GRID VIRTUAL GPU

DU-06920-001 | August 2015

User Guide

DOCUMENT CHANGE HISTORY

DU-06920-001

Version	Date	Authors	Description of Change
0.3	7/1/2013	AC	Initial release for vGPU private beta
0.9	9/1/2013	AC	Updated for vGPU Tech Preview.
0.95	11/8/2013	AC	vGPU Tech Preview R2
1.0	12/13/2013	AC	vGPU 1.0 RTM
1.1	3/31/2014	AC	vGPU 1.1 RTM
1.15	7/1/2014	AC	vGPU 1.2 Tech Preview
1.2	9/30/2014	AC	vGPU 1.2 RTM
1.4	5/11/2015	Avi	vGPU 1.4 RTM
1.45	7/10/2015	Avi	vGPU 1.4 RTM R2
2.0	9/1/2015	AC	vGPU 2.0

Chapter 1. Introduction 1
1.1 Architecture 1
1.2 Supported GPUs 3
1.2.1 Virtual GPU types 3
1.2.2 Homogeneous virtual GPUs 4
1.3 Guest OS Support 6
1.3.1 Windows 6
1.3.2 Linux 6
1.4 Features 6
Chapter 2. Getting Started 7
2.1 Citrix XenServer 7
2.1.1 Prerequisites 7
2.1.2 Installing Citrix XenServer and XenCenter 8
2.1.3 Installing the NVIDIA Virtual GPU Manager for XenServer 8
2.1.4 Configuring a VM with Virtual GPU 12
2.1.5 Booting the VM and Installing Drivers 13
2.1.6 Applying a vGPU license 15
2.1.7 Removing a VM's vGPU configuration 15
2.2 VMware vSphere 17
2.2.1 Prerequisites 17
2.2.2 Installing VMware vSphere 17
2.2.3 Installing the NVIDIA Virtual GPU Manager for vSphere 17
2.2.4 Configuring a VM with Virtual GPU 19
2.2.5 Booting the VM and Installing Drivers 21
2.2.6 Applying a vGPU license 22
2.2.7 Removing a VM's vGPU configuration 22
2.2.8 Modifying GPU assignment for vGPU-Enabled VMs 22
2.3 vGPU Licensing on Windows 23
Chapter 3. Using vGPU on Linux 25
3.1 Installing vGPU drivers 25
3.1.1 Prerequisites 25
3.1.2 Running the driver installer 26
3.2 vGPU Licensing on Linux 28
Chapter 4. Performance monitoring 30
4.1 Using nvidia-smi 30
4.2 Using Citrix XenCenter 32
Chapter 5. XenServer vGPU Management 33
5.1 Management objects for GPUs 33
5.1.1 pgpu 33
5.1.2 vgpu-type 35
5.1.3 gpu-group 37
5.1.4 vgpu 37
5.2 Creating a vGPU using xe 38
5.3 Controlling vGPU allocation 38
5.3.1 GPU allocation policy 39

5.3.2 Determining the physical GPU that a virtual GPU is resident on 40
5.3.3 Controlling the vGPU types enabled on specific physical GPUs 41
5.3.4 Creating vGPUs on specific physical GPUs 43
5.4 Cloning vGPU-enabled VMs 45
5.5 Using GPU pass-through 45
Chapter 6. XenServer Performance Tuning 48
6.1 XenServer tools 48
6.2 Using remote graphics 48
6.2.1 Disabling console VGA 49
6.4 Allocation strategies 50
6.4.1 NUMA considerations 50
6.4.2 Maximizing performance 51
Chapter 7. Troubleshooting 52
7.1 Known issues 52
7.2 Troubleshooting steps 52
7.2.1 Verify the NVIDIA kernel driver is loaded 52
7.2.2 Verify that nvidia-smi works 53
7.2.3 NVIDIA kernel driver output 53
7.2.4 GRID Virtual GPU Manager messages 53
7.3 Filing a bug report 55
7.3.1 nvidia-bug-report.sh 55
7.3.2 XenServer status report 55
Appendix A. XenServer Basics 57
A.1 Opening a dom0 shell 57
A.1.1 Accessing the dom0 shell via XenCenter 57
A.1.2 Accessing the dom0 shell using ssh 58
A.2 Copying files to dom0 58
A.2.1 Copying files using scp 58
A.2.2 Copying files via a CIFS-mounted filesystem 59
A.3 Determining a VM's UUID 59
A.3.1 Using xe vm-list 60
A.3.2 Using XenCenter 60
A.4 Using more than two vCPUs with Windows client VMs 61
A.5 Pinning VMs to a specific CPU socket and cores 61
A.6 Changing dom0 vCPUs and pinning 63
A.7 Determining GPU locality 64

LIST OF FIGURES

Figure 1 GRID vGPU System Architecture 2
Figure 2 GRID vGPU Internal Architecture.. 2
Figure 3 Example vGPU configurations on GRID K2.. 5
Figure 4 GRID vGPU Manager supplemental pack selected in XenCenter 10
Figure 5 Successful installation of GRID vGPU Manager supplemental pack............. 11
Figure 6 Using XenCenter to configure a VM with a vGPU 13
Figure 7 NVIDIA driver installation in the guest VM... 14
Figure 8 Verifying NVIDIA driver operation using NVIDIA Control Panel 15
Figure 9 Using XenCenter to remove a vGPU configuration from a VM.................... 16
Figure 10 VM settings for vGPU.. 20
Figure 11 Verifying NVIDIA driver operation using NVIDIA Control Panel................. 21
Figure 12 Configuring vGPU licensing on Windows.. 24
Figure 13 NVIDIA Linux driver installer 26
Figure 14 Update xorg.conf settings 27
Figure 15 Verifying operation with nvidia-settings.. 28
Figure 16 Sample gridd.conf for GRID vGPU 29
Figure 17 Using XenCenter to monitor GPU performance 32
Figure 18 Physical GPU display in XenCenter... 35
Figure 19 Modifying GPU placement policy in XenCenter................................... 40
Figure 20 Editing a GPU's enabled vGPU types using XenCenter 42
Figure 21 Using a custom GPU group within XenCenter 44
Figure 22 Cloning a VM using XenCenter 45
Figure 23 Using XenCenter to configure a passthrough GPU................................ 46
Figure 24 A NUMA server platform 50
Figure 25 Including NVIDIA logs in server status report.................................... 56
Figure 26 Connecting to the dom0 shell via XenCenter..................................... 58
Figure 27 Using XenCenter to determine a VM's UUID 61

LIST OF TABLES

Table 1 GRID K1 and K2 Virtual GPU types 3
Table 2 Tesla M60 and M6 Virtual GPU types 4
Table 3 Virtual GPUs supporting Linux 6
Table 4 Virtual GPUs supporting Linux 25

Chapter 1. INTRODUCTION

NVIDIA GRID™ vGPU™ enables multiple virtual machines (VMs) to have simultaneous, direct access to a single physical GPU, using the same NVIDIA graphics drivers that are deployed on non-virtualized Operating Systems. By doing this, GRID vGPU provides VMs with unparalleled graphics performance and application compatibility, together with the cost-effectiveness and scalability brought about by sharing a GPU among multiple workloads.

This chapter introduces the architecture and features of vGPU. Chapter 2 provides a step-by-step guide to getting started with vGPU on Citrix XenServer and VMware ESXi. Chapter 3 describes using vGPU with Linux VMs. Chapters 4 and 5 cover vGPU management and performance optimization on XenServer, and Chapter 6 provides guidance on troubleshooting.

1.1 ARCHITECTURE

GRID vGPU's high-level architecture is illustrated in Figure 1. Under the control of NVIDIA's GRID Virtual GPU Manager running under the hypervisor, GRID physical GPUs are capable of supporting multiple virtual GPU devices (vGPUs) that can be assigned directly to guest VMs.

Guest VMs use GRID virtual GPUs in the same manner as a physical GPU that has been passed through by the hypervisor: an NVIDIA driver loaded in the guest VM provides direct access to the GPU for performance-critical fast paths, and a paravirtualized interface to the GRID Virtual GPU Manager is used for non-performant management operations.

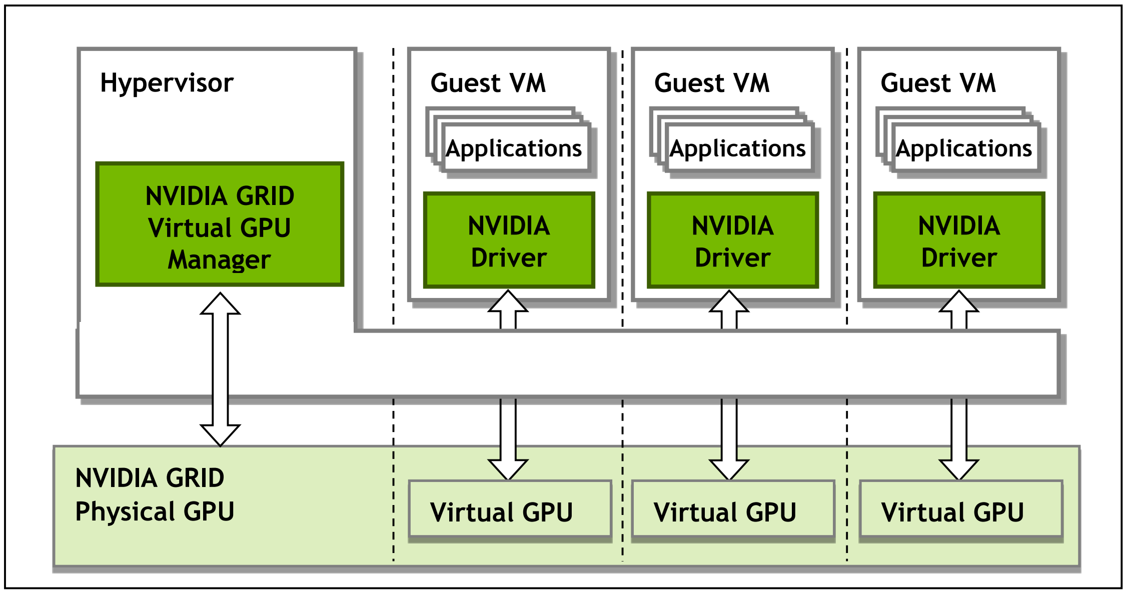


Figure 1 GRID vGPU System Architecture

GRID vGPUs are analogous to conventional GPUs, having a fixed amount of GPU framebuffer, and one or more virtual display outputs or "heads". The vGPU's framebuffer is allocated out of the physical GPU's framebuffer at the time the vGPU is created, and the vGPU retains exclusive use of that framebuffer until it is destroyed.

All vGPUs resident on a physical GPU share access to the GPU's engines including the graphics (3D), video decode, and video encode engines.

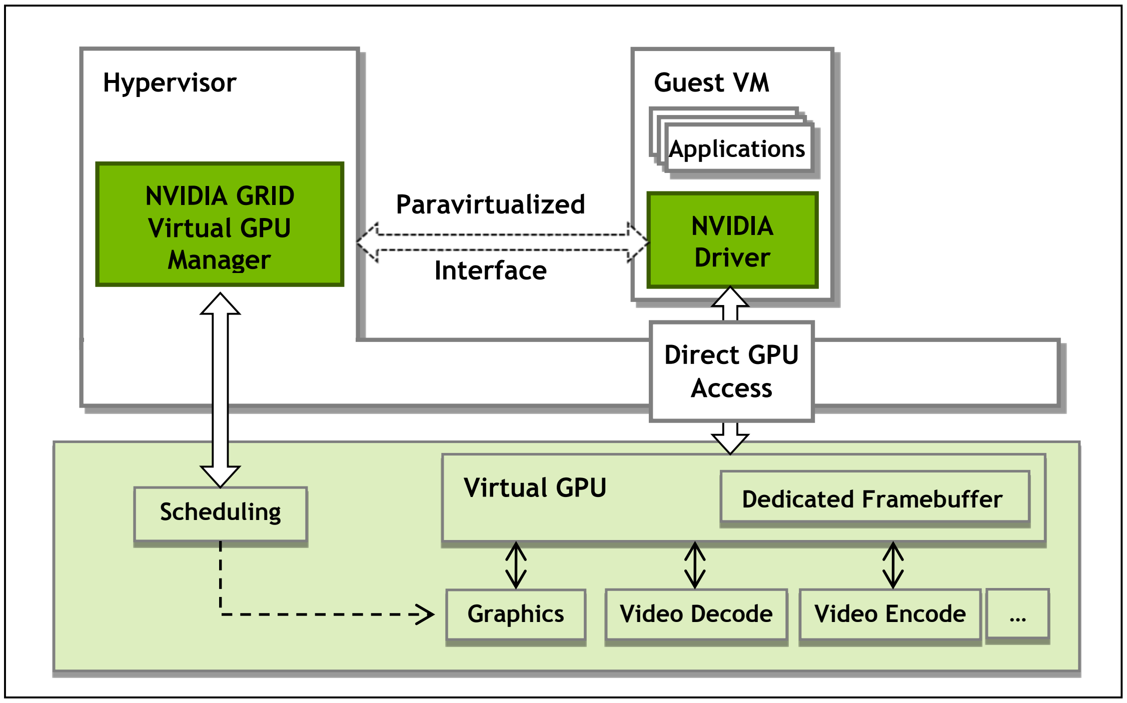


Figure 2 GRID vGPU Internal Architecture

1.2 SUPPORTED GPUS

GRID vGPU is supported on NVIDIA GRID K1, K2, and is available as a licensed feature on Tesla M60, M6. Refer to the release notes for a list of recommended server platforms to use with GRID GPUs.

