be used in containers).
- Users can now create Pods in their namespace exactly the same way as it is recommended in the Kubernetes documentation by specifying the necessary GPU resources.
- If users are not using their GPU quota, the GPUs are available to other users in another Namespace on the same cluster or another DECC cluster.

4.3 Nimbus (Virtual Machines)

4.3.1 Setup

- As with any driver, NVIDIA GPU Linux driver versions should match with the drivers that are installed on the ESXi hosts. The compatibility matrix for various drivers is already available from the Graphics team at VMware.
- The SDDC for GPU-as-a-Service described in Section 4.1 above is similar to what a Nimbus vSphere Pod looks like, with the only major difference being the servers have GPUs and they support running VMs with vGPU devices.

4.3.2 Pre-configured VM templates

- Nimbus commands support creating VMs from OVF templates. These templates can be created by users or by the Nimbus team and are hosted on a Nimbus template NFS share that's accessible via HTTP. Nimbus users already do this for most of the other VMs that are deployed on Nimbus.
- We create the necessary VM templates for the supported vGPU profiles. Existing Nimbus commands (nimbus-ovfdeploy or nimbus-worker-deploy) can then be used to deploy these VMs. DRS will automatically take care of placing the VMs on the appropriate physical hosts that can support the requested profile.
- It's not unusual for Nimbus to have a dedicated command for deploying a VM of a particular type (ex. Nimbus-esxdeploy, nimbus-loginsight-deploy, etc.). It would be very easy to add a dedicated deploy command called nimbus-gpu-deploy to take the necessary parameters as command-line options (ex. GPU device count, VM template, vGPU Profile, etc.) and create and setup a VM with the necessary drivers with a single command.
- The only critical requirement for Nimbus scripts to easily support creating VMs with vGPU devices attached is 'rbvmomi' support. The DECC integration already uses rbvmomi to attach vGPU devices to DECC nodes so this should not be an issue. Both pyVmomi and govc also support this.

4.3.3 GPU VMs without using templates

- While using pre-configured templates is more common and similar to how users can create GPU VMs in the Public clouds, this approach may not work for some users and teams (ex: BitFusion product team for dev/testing the product)
- In this case, Nimbus need to simply publish the NVIDIA GPU Driver version running on the ESXi hosts in the Nimbus Pod and provide access to the corresponding Linux drivers.
- Users can then use Nimbus testbed configuration to auto-install these drivers as part of Nimbus postBoot or do it manually or via custom automation depending on their use case.

4.3.4 Usage

- Users create VMs using Nimbus commands like they are used to but in this case they are able to attach vGPU devices to those VMs.
- Nimbus Policy framework can be used to select vGPU enabled Nimbus pods for such workloads. This is similar to how users

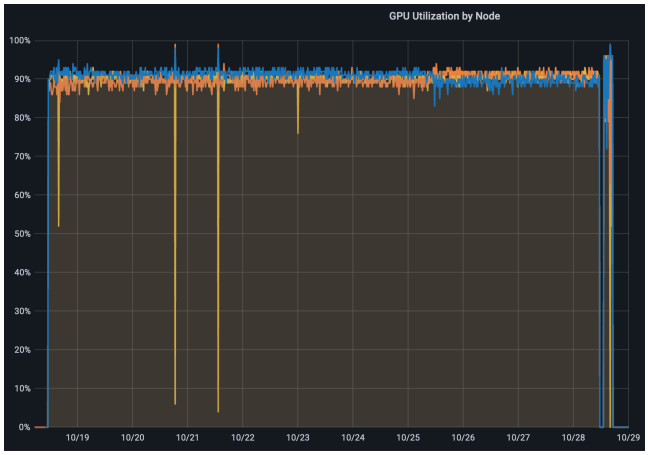
are able to select pods that support NSX-t, vSAN, IPv6, etc. today.

4.4 Monitoring

We use Prometheus, AlertManager and Grafana to monitor usage of the GPUs from vSphere. The open source Prometheus Exporter for VMware vSphere exposes some useful physical GPU metrics for GPU temperature, power consumption in watts, fan speed, etc. to show the health of the GPU servers.

Additionally, NVIDIA has an open-source DCGM Exporter tool that we use to collect GPU utilization metrics in the Prometheus format. The GPU memory utilization metric shows what percentage of the GPU Memory is actively being used [9]. These tools and metrics, can help identify and alert on idling GPU resources so they can be reclaimed for higher priority workloads if needed.

Figure 2. 3x GPUs on a node nearly 100% utilized for ~10 days



5. Evaluation

5.1 Costs

In this section, we compare the cost of using GPUs from Amazon AWS to that of GPU-as-a-Service. While AWS provides a variety of instance types with GPUs, in our evaluation we focus on the "p3" EC2 instance type.

There are two reasons for this choice:

1. This was the most requested instance type based on VMware's AWS account data available via CloudHealth [4][8]
2. It uses the same Tesla V100 GPU device that is currently available in GPU-as-a-Service. [10]

5.1.1 Amazon AWS EC2

Table 1. Amazon AWS P3 EC2 Instance types

Instance	GPUs	vCPU	Mem (GiB)	GPU Mem (GiB)
p3.2xlarge	1	8	61	16
p3.8xlarge	4	32	244	64
p3.16xlarge	8	64	488	128

Table 2. Amazon AWS P3 EC2 Instance costs - US West (Oregon)

Instance	GPUs	GPU Mem (GiB)	On-demand Hourly Rate	Spot Price Hourly
p3.2xlarge	1	16	$3.06	$0.918
p3.8xlarge	4	64	$12.24	$3.672
p3.16xlarge	8	128	$24.48	$8.0091

5.1.2 GPU-as-a-Service hardware cost from HaaS

On-prem hardware costs from HaaS for the GPUaaS servers is $0.70/GB of memory (RAM) per month. This is also the proposed base rate for HaaS servers for Managed SDDCs.

At the moment, there is no surcharge for specialized hardware (like GPUs) to avoid complexity and also since there are very limited GPU servers as compared to the rest of the HaaS server inventory.

5.1.3 VMware Spend in AWS for GPUs

Based on the data available from CloudHealth [4], between 3/22/2022 & 5/21/2022 (~60 days), VMware spent $25,958.69 in AWS for EC2 instances with GPUs. Out of this, $14,724.98 was spent on the NVIDIA Tesla V100 GPUs we currently have in GPU-aaS (p3.2xlarge & p3.8xlarge EC2 instance types from Table 1).

5.1.4 AWS vs. GPUaaS Cost Comparison

Potential HaaS costs for a single GPU ESXi server that's part of a Managed SDDC would be $270.20/month ($0.70 * 386GB). So, the hourly rate for this server is $0.375.

This server has 3x NVIDIA Tesla V100 32GB GPUs (total GPU memory = 96GB) which compares to 1.5x p3.8xlarge instances from AWS since those have the same NVIDIA Tesla V100 GPUs but with 8x NVIDIA Tesla V100 16GB GPUs (total GPU memory = 64GB).

The on-demand hourly rate for AWS p3.8xlarge instances in US West (Oregon) is $3.672. (See Table 2).

5.1.5 Bottom line

A 1.5x p3.8xlarge instance would cost $3.672*1.5 = $5.508/hour. And the equivalent GPU-aaS cost for a similar instance (either as a Nimbus VM or DECC container) is $0.375/hour which is **~14.7x cheaper than AWS!**

Based on the above calculations and VMware AWS costs for p3 instances between 3/22/2022 & 5/21/2022, GPUaaS could have potentially saved VMware $13,723.28 in AWS costs for GPU resources. ($14,724.98 - ($14,724.98 / 14.7) = $13,723.28). [14]