1.2.1 Virtual GPU types

GRID K1, K2, and Tesla M60 each implement multiple physical GPUs; K2 and M60 have 2 GPUs onboard; GRID K1 has 4 GPUs. Tesla M60 implements a single physical GPU.

Each physical GPU can support several different types of virtual GPU. Virtual GPU types have a fixed amount of framebuffer, number of supported display heads and maximum resolutions, and are targeted at different classes of workload

The virtual GPU types supported by GRID GPUs are defined in Table 1 and Table 2.

Due to their differing resource requirements, the maximum number of vGPUs that can be created simultaneously on a physical GPU varies according to the vGPU type. For example, a GRID K2 physical GPU can support up to 4 K240Q vGPUs on each of its two physical GPUs, for a total of 8 vGPUs, but only 2 K260Qs vGPUs, for a total of 4 vGPUs.

Card	Physical GPUs	GRID Virtual GPU	Intended Use Case	Frame Buffer (Mbytes)	Virtual Display Heads	Max Resolution per Display Head	Maximum vGPUs	
							Per GPU	Per Board
GRID K1	4	K180Q	Power User	4096	4	2560x1600	1	4
		K160Q	Power User	2048	4	2560x1600	2	8
		K140Q	Power User	1024	2	2560x1600	4	16
		K120Q	Power User	512	2	2560x1600	8	32
		K100	Knowledge Worker	256	2	1920x1200	8	32
GRID K2	2	K280Q	Designer	4096	4	2560x1600	1	2
		K260Q	Power User, Designer	2048	4	2560x1600	2	4
		K240Q	Power User, Designer	1024	2	2560x1600	4	8
		K220Q	Power User, Designer	512	2	2560x1600	8	16
		K200	Knowledge Worker	256	2	1920x1200	8	16

Table 1 GRID K1 and K2 Virtual GPU types

Card	Physical GPUs	GRID Virtual GPU	Intended Use Case	Frame Buffer (Mbytes)	Virtual Display Heads	Max Resolution per Display Head	Maximum vGPUs	
							Per GPU	Per Board
Tesla M60	2	M60-8Q	Designer	8192	4	3840x2160	1	2
		M60-4Q	Designer	4096	4	3840x2160	2	4
		M60-2Q	Designer	2048	4	2560x1600	4	8
		M60-1Q	Power User, Designer	1024	2	2560x1600	8	16
		M60-0Q	Power User, Designer	512	2	2560x1600	16	32
		M60-2B	Power User	2048	2	2560x1600	4	8
		M60-1B	Power User	1024	2	2560x1600	8	16
		M60-0B	Power User	512	2	2560x1600	16	32
Tesla M6	1	M6-8Q	Designer	8192	4	3840x2160	1	1
		M6-4Q	Designer	4096	4	3840x2160	2	2
		M6-2Q	Designer	2048	4	2560x1600	4	4
		M6-1Q	Power User, Designer	1024	2	2560x1600	8	8
		M6-0Q	Power User, Designer	512	2	2560x1600	16	16
		M6-2B	Power User	2048	2	2560x1600	4	4
		M6-1B	Power User	1024	2	2560x1600	8	8
		M6-0B	Power User	512	2	2560x1600	16	16

Table 2 Tesla M60 and M6 Virtual GPU types

Note: GRID vGPU is a licensed feature on Tesla M6/M60. A software license is required to enable full vGPU features within the guest VM. For more details, see sections 2.3, 3.2, and the GRID Licensing User Guide.

1.2.2 Homogeneous virtual GPUs

This release of GRID vGPU supports homogeneous virtual GPUs: at any given time, the virtual GPUs resident on a single physical GPU must be all of the same type. However, this restriction doesn't extend across physical GPUs on the same card. Each physical GPU on a K1 or K2 may host different types of virtual GPU at the same time.

For example, a GRID K2 card has two physical GPUs, and can support five types of virtual GPU; GRID K200, K220Q, K240Q, K260Q, and K280Q. Figure 3 shows some example virtual GPU configurations on K2:

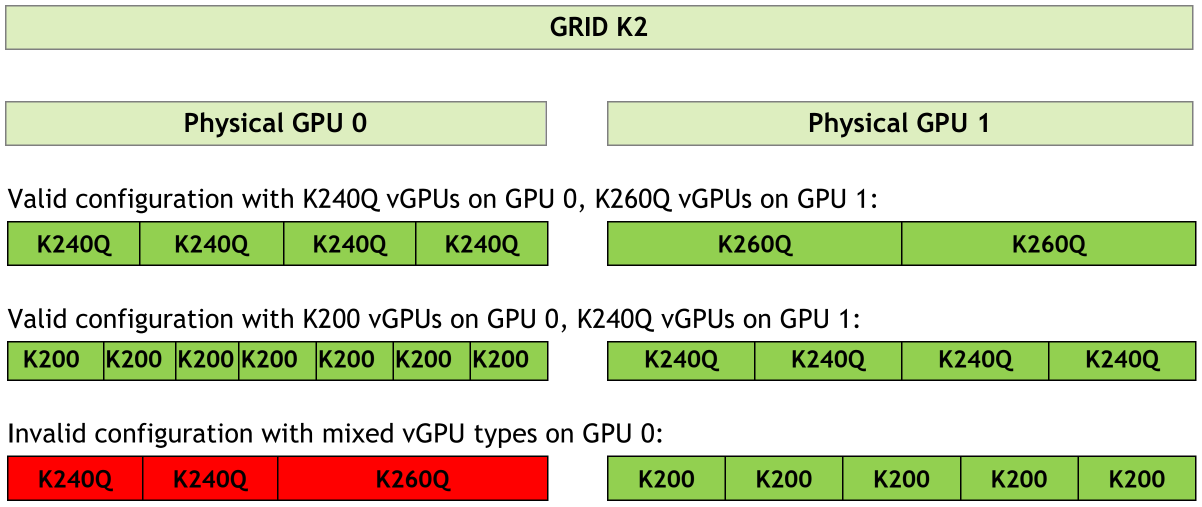

GRID K2
Physical GPU 0
Physical GPU 1
Valid configuration with K240Q vGPUs on GPU 0, K260Q vGPUs on GPU 1:
K240Q
K240Q
K240Q
K240Q
K260Q
K260Q
Valid configuration with K200 vGPUs on GPU 0, K240Q vGPUs on GPU 1:
K200
K200
K200
K200
K200
K200
K200
K240Q
K240Q
K240Q
K240Q
Invalid configuration with mixed vGPU types on GPU 0:
K240Q
K240Q
K260Q
K200
K200
K200
K200
K200

Figure 3 Example vGPU configurations on GRID K2

1.3 GUEST OS SUPPORT

This release of GRID vGPU includes support for the following guest VM operating systems. Refer to the driver release notes for further information on supported configurations:

1.3.1 Windows

- Windows 7 (32/64-bit)
- Windows 8 (32/64-bit), Windows 8.1 (32/64-bit)
- Windows 10 (32/64-bit) – preview / pre-production drivers
- Windows Server 2008 R2
- Windows Server 2012 R2

1.3.2 Linux

64-bit Linux guest VMs are supported on the following virtual GPU types:

Tesla M60	M60-8Q
	M60-4Q
	M60-2Q
	M60-1Q
	M60-0Q

Tesla M6	M6-8Q
	M6-4Q
	M6-2Q
	M6-1Q
	M6-0Q

Table 3 Virtual GPUs supporting Linux

1.4 FEATURES

This release of GRID vGPU includes support for:

- DirectX 9/10/11, Direct2D, and DirectX Video Acceleration (DXVA)
- OpenGL 4.5.
- NVIDIA GRID SDK (remote graphics acceleration).

CUDA and OpenCL are supported on these virtual GPUs:

- GRID M60-8Q, M6-8Q

Chapter 2. GETTING STARTED

This chapter provides a step-by-step guide to booting a Windows VM on Citrix XenServer and VMware vSphere with NVIDIA Virtual GPU.

2.1 CITRIX XENSERVER

These setup steps assume familiarity with the XenServer skills covered in Appendix A

2.1.1 Prerequisites

Before proceeding, ensure that you have these prerequisites:

- NVIDIA GRID K1,K2, or Tesla M6, M60 cards.
- A server platform capable of hosting XenServer and the NVIDIA GRID or Tesla cards. Refer to the release notes for a list of recommended servers.
- The NVIDIA GRID vGPU software package for Citrix XenServer, consisting of the GRID Virtual GPU Manager for XenServer, and NVIDIA GRID vGPU drivers for Windows, 32- and 64-bit.
- Citrix XenServer 6.2 SP1 with applicable hotfixes, or later, obtainable from Citrix.
- An installed Windows VM to be enabled with vGPU.

To run Citrix XenDesktop with virtual machines running NVIDIA Virtual GPU, you will also need:

- Citrix XenDesktop 7.1 or later, obtainable from Citrix.

Note: Earlier versions of Citrix XenServer and XenDesktop are not supported for use with NVIDIA Virtual GPU.

Review the release notes and known issues for GRID Virtual GPU before proceeding with installation.

2.1.2 Installing Citrix XenServer and XenCenter

Install Citrix XenServer and any applicable patches, following Citrix's installation instructions. Install the Citrix XenCenter management GUI on a PC.

2.1.3 Installing the NVIDIA Virtual GPU Manager for XenServer

The NVIDIA Virtual GPU Manager runs in XenServer's dom0. It is provided as a RPM file for XenServer 6.2 SP1 and as a Supplemental Pack for XenServer 6.5 SP1.

Note: there are separate Virtual GPU Manager files for different versions of XenServer. Consult the release notes for guidance on which package to use for each version of XenServer.

2.1.3.1 RPM package installation

The RPM file must be copied to XenServer's dom0 prior to installation (see A.2, Copying files to dom0). Use the `rpm` command to install the package:

```
[root@xenserver ~]# rpm -iv  NVIDIA-vgx-xenserver-6.2-340.57.i386.rpm
Preparing packages for installation...
NVIDIA-vgx-xenserver-6.2-340.57
[root@xenserver ~]#
```

Reboot the XenServer platform:

```
[root@xenserver ~]# shutdown -r now

Broadcast message from root (pts/1) (Fri Dec 6 14:24:11 2013):

The system is going down for reboot NOW!
[root@xenserver ~]#
```

2.1.3.2 Update RPM package

If an existing GRID Virtual GPU Manager is already installed on the system and you wish to upgrade, follow these steps:

- Shut down any VMs that are using GRID vGPU.

- Install the new package using the `-U` option to the `rpm` command, to upgrade from the previously installed package:

```
[root@xenserver ~]# rpm -Uv NVIDIA-vgx-xenserver-6.2-340.57.i386.rpm
Preparing packages for installation...
NVIDIA-vgx-xenserver-6.2-340.57
[root@xenserver ~]#
```

Note: You can query the version of the current GRID package using the `rpm -q` command:

```
[root@xenserver ~]# rpm -q NVIDIA-vgx-xenserver
NVIDIA-vgx-xenserver-6.2-340.57
[root@xenserver ~]#
```

If an existing NVIDIA GRID package is already installed and you don't select the upgrade (-U) option when installing a newer GRID package, the rpm command will return many conflict errors.

```
Preparing packages for installation...

        file /usr/bin/nvidia-smi from install of NVIDIA-vgx-
xenserver-6.2-340.57.i386 conflicts with file from package
NVIDIA-vgx-xenserver-6.2-331.59.i386

        file /usr/lib/libnvidia-ml.so from install of NVIDIA-
vgx-xenserver-6.2-340.57.i386 conflicts with file from package
NVIDIA-vgx-xenserver-6.2-331.59.i386

        ...
```

Reboot the XenServer platform:

```
[root@xenserver ~]# shutdown -r now

Broadcast message from root (pts/1) (Fri Dec 6 14:24:11 2013):

The system is going down for reboot NOW!
[root@xenserver ~]#
```