5.2 Utilization

- Hundreds of users in-directly consume GPU-as-a-Service today through projects like InstaML Hub [12] (formerly Jupyter Notebook as a Service), services like vAQA (VMware's Automated Question Answering Service) [11], or directly as DECC Namespace users (for example, the learning group for Quantum Computing at VMware[13]).
- More recently, a second SDDC with GPUs that belonged to the SuperCollider team was transferred over to GPU-as-a-Service so it can be shared by SuperCollider and athe other

users of GPU-aaS. This second SDDC, also has the same Tesla V100 GPUs and is in a different region, which enables GPUaaS to serve an even wider user base and also provide capacity in case of a Disaster Recovery (DR) event.

- GPUaaS now has 5 ESXi hosts with GPUs each in the Santa Clara Datacenter (SCDC) and 8 ESXi hosts with 3 GPUs each in Wenatchee Datacenter (WDC). That's a total of 15 physical GPUs in SCDC and 24 physical GPUs in WDC.

- Similar to how using a bare metal server with 32GB RAM for a workload that only needs 2GB is an overkill and waste of resources, likewise, using a 32GB vGPU profile for a training job that only needs 8GB is waste of resources. GPU-as-a-Service supports different GPU sizes and enable users to select the appropriate vGPU profile that meets the requirements of their workloads to reduce wastage of resources.

- In order to get the GPU utilization to as close to 100% almost all the time, the platform must enable efficient and aggressive time-slicing for GPU access. And while this document highlights GPU access via Containers and VMs, utilization can be increased further at the ML framework level using tools like KubeFlow which are extremely popular in the community and integrate well with serverless tools like Knative to make access to GPUs even less wasteful. KubeFlow and Knative only need Kubernetes to run, which is already supported with GPU-as-a-Service.

6. Limitations

GPU-as-a-Service is still in its early phase so there are known limitations to the service. Some of these are due to the vGPU support in vSphere and some others are from the service itself. They are listed below.

6.1 NVIDIA GRID with vSphere

With every new release of vSphere, the vGPU support is also improving. However, as of today, there are some limitations with vSphere and NVIDIA GRID vGPU that this platform must work around.

- A physical GPU card can only be partitioned into a single vGPU profile type. For example, a NVIDIA Tesla v100 32GB [10] card can only be partitioned as 8x 4GB, 4x 8GB, 2x 16GB or 1x 32GB C-series vGPU profiles. It cannot be a mix of different profiles. This leads to some fragmentation of GPUs.

- VMs can only support 1 vGPU if it is partitioned to smaller memory profiles. Using the example above, it's possible to have 2+ vGPUs with 32GB profiles (full GPU memory) attached to the same VM which is the same as using 2 physical GPUs of 32GB each. However, vSphere doesn't support adding 2x vGPUs with 16GB profiles.

6.2 Platform Limitations

In case of vGPUs on Kubernetes, if all vGPUs of a particular profile are in use and a user who is not completely utilizing their quota tries to create a Pod then this request will wait in Pending state until a GPU resource becomes available again.

Similarly, in case of vGPU VMs, DRS will fail to schedule the VM because there are no vGPUs available. This is similar to a "No Pods available" error in Nimbus, where users have quota but there isn't enough capacity to meet the demands.

This can be mitigated by adding more resources, or by enforcing a lease/TTL for GPU Pods in the cluster so users do not hang on to the GPU resource forever.

7. Future Work

At the moment, majority of the demand for GPUs requires GPU access via containers or frameworks (like KubeFlow, etc.) running on Kubernetes. For this reason, GPU-as-a-Service is currently focusing on enhancing the DECC integration by enabling GPU node deployments using Cluster API for vSphere and using Cluster Autoscalar to make GPU nodes available to all DECC clusters, even those that are not primarily comprised of GPU nodes. DECC uses NSX-t and Antrea overlay networking. So, the nodes for a DECC Kubenetests cluster can come from multiple SDDCs (GPU and non-GPU).