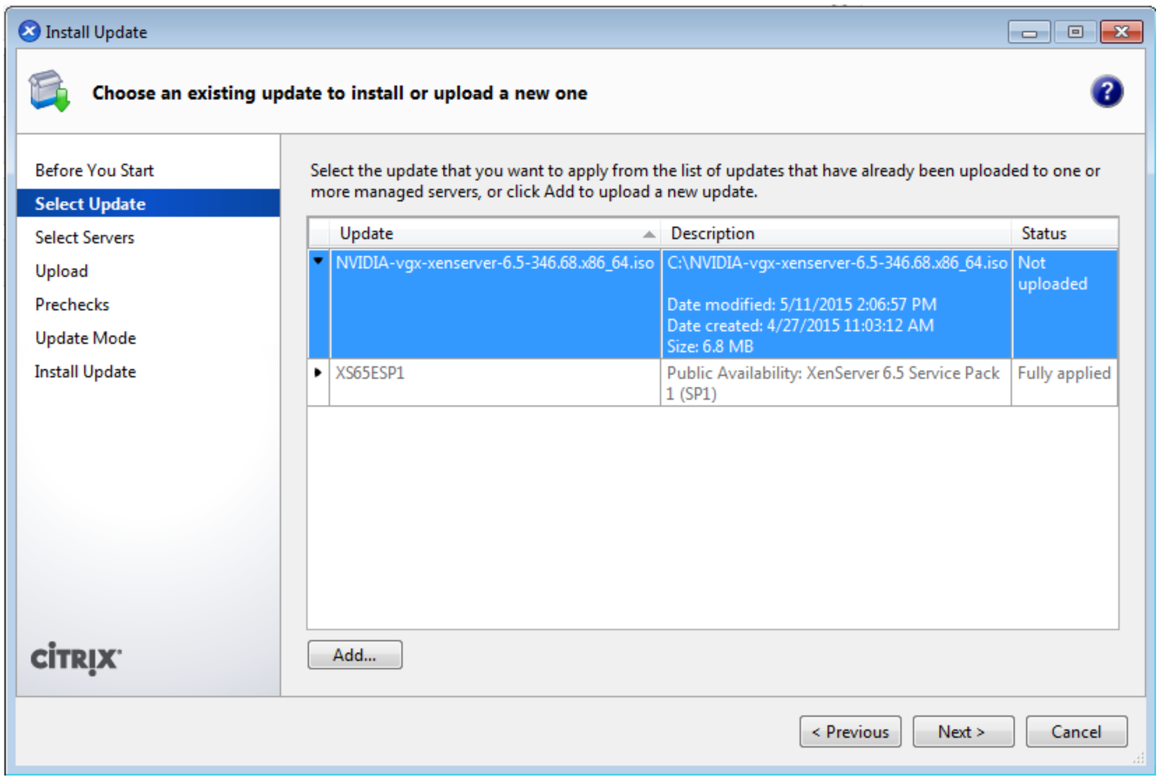
Note: GRID Virtual GPU Manager and Guest VM drivers must be matched from the same release. After updating vGPU Manager, guest VMs will boot with vGPU disabled until their guest vGPU driver is updated to match the vGPU Manager version. Consult the release notes for further details.

2.1.3.3 Supplemental Pack installation / update

XenCenter can be used to install / update Supplemental Packs on XenServer hosts. The NVIDIA GRID Virtual GPU Manager supplemental pack is provided as an ISO.

Note: NVIDIA GRID Virtual GPU Manager supplemental pack installation / update is supported from XenServer 6.5 SP1 and XenCenter version 6.5 (build 6.5.2.2477) onwards.

- Select `Install Update…` from the `Tools` menu
- Click `Next` after going through the instructions on the `Before You Start` section
- Click `Add` on the `Select Update` section and open NVIDIA's XenServer Supplemental Pack ISO



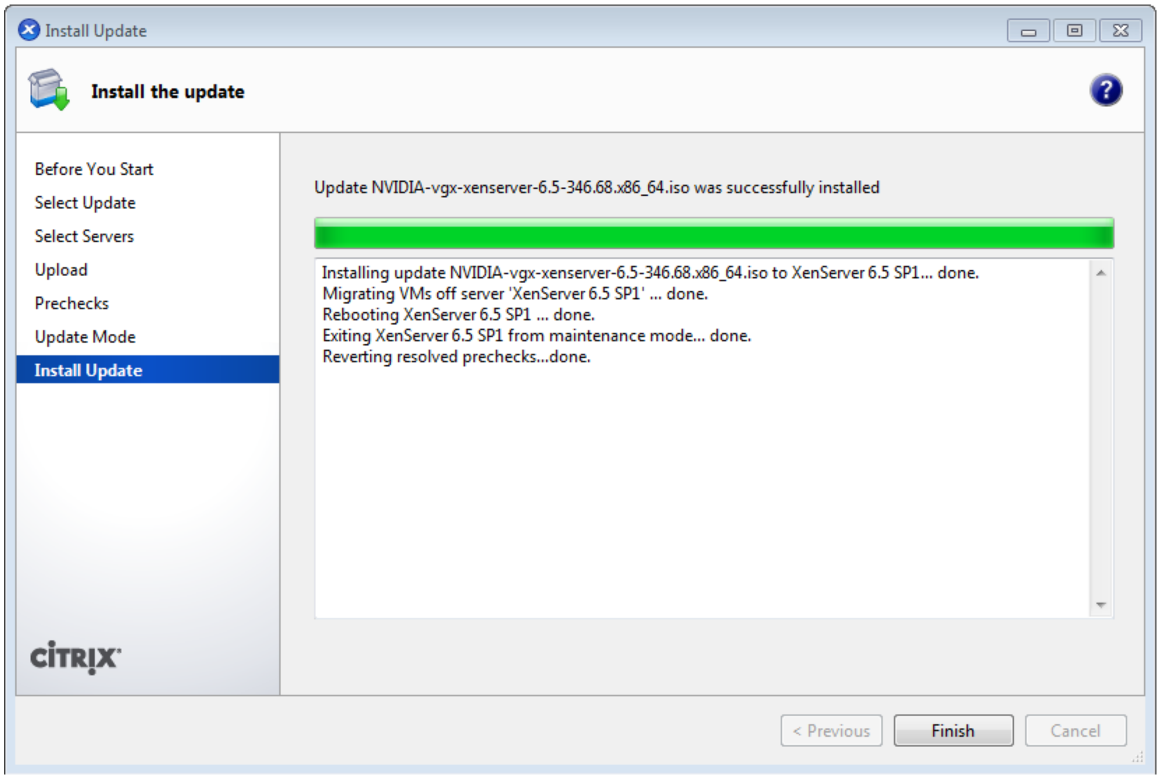
Select the update that you want to apply from the list of updates that have already been uploaded to one or more managed servers, or click Add to upload a new update.

Update	Description	Status
NVIDIA-vgx-xenserver-6.5-346.68.x86_64.iso	C:\NVIDIA-vgx-xenserver-6.5-346.68.x86_64.iso Date modified: 5/11/2015 2:06:57 PM Date created: 4/27/2015 11:03:12 AM Size: 6.8 MB	Not uploaded
XS65ESP1	Public Availability: XenServer 6.5 Service Pack 1 (SP1)	Fully applied

Figure 4 GRID vGPU Manager supplemental pack selected in XenCenter

- Click `Next` on the `Select Update` section
- In the `Select Servers` section select all the XenServer hosts on which the Supplemental Pack should be installed on and click `Next`
- Click `Next` on the `Upload` section once the Supplemental Pack has been uploaded to all the XenServer hosts

- Click `Next` on the `Prechecks` section
- Click `Install Update` on the `Update Mode` section
- Click `Finish` on the `Install Update` section



Update NVIDIA-vgx-xenserver-6.5-346.68.x86_64.iso was successfully installed

```
Installing update NVIDIA-vgx-xenserver-6.5-346.68.x86_64.iso to XenServer 6.5 SP1... done.
Migrating VMs off server 'XenServer 6.5 SP1' ... done.
Rebooting XenServer 6.5 SP1 ... done.
Exiting XenServer 6.5 SP1 from maintenance mode... done.
Reverting resolved prechecks...done.
```

Figure 5 Successful installation of GRID vGPU Manager supplemental pack

2.1.3.4 Verifying installation

After the XenServer platform has rebooted, verify that the GRID package installed and loaded correctly by checking for the NVIDIA kernel driver in the list of kernel loaded modules.

```
[root@xenserver ~]# lsmod | grep nvidia
nvidia               9522927  0
i2c_core               20294  2 nvidia,i2c_i801
[root@xenserver ~]#
```

Verify that the NVIDIA kernel driver can successfully communicate with the GRID physical GPUs in your system by running the `nvidia-smi` command, which should produce a listing of the GPUs in your platform:

```
[root@xenserver ~]# nvidia-smi
```

```
Mon Nov 10 18:46:50 2014
+------------------------------------------------------+
| NVIDIA-SMI 340.57     Driver Version: 340.57         |
|-------------------------------+----------------------+----------------------+
| GPU  Name                     | Bus-Id        Disp.  | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap| Memory-Usage         | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GRID K1                  | 0000:04:00.0     Off |                  N/A |
| N/A   27C    P0    13W /  31W |   0%     9MB / 4095MB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  GRID K1                  | 0000:05:00.0     Off |                  N/A |
| N/A   25C    P0    13W /  31W |   0%     9MB / 4095MB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  GRID K1                  | 0000:06:00.0     Off |                  N/A |
| N/A   21C    P0    13W /  31W |   0%     9MB / 4095MB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  GRID K1                  | 0000:07:00.0     Off |                  N/A |
| N/A   23C    P0    13W /  31W |   0%     9MB / 4095MB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   4  GRID K1                  | 0000:86:00.0     Off |                  N/A |
| N/A   24C    P0    13W /  31W |   0%     9MB / 4095MB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   5  GRID K1                  | 0000:87:00.0     Off |                  N/A |
| N/A   24C    P0    13W /  31W |   0%     9MB / 4095MB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   6  GRID K1                  | 0000:88:00.0     Off |                  N/A |
| N/A   25C    P0    13W /  31W |   0%     9MB / 4095MB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   7  GRID K1                  | 0000:89:00.0     Off |                  N/A |
| N/A   25C    P0    12W /  31W |   0%     9MB / 4095MB |      0%      Default |
+-------------------------------+----------------------+----------------------+


+-----------------------------------------------------------------------------+
| Compute processes:                                               GPU Memory |
|  GPU       PID  Process name                                     Usage      |
|=============================================================================|
|  No running compute processes found                                         |
+-----------------------------------------------------------------------------+
[root@xenserver ~]#
```

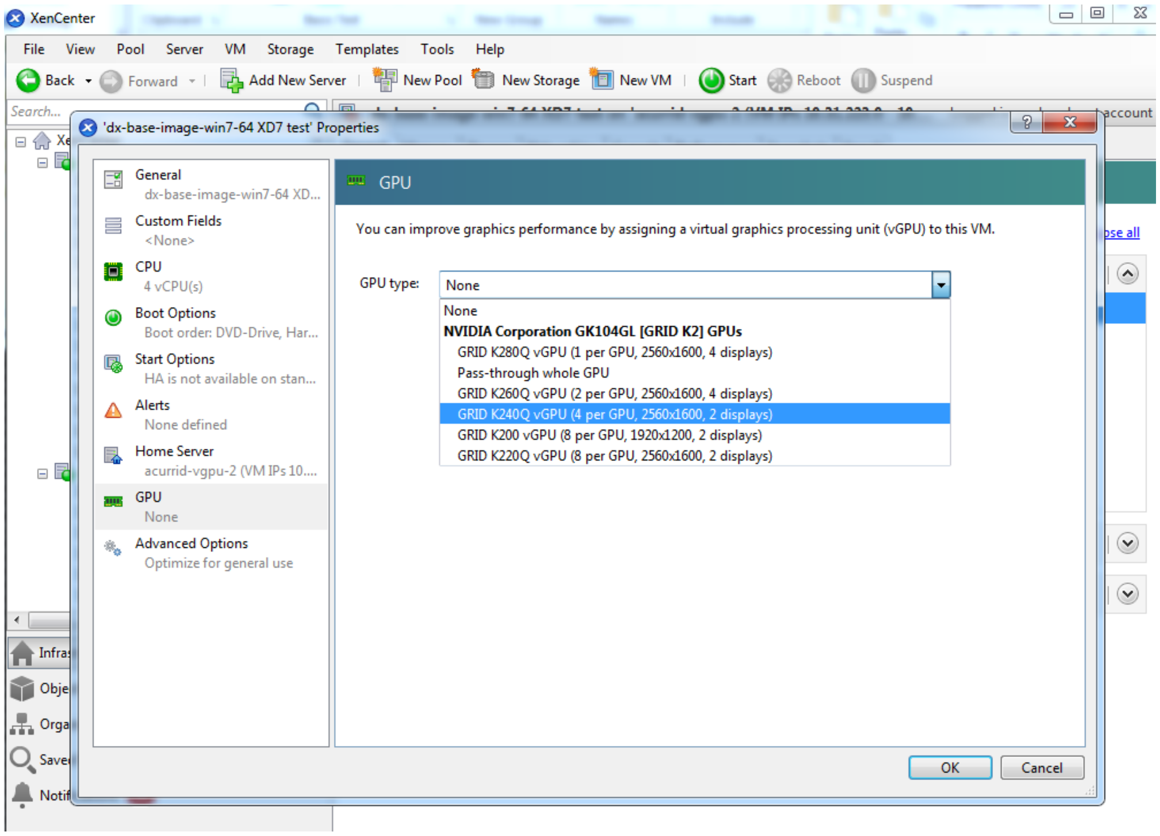
The `nvidia-smi` command is described in more detail in section 4.1.

If `nvidia-smi` fails to run or doesn't produce the expected output for all the NVIDIA GPUs in your system, see Chapter 6 for troubleshooting steps.

2.1.4 Configuring a VM with Virtual GPU

XenServer supports configuration and management of virtual GPUs using XenCenter, or the `xe` command line tool that is run in a XenServer dom0 shell. Basic configuration using XenCenter is described in the following sections. Command line management using `xe` is described in Chapter 6.

To configure a VM to use virtual GPU, first ensure the VM is powered off, then right-click on the VM in XenCenter, select “Properties” to open the VM's properties, and select the “GPU” property. The available GPU types are listed in the GPU type dropdown:


XenCenter
File View Pool Server VM Storage Templates Tools Help
Back Forward Add New Server New Pool New Storage New VM Start Reboot Suspend
Search...
Infras
Obje
Orga
Save
Notif
OK
Cancel

'dx-base-image-win7-64 XD7 test' Properties

General
dx-base-image-win7-64 XD...
Custom Fields
<None>
CPU
4 vCPU(s)
Boot Options
Boot order: DVD-Drive, Har...
Start Options
HA is not available on stan...
Alerts
None defined
Home Server
acurrid-vgpu-2 (VM IPs 10....
GPU
None
Advanced Options
Optimize for general use

GPU

You can improve graphics performance by assigning a virtual graphics processing unit (vGPU) to this VM.

GPU type: None

None
NVIDIA Corporation GK104GL [GRID K2] GPUs
GRID K280Q vGPU (1 per GPU, 2560x1600, 4 displays)
Pass-through whole GPU
GRID K260Q vGPU (2 per GPU, 2560x1600, 4 displays)
GRID K240Q vGPU (4 per GPU, 2560x1600, 2 displays)
GRID K200 vGPU (8 per GPU, 1920x1200, 2 displays)
GRID K220Q vGPU (8 per GPU, 2560x1600, 2 displays)

Figure 6 Using XenCenter to configure a VM with a vGPU

2.1.5 Booting the VM and Installing Drivers

Once you have configured a VM with a vGPU, start the VM, either from XenCenter or by using `xe vm-start` in a dom0 shell.

Viewing the VM’s console in XenCenter, the VM should boot to a standard Windows desktop in VGA mode at 800x600 resolution. The Windows screen resolution control panel may be used to increase the resolution to other standard resolutions, but to fully enable vGPU operation, as for a physical NVIDIA GPU, the NVIDIA driver must be installed.

- Copy the 32- or 64-bit NVIDIA Windows driver package to the guest VM and execute it to unpack and run the driver installer:

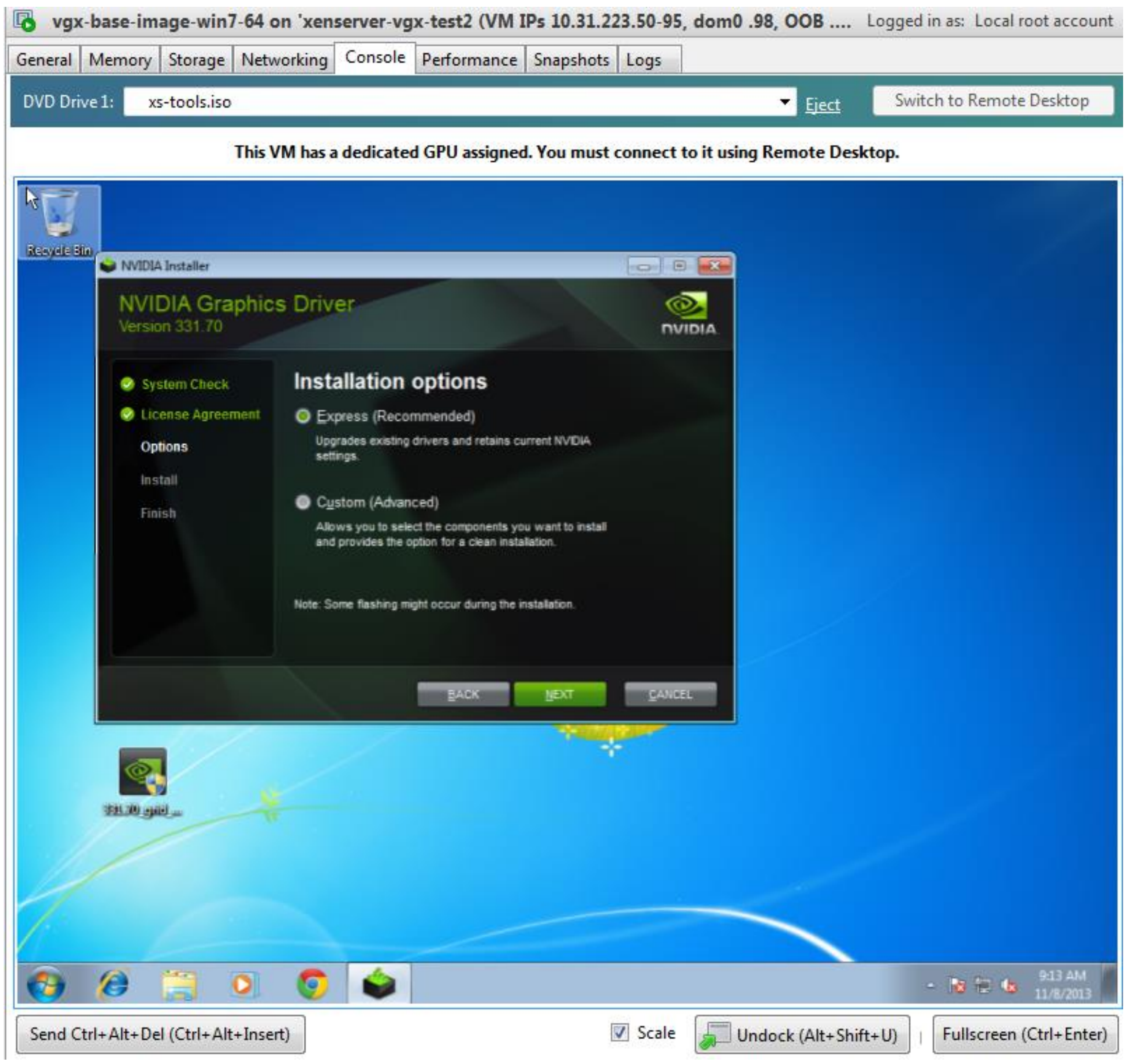

vgx-base-image-win7-64 on 'xenserver-vgx-test2 (VM IPs 10.31.223.50-95, dom0 .98, OOB ....
Logged in as: Local root account
General
Memory
Storage
Networking
Console
Performance
Snapshots
Logs
DVD Drive 1:
xs-tools.iso
Eject
Switch to Remote Desktop
This VM has a dedicated GPU assigned. You must connect to it using Remote Desktop.
Recycle Bin
NVIDIA Installer
NVIDIA Graphics Driver
Version 331.70
System Check
License Agreement
Options
Install
Finish
Installation options
Express (Recommended)
Upgrades existing drivers and retains current NVIDIA settings.
Custom (Advanced)
Allows you to select the components you want to install and provides the option for a clean installation.
Note: Some flashing might occur during the installation.
BACK
NEXT
CANCEL
Send Ctrl+Alt+Del (Ctrl+Alt+Insert)
Scale
Undock (Alt+Shift+U)
Fullscreen (Ctrl+Enter)

Figure 7 NVIDIA driver installation in the guest VM

- Click through the license agreement
- Select Express Installation
- Once driver installation completes, the installer may prompt you to restart the platform. Select Restart Now to reboot the VM, or exit the installer and reboot the VM when ready.

Once the VM restarts, it will boot to a Windows desktop. Verify that the NVIDIA driver is running by right-clicking on the desktop. The NVIDIA Control Panel will be listed in the menu; select it to open the control panel. Selecting "System Information" in the NVIDIA control panel will report the Virtual GPU that the VM is using, its capabilities, and the NVIDIA driver version that is loaded.

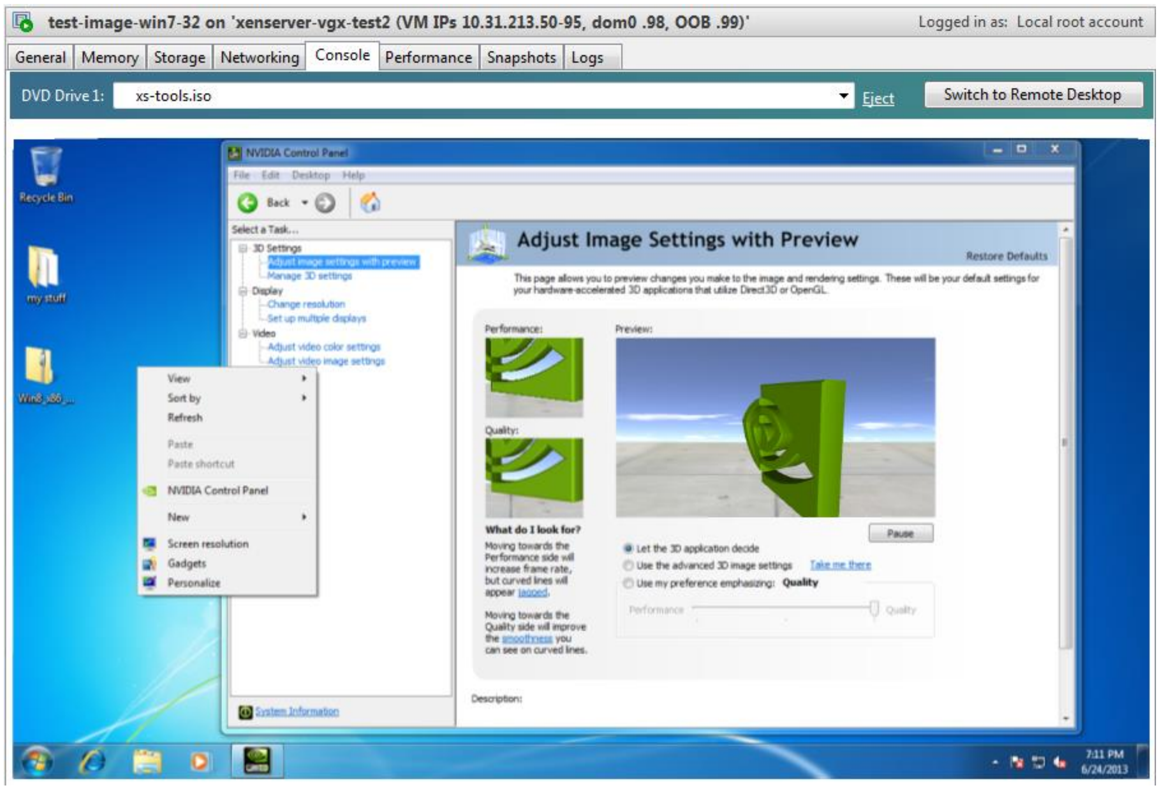


Figure 8 Verifying NVIDIA driver operation using NVIDIA Control Panel

This completes the process of setting up a single VM to use GRID vGPU. The VM is now capable of running the full range of DirectX and OpenGL graphics applications.

2.1.6 Applying a vGPU license

GRID vGPU is a licensed feature on Tesla M6, M60. When booted on these GPUs, a vGPU runs at reduced capability until a license is acquired. See section 2.3 for details on how to configure licensing on Windows.

2.1.7 Removing a VM's vGPU configuration

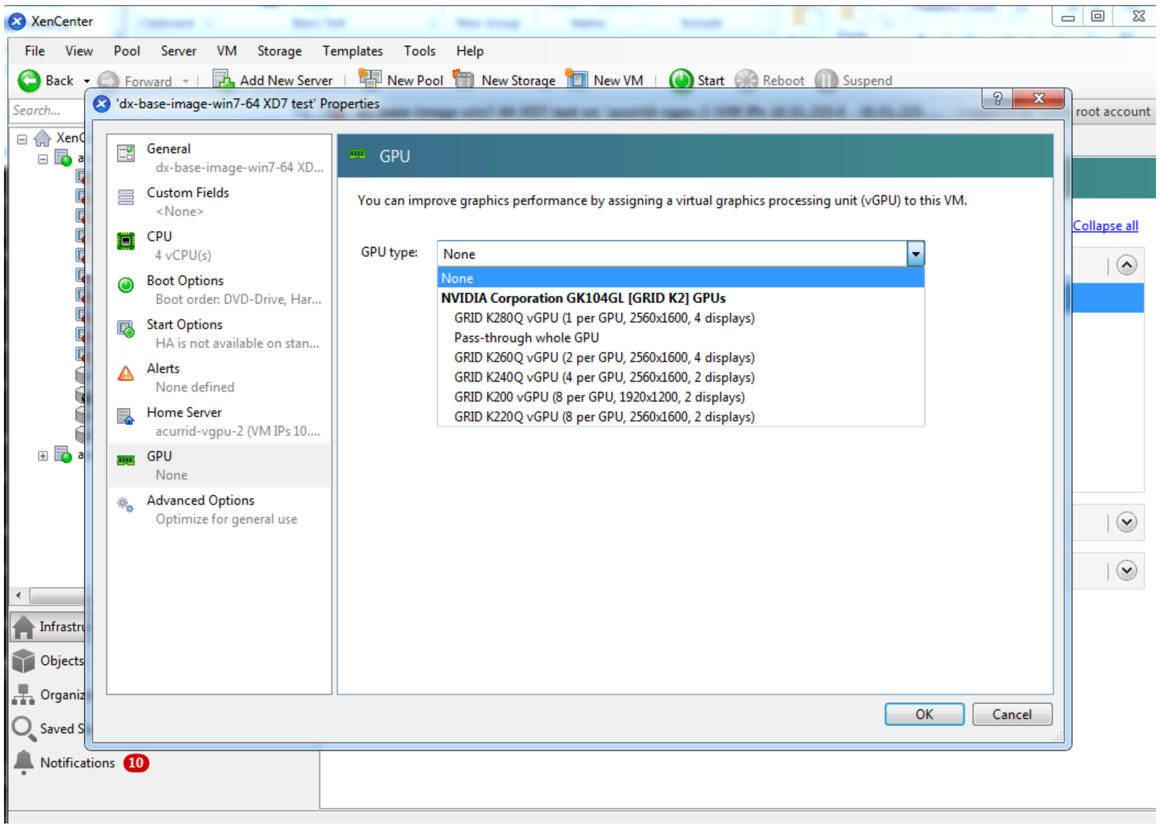
To remove a virtual GPU assignment from a VM, such that it no longer uses a virtual GPU, set the GPU type to 'None' in the VM's GPU Properties, as shown in Figure 9.