Hybrid DRS that is used for VM placement decisions in Nimbus does not support GPU based scheduling yet and we would need to add that if there are more than a few SDDCs with GPU servers.

Like DECC and Nimbus, GPU-as-a-Service doesn't have chargeback. However, we intend to add cost show-back soon.

There are some new efforts in VMware to better integrate ML/AI workloads in the VMware product ecosystem. One such project is Project Peach [19]. Since GPU-as-a-Service is built using core VMware and NVIDIA technology, it will be useful to have Project Peach validate GPUaaS as a viable integration option so it can benefit both the VMware ML/AI community as well as VMware customers who are looking to build an on-premise ML/AI platform using VMware products.

Finally, the

8. Conclusion

In this paper, we described how existing GPUs in VMware can be centrally managed and pooled together to provide access to these resources from a wider user community, instead of only a few users. We presented calculations on how this approach increases utilization and reduces costs by more than 14x as compared to equivalent offering from Amazon AWS.

9. References

[1] Accelerating Workloads on vSphere 7 with Tanzu - A Technical Preview of Kubernetes Clusters with GPUs. VMware Blog, Justin Murray. https://core.vmware.com/blog/accelerating-workloads-vsphere-7-tanzu-technical-preview-kubernetes-clusters-gpus

[2] Virtual GPUs and Passthrough GPUs on VMware vSphere – Can They be Used Together? VMware Blog, Justin Murray. https://core.vmware.com/blog/virtual-gpus-and-passthrough-gpus-vmware-vsphere-%E2%80%93-can-they-be-used-together

[3] HaaS – Hardware as a Service https://confluence.eng.vmware.com/display/HHP/HaaS+Home+Page

[4] CloudHealth https://confluence.eng.vmware.com/display/CHT/CloudHealth

[5] DECC micro-frontend in DevHub https://devhub.eng.vmware.com/decc

[6] Scheduling GPUs on Kubernetes
https://kubernetes.io/docs/tasks/manage-gpus/scheduling-gpus/

[7] NVIDIA Linux drivers on vSphere
https://wiki.eng.vmware.com/IDEsxGpu/2020/Drivers

[8] CloudHealth Instance Costs from March 22, 2022 – May 25, 2022
https://jira.eng.vmware.com/secure/attachment/2286236/2286236_InstanceCosts--2022-05-23T23+20+25.750Z.csv

[9] NVIDIA GPU metrics
https://docs.nvidia.com/datacenter/dcgm/latest/dcgm-api/group__dcgmFieldIdentifiers.html#group__dcgmFieldIdentifiers_1gfbfcf61a6530f90fe230f2084c4b1641

[10] NVIDIA Tesla V100 vGPU types
https://docs.nvidia.com/grid/latest/grid-vgpu-user-guide/index.html#vgpu-types-tesla-v100-pcie-32gb

[11] vAQA
https://confluence.eng.vmware.com/pages/viewpage.action?spaceKey=RAIL&title=vAQA

[12] InstaML Hub
https://confluence.eng.vmware.com/display/CPF/InstaML+Hub

[13] Quantum Computing at VMware learning group
https://confluence.eng.vmware.com/display/QUANT/Quantum

[14] GPUaaS vs. AWS Costs
https://confluence.eng.vmware.com/x/OxG6U

[15] Kubernetes Cluster API Provider for vSphere
https://github.com/kubernetes-sigs/cluster-api-provider-vsphere

[16] Kubernetes Cluster Autoscalar
https://github.com/kubernetes/autoscaler/tree/master/cluster-autoscaler

[17] BitFusion vs. Native GPU Benchmarking
https://confluence.eng.vmware.com/pages/viewpage.action?pageId=601244074

[18] Prometheus VMware exporter
https://github.com/pryorda/vmware_exporter

[19] Project Peach
https://confluence.eng.vmware.com/display/VCP/Project+Peach