Alternatively, use `vgpu-destroy` to delete the virtual GPU object associated with the VM. To discover the vGPU object UUID associated with a given VM, use `vgpu-list`:

```
[root@xenserver ~]# xe vgpu-list vm-uuid=e71afda4-53f4-3a1b-6c92-a364a7f619c2
uuid ( RO)                : c1c7c43d-4c99-af76-5051-119f1c2b4188
          vm-uuid ( RO): e71afda4-53f4-3a1b-6c92-a364a7f619c2
```

```
    gpu-group-uuid ( RO): d53526a9-3656-5c88-890b-5b24144c3d96

[root@xenserver ~]# xe vgpu-destroy uuid=c1c7c43d-4c99-af76-5051-119f1c2b4188
[root@xenserver ~]#
```

Note: the VM must be in the powered-off state in order for its vGPU configuration to be modified or removed.



You can improve graphics performance by assigning a virtual graphics processing unit (vGPU) to this VM.

GPU type: None

None
NVIDIA Corporation GK104GL [GRID K2] GPUs
GRID K280Q vGPU (1 per GPU, 2560x1600, 4 displays)
Pass-through whole GPU
GRID K260Q vGPU (2 per GPU, 2560x1600, 4 displays)
GRID K240Q vGPU (4 per GPU, 2560x1600, 2 displays)
GRID K200 vGPU (8 per GPU, 1920x1200, 2 displays)
GRID K220Q vGPU (8 per GPU, 2560x1600, 2 displays)

Figure 9 Using XenCenter to remove a vGPU configuration from a VM

2.2 VMWARE VSPHERE

2.2.1 Prerequisites

Before proceeding, ensure that you have these prerequisites:

- NVIDIA GRID K1,K2, or Tesla M60, M6 cards.
- A server platform capable of hosting VMware vSphere Hypervisor (ESXi) and the NVIDIA GRID or Tesla cards. Refer to the release notes for a list of recommended servers.
- The NVIDIA GRID vGPU software package for VMware vSphere, consisting of the GRID Virtual GPU Manager for ESXi, and NVIDIA GRID vGPU drivers for Windows, 32- and 64-bit.
- VMware vSphere 2015 or later, obtainable from VMware.
- An installed Windows VM to be enabled with vGPU.

To run VMware Horizon with virtual machines running NVIDIA Virtual GPU, you will also need:

- VMware Horizon 6.1 or later, obtainable from VMware.

Note: Earlier versions of VMware vSphere and Horizon are not supported for use with NVIDIA Virtual GPU.

Review the release notes and known issues for GRID Virtual GPU before proceeding with installation.

2.2.2 Installing VMware vSphere

Install VMware vSphere Hypervisor (ESXi), following VMware's installation instructions. Install VMware vCenter Server, following VMware's installation instructions.

2.2.3 Installing the NVIDIA Virtual GPU Manager for vSphere

The NVIDIA Virtual GPU Manager runs on ESXi host. It is provided as a VIB file, which must be copied to the ESXi host and then installed.

2.2.3.1 Package installation

To install the vGPU Manager VIB you need to access the ESXi host via the ESXi Shell or SSH. Refer to VMware's documentation on how to enable ESXi Shell or SSH for an ESXi host.

Note: Before proceeding with the vGPU Manager installation make sure that all VM's are powered off and the ESXi host is placed in maintenance mode. Refer to VMware's documentation on how to place an ESXi host in maintenance mode.

Use the `esxcli` command to install the vGPU Manager package:

```
[root@esxi:~] esxcli software vib install -v /NVIDIA-vgx-
VMware_ESXi_6.0_Host_Driver_346.42-1OEM.600.0.0.2159203.vib
Installation Result
   Message: Operation finished successfully.
   Reboot Required: false
   VIBs Installed: NVIDIA_bootbank_NVIDIA-vgx-
VMware_ESXi_6.0_Host_Driver_346.42-1OEM.600.0.0.2159203
   VIBs Removed:
   VIBs Skipped:
```

Reboot the ESXi host and remove it from maintenance mode.

Note: GRID Virtual GPU Manager and Guest VM drivers must be matched from the same release. After updating vGPU Manager, guest VMs will boot with vGPU disabled until their guest vGPU driver is updated to match the vGPU Manager version. Consult the release notes for further details.

2.2.3.2 Verifying installation

After the ESXi host has rebooted, verify that the GRID package installed and loaded correctly by checking for the NVIDIA kernel driver in the list of kernel loaded modules.

```
[root@esxi:~] vmkload_mod -l | grep nvidia
nvidia                    5    8420
```

If the `nvidia` driver is not listed in the output, check `dmesg` for any load-time errors reported by the driver.

Verify that the NVIDIA kernel driver can successfully communicate with the GRID physical GPUs in your system by running the `nvidia-smi` command, which should produce a listing of the GPUs in your platform:

```
[root@esxi:~] nvidia-smi
Tue Mar 10 17:56:22 2015
+------------------------------------------------------+
| NVIDIA-SMI 346.42     Driver Version: 346.42         |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GRID K2             On   | 0000:04:00.0     Off |                  Off |
| N/A   27C    P8    27W / 117W |     11MiB /  4095MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  GRID K2             On   | 0000:05:00.0     Off |                  Off |
| N/A   27C    P8    27W / 117W |     10MiB /  4095MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  GRID K2             On   | 0000:08:00.0     Off |                  Off |
| N/A   32C    P8    27W / 117W |     10MiB /  4095MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  GRID K2             On   | 0000:09:00.0     Off |                  Off |
| N/A   32C    P8    27W / 117W |     10MiB /  4095MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   4  GRID K2             On   | 0000:86:00.0     Off |                  Off |
| N/A   24C    P8    27W / 117W |     10MiB /  4095MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+


+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

The `nvidia-smi` command is described in more detail in section 4.1.

If `nvidia-smi` fails to report the expected output for all the NVIDIA GPUs in your system, see Chapter 7 for troubleshooting steps.

2.2.4 Configuring a VM with Virtual GPU

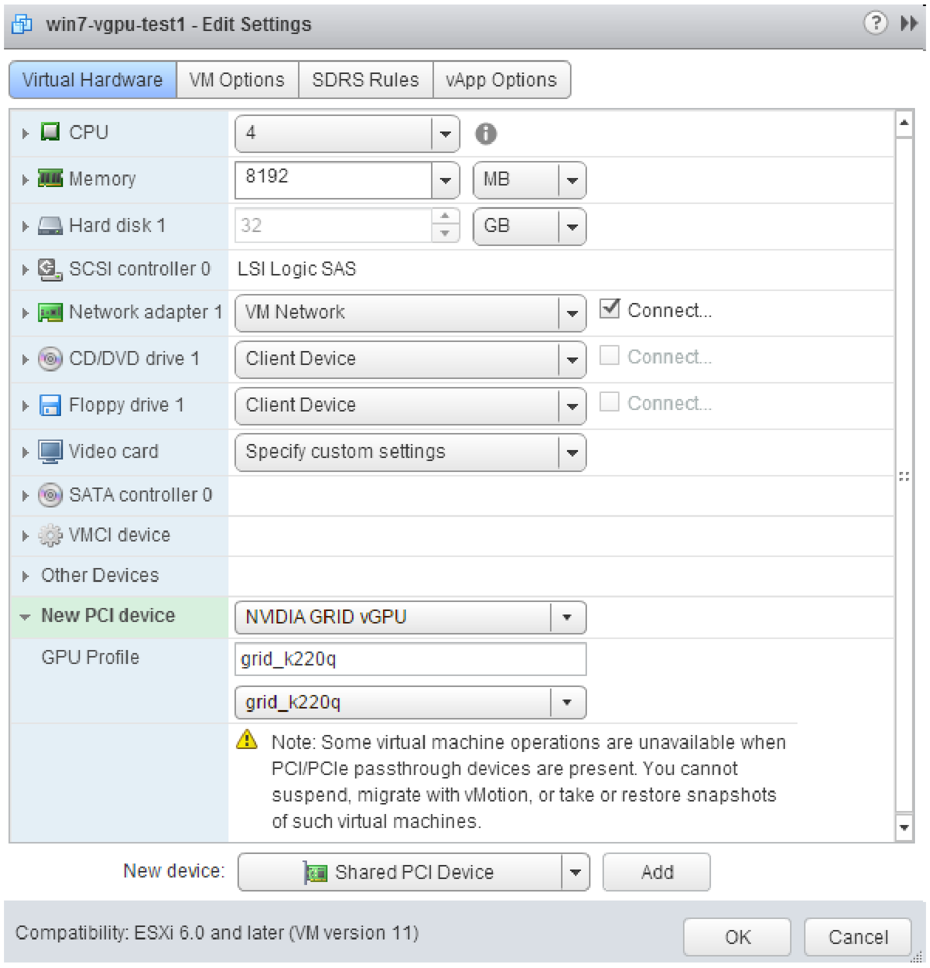
! **Note:** VMware vSphere does not support VM console in vSphere Web Client for VMs configured with vGPU. Make sure that you have installed an alternate means of accessing the VM (such as VMware Horizon or a VNC server) before you configure vGPU.

VM console in vSphere Web Client will become active again once the vGPU parameters are removed from the VM's configuration.

To configure vGPU for a VM:

- Select `Edit Settings` after right-clicking on the VM in the vCenter Web UI
- Select the `Virtual Hardware` tab
- In the `New device` selection, select `Shared PCI Device` and hit `Add`

This should auto-populate `NVIDIA GRID vGPU` in the PCI device field, as shown in Figure 10. In the GPU Profile dropdown menu, select the type of vGPU you wish to configure.



win7-vgpu-test1 - Edit Settings

Virtual Hardware | VM Options | SDRS Rules | vApp Options

CPU	4	
Memory	8192	MB
Hard disk 1	32	GB
SCSI controller 0	LSI Logic SAS	
Network adapter 1	VM Network	✓ Connect...
CD/DVD drive 1	Client Device	Connect...
Floppy drive 1	Client Device	Connect...
Video card	Specify custom settings	
SATA controller 0		
VMCI device		
Other Devices		
New PCI device	NVIDIA GRID vGPU	
GPU Profile	grid_k220q	
	grid_k220q	

Note: Some virtual machine operations are unavailable when PCI/PCIe passthrough devices are present. You cannot suspend, migrate with vMotion, or take or restore snapshots of such virtual machines.

New device: Shared PCI Device — Add

Compatibility: ESXi 6.0 and later (VM version 11)

OK Cancel

Figure 10 VM settings for vGPU

VMs running vGPU should have all their memory reserved:

- Select `Edit virtual machine settings` from vCenter Web UI
- Expand `Memory` section and click `Reserve all guest memory (All locked)`

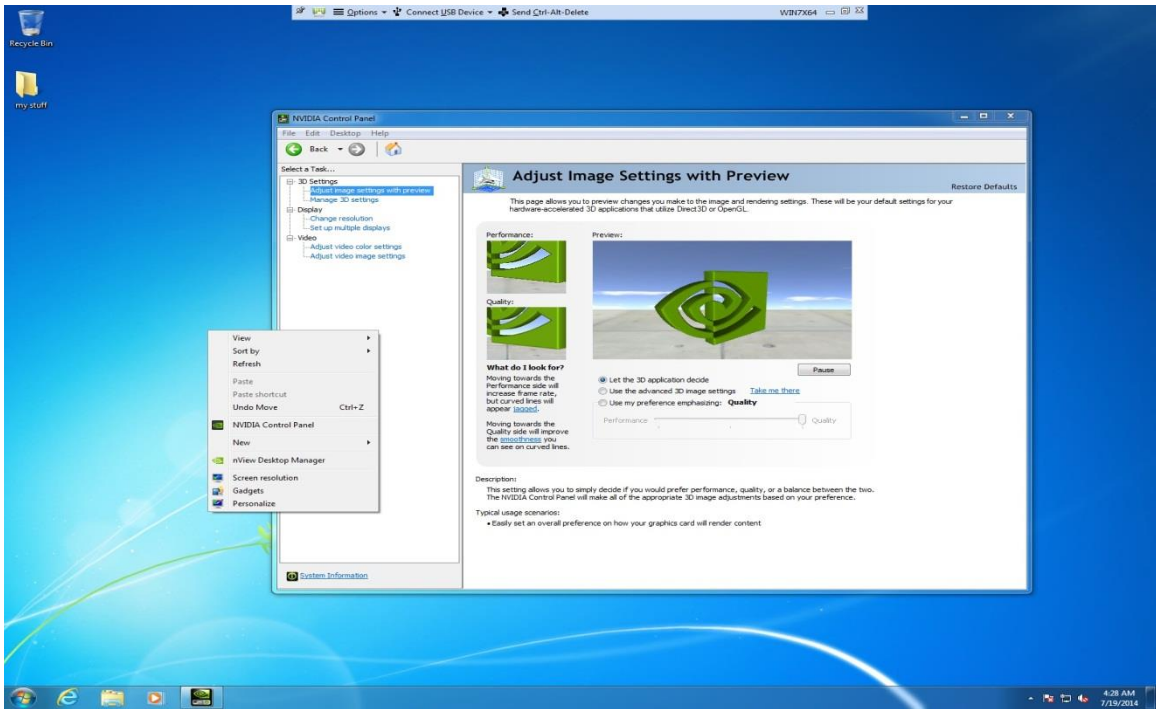
2.2.5 Booting the VM and Installing Drivers

Once you have configured a VM with a vGPU, start the VM. VM console in vSphere Web Client is not supported in this vGPU release; use VMware Horizon or VNC to access the VM's desktop.

The VM should boot to a standard Windows desktop in VGA mode at 800x600 resolution. The Windows screen resolution control panel may be used to increase the resolution to other standard resolutions, but to fully enable vGPU operation, as for a physical NVIDIA GPU, the NVIDIA driver must be installed.

- Copy the 32- or 64-bit NVIDIA Windows driver package to the guest VM and execute it to unpack and run the driver installer.
- Click through the license agreement
- Select `Express Installation`
- Once driver installation completes, the installer may prompt you to restart the platform. Select `Restart Now` to reboot the VM, or exit the installer and reboot the VM when ready.

Once the VM restarts, it will boot to a Windows desktop. Verify that the NVIDIA driver is running by right-clicking on the desktop. The NVIDIA Control Panel will be listed in the menu; select it to open the control panel. Selecting "System Information" in the NVIDIA control panel will report the Virtual GPU that the VM is using, its capabilities, and the NVIDIA driver version that is loaded.



This page allows you to preview changes you make to the image and rendering settings. These will be your default settings for your hardware-accelerated 3D applications that utilize Direct3D or OpenGL.

What do I look for?

Moving towards the Performance side will increase frame rate, but curved lines will appear jagged.

Moving towards the Quality side will improve the smoothness you can see on curved lines.

[✓ Let the 3D application decide]
Use the advanced 3D image settings Take me there
Use my preference emphasizing: Quality
Performance — Quality

Description:
This setting allows you to simply decide if you would prefer performance, quality, or a balance between the two. The NVIDIA Control Panel will make all of the appropriate 3D image adjustments based on your preference.

Typical usage scenarios:
- Easily set an overall preference on how your graphics card will render content

Figure 11 Verifying NVIDIA driver operation using NVIDIA Control Panel

This completes the process of setting up a single VM to use GRID vGPU. The VM is now capable of running the full range of DirectX and OpenGL graphics applications.

2.2.6 Applying a vGPU license

GRID vGPU is a licensed feature on Tesla M6, M60. When booted on these GPUs, a vGPU runs at reduced capability until a license is acquired. See section 2.3 for details on how to configure licensing on Windows.

2.2.7 Removing a VM's vGPU configuration

To remove a vGPU configuration from a VM:

- Select `Edit settings` after right-clicking on the VM in the vCenter Web UI
- Select the `Virtual Hardware` tab
- Mouse over the `PCI Device` entry showing `NVIDIA GRID vGPU` and click on the (X) icon to mark the device for removal.
- Click `OK` to remove the device and update the VM settings.

2.2.8 Modifying GPU assignment for vGPU-Enabled VMs

VMware vSphere Hypervisor (ESXi) by default uses a *breadth-first* allocation scheme for vGPU-enabled VMs; allocating new vGPU-enabled VMs on an available, least loaded physical GPU. This policy generally leads to higher performance because it attempts to minimize sharing of physical GPUs, but in doing so it may artificially limit the total number of vGPUs that can run.

ESXi also provides a *depth-first* allocation scheme for vGPU-enabled VMs. The depth-first allocation policy attempts to maximize the number of vGPUs running on each physical GPU, by placing newly-created vGPUs on the physical GPU that can support the new vGPU and that has the most number of vGPUs already resident. This policy generally leads to higher density of vGPUs, particularly when different types of vGPUs are being run, but may result in lower performance because it attempts to maximize sharing of physical GPUs.

To switch to depth-first allocation scheme add the following parameter to `/etc/vmware/config`:

```
vGPU.consolidation = true
```

2.3 VGPU LICENSING ON WINDOWS

GRID vGPU is a licensed feature on Tesla M6, M60. When booted on these GPUs, a vGPU runs at reduced capability until a license is acquired:

- Screen resolution is limited to no higher than 1280x1024.
- Frame rate is capped at 3 frames per second.
- GPU resource allocations are limited, which will prevent some applications from running correctly.

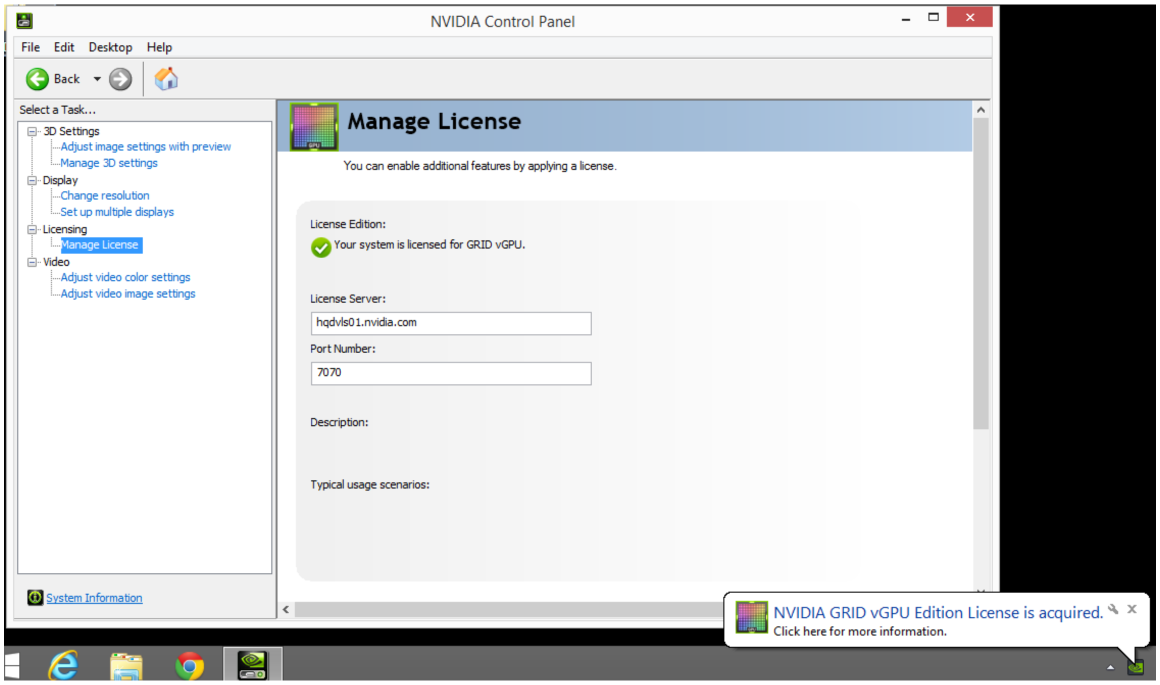
These restrictions are removed once a license is acquired.

Full information on configuring and using GRID licensed features, including vGPU, is give in the **GRID Licensing User Guide**. Basic configuration information is given here:

To configure licensing, open NVIDIA Control Panel and select **Manage License** task in the **Licensing** section of the navigation pane. Enter the address of your local GRID License Server in the **License Server** field. The address can be a fully-qualified domain name such as `gridlicense.mycompany.com`, or an IP address such as `10.31.20.45`.

The **Port Number** field can be left unset and will default to `7070`, which is the default port number used by NVIDIA GRID License Server.

Select **Apply** to assign the settings. The system will request the appropriate license for the current vGPU from the configured license server and, if successful, vGPU's full capabilities are enabled (see Figure 12). If the system fails to obtain a license, refer to the **GRID Licensing User Guide** for guidance on troubleshooting.



Manage License

You can enable additional features by applying a license.

License Edition:

Your system is licensed for GRID vGPU.

License Server:

hqdvls01.nvidia.com

Port Number:

7070

Description:

Typical usage scenarios:

NVIDIA GRID vGPU Edition License is acquired.

Click here for more information.

Figure 12 Configuring vGPU licensing on Windows

Chapter 3. USING VGPU ON LINUX

Tesla M6/M60 GPUs support vGPU on Linux VMs. 64-bit Linux guest VMs are supported on the following virtual GPU types:

Tesla M60	M60-8Q
	M60-4Q
	M60-2Q
	M60-1Q
	M60-0Q

Tesla M6	M6-8Q
	M6-4Q
	M6-2Q
	M6-1Q
	M6-0Q

Table 4 Virtual GPUs supporting Linux

Note: Refer to the driver release notes for further information on supported hypervisor and Linux VM configurations.

3.1 INSTALLING VGPU DRIVERS

After creating and booting a Linux VM on the hypervisor, the steps to install NVIDIA Linux vGPU drivers are largely the same as those for installing NVIDIA GPU drivers on a VM running passthrough GPU, or on baremetal Linux.

3.1.1 Prerequisites

Installation of the NVIDIA Linux driver requires:

- Compiler toolchain
- Kernel headers

3.1.2 Running the driver installer

- Copy the NVIDIA GRID Linux driver package, for example, `NVIDIA-Linux_x86_64-352.47-grid.run`, to the Linux VM.
- Before the driver installer can be run, you must exit the X server and terminate all OpenGL applications. On Red Hat Enterprise Linux / CentOS systems, this can typically be achieved by transitioning to runlevel 3:

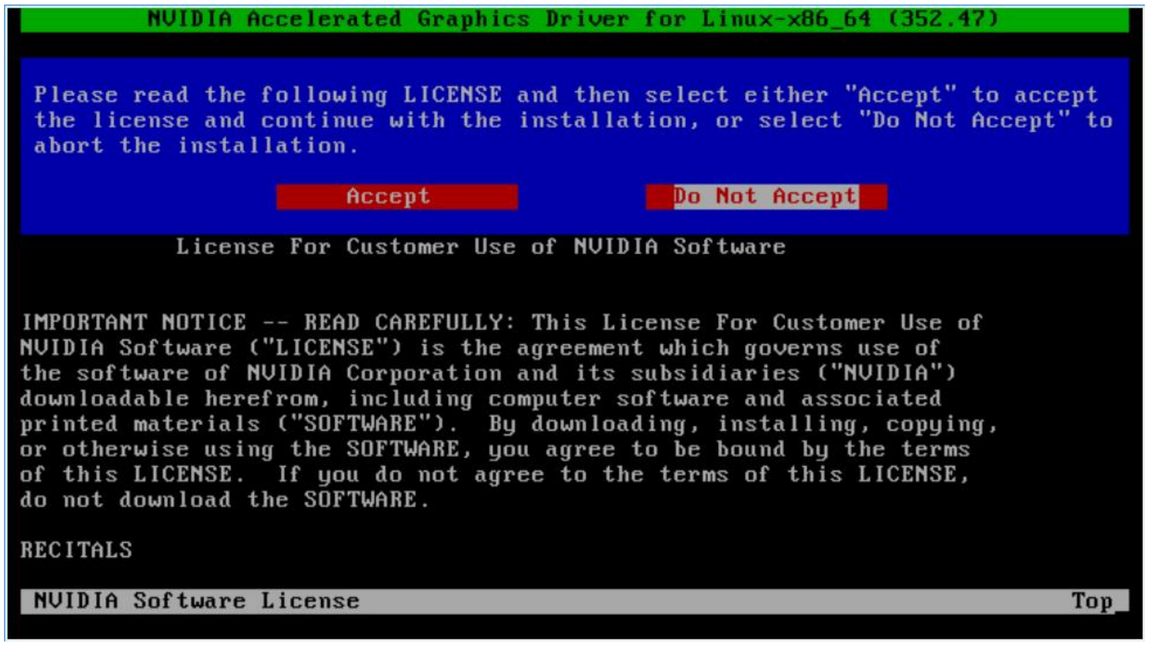
```
[nvidia@localhost ~]$ sudo init 3
```

  On Ubuntu platforms, use `CTRL-ALT-F1` to switch to a console login prompt. Log in and shut down the display manager:

```
[nvidia@localhost ~]$ sudo service lightdm stop
```

- From a console shell, run the driver installer as the root user. The installer should launch and display the driver license agreement (Figure 13):

```
sudo sh ./ NVIDIA-Linux_x86_64-352.47-grid.run
```



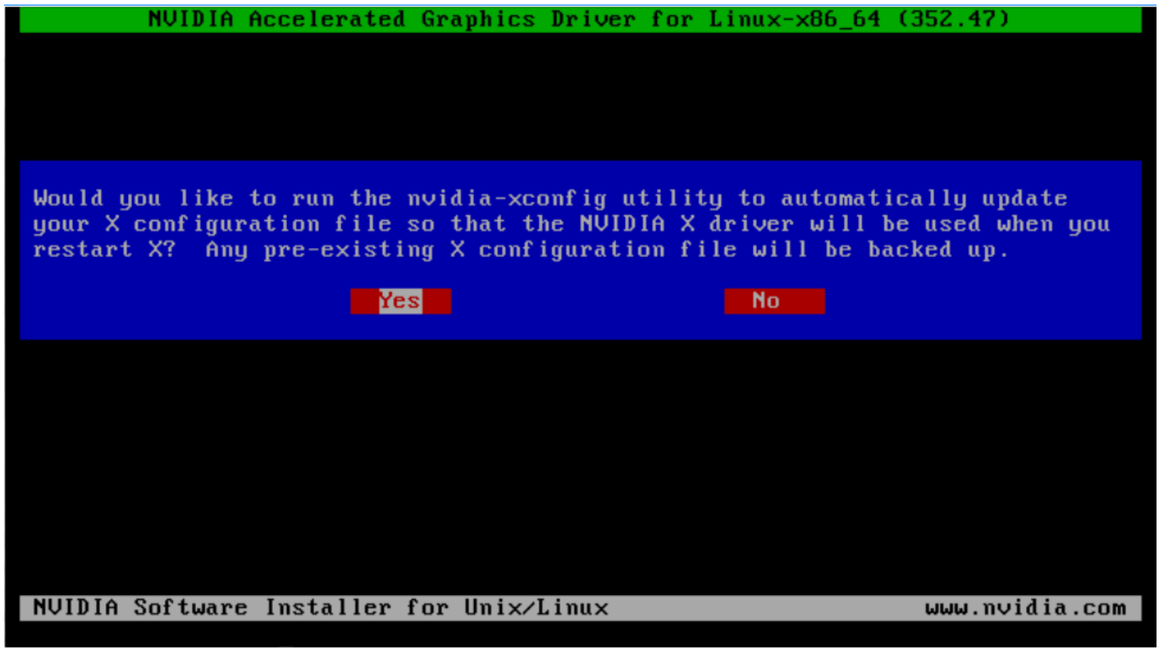
```
NVIDIA Accelerated Graphics Driver for Linux-x86_64 (352.47)

Please read the following LICENSE and then select either "Accept" to accept
the license and continue with the installation, or select "Do Not Accept" to
abort the installation.

                Accept                    Do Not Accept

          License For Customer Use of NVIDIA Software


IMPORTANT NOTICE -- READ CAREFULLY: This License For Customer Use of
NVIDIA Software ("LICENSE") is the agreement which governs use of
the software of NVIDIA Corporation and its subsidiaries ("NVIDIA")
downloadable herefrom, including computer software and associated
printed materials ("SOFTWARE").  By downloading, installing, copying,
or otherwise using the SOFTWARE, you agree to be bound by the terms
of this LICENSE.  If you do not agree to the terms of this LICENSE,
do not download the SOFTWARE.

RECITALS

NVIDIA Software License                                              Top
```

Figure 13 NVIDIA Linux driver installer

- Accept the license agreement to continue with the driver installation. In some instances the installer may fail to detect the installed kernel headers / sources, in which case the installer should be re-run with the kernel source path specified using the `--kernel-source-path` option:

```
sudo sh ./ NVIDIA-Linux_x86_64-352.47-grid.run \
     -kernel-source-path=/usr/src/kernels/3.10.0-229.11.1.el7.x86_64
```

- When prompted, accept the option to update the X configuration file (xorg.conf) settings (Figure 14):

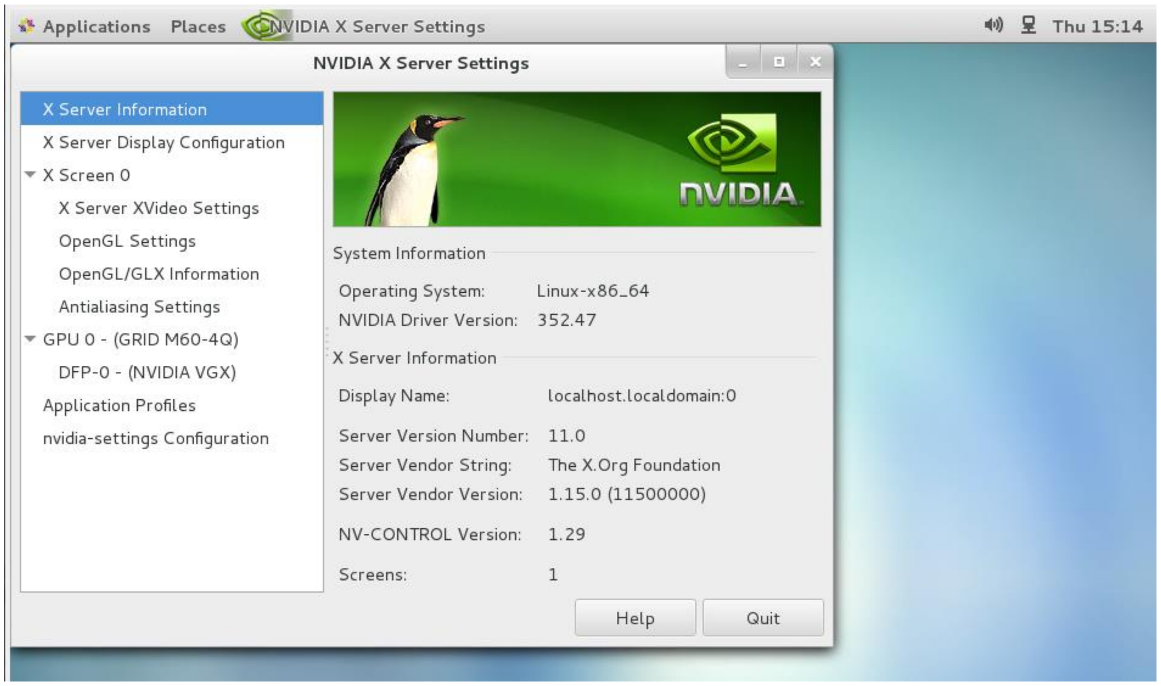
- 

```
NVIDIA Accelerated Graphics Driver for Linux-x86_64 (352.47)

Would you like to run the nvidia-xconfig utility to automatically update
your X configuration file so that the NVIDIA X driver will be used when you
restart X?  Any pre-existing X configuration file will be backed up.

NVIDIA Software Installer for Unix/Linux                    www.nvidia.com
```

Figure 14 Update xorg.conf settings

- Once installation has completed, select OK to exit the installer.
- Reboot the system, log in, and run nvidia-settings to verify that the NVIDIA is operational with vGPU (Figure 15):

```
[nvidia@localhost ~]$ nvidia-settings
```



System Information

Operating System:	Linux-x86_64
NVIDIA Driver Version:	352.47

X Server Information

Display Name:	localhost.localdomain:0
Server Version Number:	11.0
Server Vendor String:	The X.Org Foundation
Server Vendor Version:	1.15.0 (11500000)
NV-CONTROL Version:	1.29
Screens:	1

Figure 15 Verifying operation with nvidia-settings

3.2 VGPU LICENSING ON LINUX

GRID vGPU is a licensed feature on Tesla M6, M60. When booted on these GPUs, a vGPU runs at reduced capability until a license is acquired:

- Screen resolution is limited to no higher than 1280x1024.
- Frame rate is capped at 3 frames per second.
- GPU resource allocations are limited, which will prevent some applications from running correctly.

These restrictions are removed once a license is acquired.

Full information on configuring and using GRID licensed features, including vGPU, is give in the **GRID Licensing User Guide**. Basic configuration information is given here:

To license GRID vGPU, edit `/etc/nvidia/gridd.conf`:

```
[nvidia@localhost ~]$ sudo vi /etc/nvidia/gridd.conf
```

Set `ServerURL` to the address and port number of your local NVIDIA GRID License Server. The address can be a fully-qualified domain name such as `gridlicense.mycompany.com`, or an IP address such as `10.31.20.45`. The port number is appended to the address with a colon, for example `:7070`.

Set `FeatureType` to 1, to license vGPU:

```
# /etc/nvidia/gridd.conf - Configuration file for NVIDIA Grid Daemon

# Description: Set License Server URL
# Data type: string
# Format:  "<address>:<port>"
ServerUrl=hqdvls01.nvidia.com:7070

# Description: Set Feature to be enabled
# Data type: integer
# Possible values:
#    1 => for GRID vGPU
#    2 => for GRID Virtual Workstation
FeatureType=1

# Description: Parameter to enable or disable Grid Licensing tab in nvidia-
settings
# Data type: boolean
# Possible values: TRUE or FALSE, default is TRUE
#EnableUI=TRUE
```

Figure 16 Sample gridd.conf for GRID vGPU

Restart the `nvidia-gridd` service:

```
[nvidia@localhost ~]$  sudo service nvidia-gridd restart
```

The service should automatically obtain a license. This can be confirmed with log messages written to `/var/log/messages`, and the vGPU within the VM should now exhibit full framerate, resolution, and display output capabilities:

```
[nvidia@localhost ~]$ sudo grep gridd /var/log/messages
…
Sep 13 15:40:06 localhost nvidia-gridd: Started (10430)
Sep 13 15:40:24 localhost nvidia-gridd: License acquired successfully.
```

Once configured in `gridd.conf`, licensing settings persist across reboots and need only be modified if the license server address changes, or the VM is switched to running GPU passthrough.

If the system fails to obtain a license, refer to the **GRID Licensing User Guide** for guidance on troubleshooting

Chapter 4. PERFORMANCE MONITORING

Physical GPU performance monitoring can be done using the `nvidia-smi` command line utility and, on Citrix XenServer platforms, using Citrix XenCenter.

Note: It is not currently possible to monitor the performance of GPUs being used for GPU passthrough.

4.1 USING NVIDIA-SMI

NVIDIA System Management Interface, `nvidia-smi`, is a command line tool that reports management information for NVIDIA physical GPUs present in the system. `nvidia-smi` is run from the XenServer dom0 or ESXi host shells and, when invoked without additional arguments, it provides a summary of all GPUs in the system, along with PCI bus IDs, power state, temperature, current memory usage, and so on.

Note: `nvidia-smi` will not list any GPU currently allocated for GPU passthrough.

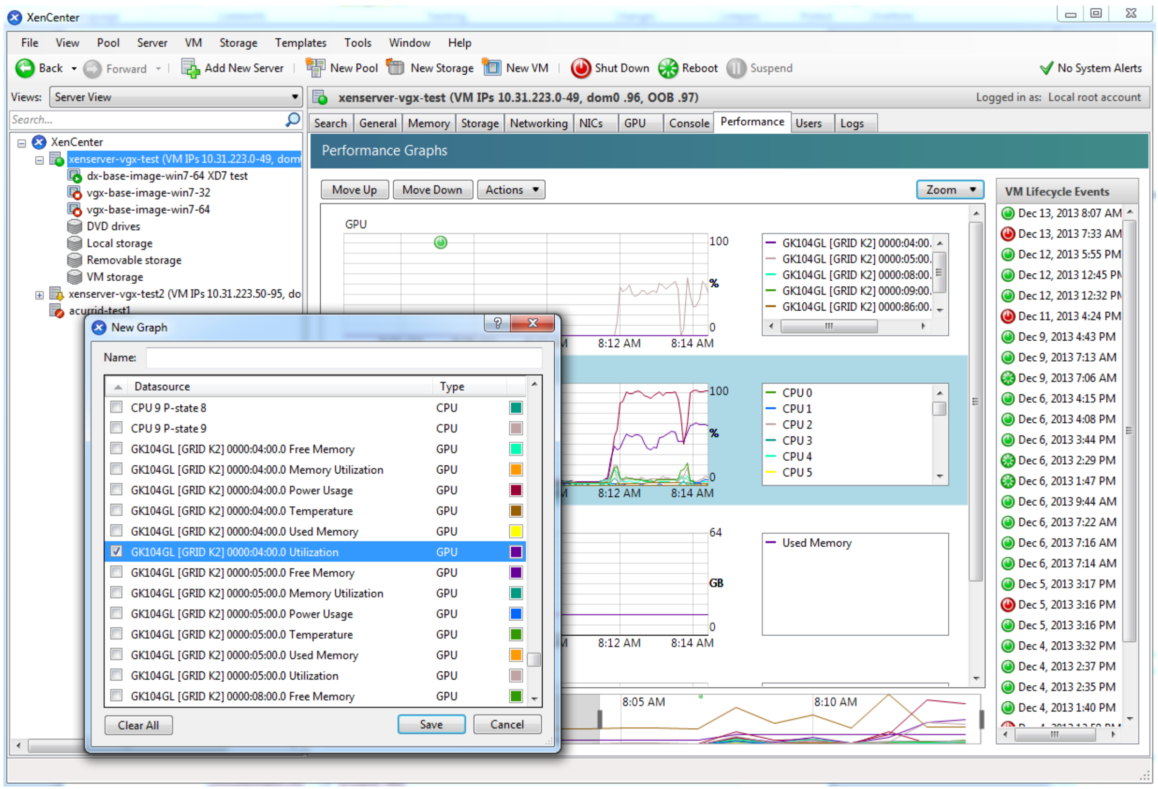
In this release of GRID vGPU, `nvidia-smi` provides basic reporting of vGPU instances running on physical GPUs; each vGPU instance is reported in the “Compute processes” section, together with its physical GPU index and the amount of framebuffer memory assigned to it. In the example that follows, five vGPUs are running; one on physical GPU 0, and four on physical GPU 1:

```
[root@xenserver ~]# nvidia-smi
Mon Nov 10 18:46:50 2014
+------------------------------------------------------+
| NVIDIA-SMI 340.57     Driver Version: 340.57         |
|-------------------------------+----------------------+----------------------+
| GPU  Name                     | Bus-Id        Disp.  | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap| Memory-Usage         | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GRID K1                  | 0000:04:00.0     Off |                  N/A |
| N/A   27C    P8     8W /  31W |   7%  270MB / 4095MB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  GRID K1                  | 0000:05:00.0     Off |                  N/A |
| N/A   26C    P8     8W /  31W |  26% 1048MB / 4095MB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  GRID K1                  | 0000:06:00.0     Off |                  N/A |
| N/A   22C    P0    13W /  31W |   0%     9MB / 4095MB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  GRID K1                  | 0000:07:00.0     Off |                  N/A |
| N/A   25C    P0    13W /  31W |   0%     9MB / 4095MB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   4  GRID K1                  | 0000:86:00.0     Off |                  N/A |
| N/A   27C    P0    14W /  31W |   0%     9MB / 4095MB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   5  GRID K1                  | 0000:87:00.0     Off |                  N/A |
| N/A   27C    P0    13W /  31W |   0%     9MB / 4095MB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   6  GRID K1                  | 0000:88:00.0     Off |                  N/A |
| N/A   29C    P0    13W /  31W |   0%     9MB / 4095MB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   7  GRID K1                  | 0000:89:00.0     Off |                  N/A |
| N/A   28C    P0    12W /  31W |   0%     9MB / 4095MB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Compute processes:                                               GPU Memory |
|  GPU       PID  Process name                                     Usage      |
|=============================================================================|
|    0     10300  /usr/lib/xen/bin/vgpu                                256MB   |
|    1     10350  /usr/lib/xen/bin/vgpu                                256MB   |
|    1     10321  /usr/lib/xen/bin/vgpu                                256MB   |
|    1     11512  /usr/lib/xen/bin/vgpu                                256MB   |
|    1     10210  /usr/lib/xen/bin/vgpu                                256MB   |
+-----------------------------------------------------------------------------+
[root@xenserver ~]#
```

For a list of commands supported by `nvidia-smi`, run `nvidia-smi -h`. Note that not all commands apply to GRID supported GPUs.

4.2 USING CITRIX XENCENTER

To monitor GPU performance in XenCenter, click on a server's Performance Tab, then right-click on the graph window, select Actions, then select New Graph. Provide a name for the graph, and in the list of available counter resources, select one or more GPU counters. Counters are listed for each physical GPU not currently being used for GPU passthrough.


XenCenter
File View Pool Server VM Storage Templates Tools Window Help
Back Forward Add New Server New Pool New Storage New VM Shut Down Reboot Suspend No System Alerts
Views: Server View
xenserver-vgx-test (VM IPs 10.31.223.0-49, dom0 .96, OOB .97)
Logged in as: Local root account
Search General Memory Storage Networking NICs GPU Console Performance Users Logs
Performance Graphs
Move Up Move Down Actions Zoom
VM Lifecycle Events
GPU
CPU 0
CPU 1
CPU 2
CPU 3
CPU 4
CPU 5
Used Memory
New Graph
Name:
Clear All Save Cancel

Datasource	Type	
☐ CPU 9 P-state 8	CPU	🟩
☐ CPU 9 P-state 9	CPU	🟫
☐ GK104GL [GRID K2] 0000:04:00.0 Free Memory	GPU	🟦
☐ GK104GL [GRID K2] 0000:04:00.0 Memory Utilization	GPU	🟧
☐ GK104GL [GRID K2] 0000:04:00.0 Power Usage	GPU	🟥
☐ GK104GL [GRID K2] 0000:04:00.0 Temperature	GPU	🟫
☐ GK104GL [GRID K2] 0000:04:00.0 Used Memory	GPU	🟨
☑ GK104GL [GRID K2] 0000:04:00.0 Utilization	GPU	🟪
☐ GK104GL [GRID K2] 0000:05:00.0 Free Memory	GPU	🟪
☐ GK104GL [GRID K2] 0000:05:00.0 Memory Utilization	GPU	🟩
☐ GK104GL [GRID K2] 0000:05:00.0 Power Usage	GPU	🟦
☐ GK104GL [GRID K2] 0000:05:00.0 Temperature	GPU	🟩
☐ GK104GL [GRID K2] 0000:05:00.0 Used Memory	GPU	🟧
☐ GK104GL [GRID K2] 0000:05:00.0 Utilization	GPU	🟫
☐ GK104GL [GRID K2] 0000:08:00.0 Free Memory	GPU	🟩

Figure 17 Using XenCenter to monitor GPU performance

Chapter 5. XENSERVER VGPU MANAGEMENT

This chapter describes Citrix XenServer advanced vGPU management techniques using XenCenter and `xe` command line operations.

5.1 MANAGEMENT OBJECTS FOR GPUS

XenServer uses four underlying management objects for GPUs: physical GPUs, GPU groups, vGPU types, and vGPUs. These objects are used directly when managing vGPU via `xe`, and indirectly when managing vGPU via XenCenter.

5.1.1 pgpu

A `pgpu` object represents a physical GPU, such as one of the multiple GPUs present on a GRID K1 or K2 card. XenServer automatically creates `pgpu` objects at startup to represent each physical GPU present on the platform.

To list the physical GPU objects present on a platform, use `xe pgpu-list`. For example, this platform contains a single GRID K2 card with two physical GPUs:

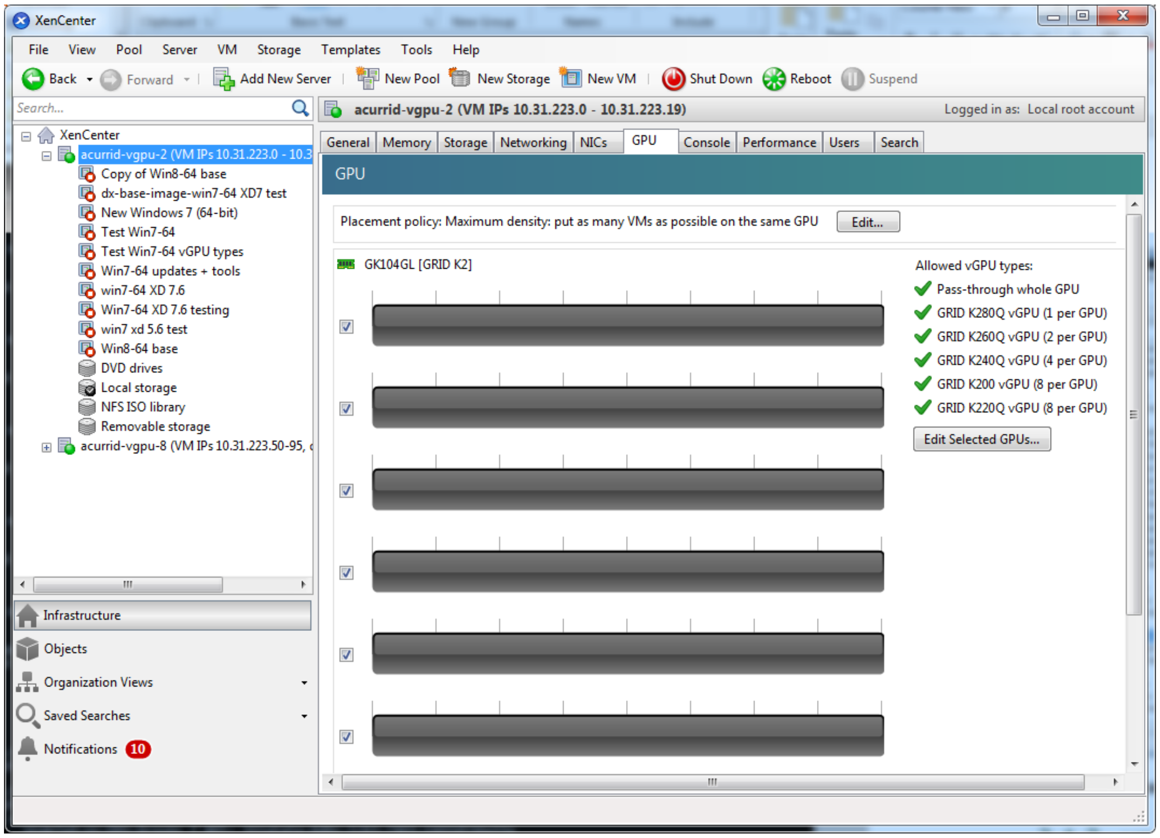
```
[root@xenserver ~]# xe pgpu-list
uuid ( RO)                : 7c1e3cff-1429-0544-df3d-bf8a086fb70a
       vendor-name ( RO): NVIDIA Corporation
       device-name ( RO): GK104GL [GRID K2]
    gpu-group-uuid ( RW): be825ba2-01d7-8d51-9780-f82cfaa64924

uuid ( RO)                : d07fa627-7dc9-f625-92be-ce5d2655e830
       vendor-name ( RO): NVIDIA Corporation
       device-name ( RO): GK104GL [GRID K2]
    gpu-group-uuid ( RW): be825ba2-01d7-8d51-9780-f82cfaa64924
```

```
[root@xenserver ~]#
```

To see detailed information about a `pgpu`, use `xe pgpu-param-list`:

```
[root@xenserver ~]# xe pgpu-param-list uuid=d07fa627-7dc9-f625-92be-
ce5d2655e830
uuid ( RO)                      : d07fa627-7dc9-f625-92be-ce5d2655e830
             vendor-name ( RO): NVIDIA Corporation
             device-name ( RO): GK104GL [GRID K2]
          gpu-group-uuid ( RW): 315a1e1e-6d0c-1cb3-7903-1602d236a33a
    gpu-group-name-label ( RO): Group of NVIDIA Corporation GK104GL [GRID K2]
GPUs
               host-uuid ( RO): 2305cc32-c2d8-4fbd-b1aa-d0b0121ba454
         host-name-label ( RO): acurrid-vgpu-2 (VM IPs 10.31.223.0 -
10.31.223.19)
                  pci-id ( RO): 0000:0a:00.0
            dependencies (SRO):
            other-config (MRW):
    supported-VGPU-types ( RO): c18ab767-ba72-b286-9350-d8010bab4f30; 7cd190db-
e4fe-e824-cf4a-ff1604882323; 24a7baa3-a70a-8c7b-ee7d-f437e0857eca; bfcfb8cd-
c01b-2691-272c-8e908937922d; 0d581f02-c601-a9b1-f440-f852f31f583e; 2c210411-
7de3-37f5-c58c-9635b40d50f6
      enabled-VGPU-types (SRW): c18ab767-ba72-b286-9350-d8010bab4f30; 7cd190db-
e4fe-e824-cf4a-ff1604882323; 24a7baa3-a70a-8c7b-ee7d-f437e0857eca; bfcfb8cd-
c01b-2691-272c-8e908937922d; 0d581f02-c601-a9b1-f440-f852f31f583e; 2c210411-
7de3-37f5-c58c-9635b40d50f6
          resident-VGPUs ( RO):
[root@xenserver ~]#
```

To view physical GPUs in XenCenter, click on the server's GPU tab:


XenCenter
File View Pool Server VM Storage Templates Tools Help
Back Forward Add New Server New Pool New Storage New VM Shut Down Reboot Suspend
Search...
XenCenter
acurrid-vgpu-2 (VM IPs 10.31.223.0 - 10.3
Copy of Win8-64 base
dx-base-image-win7-64 XD7 test
New Windows 7 (64-bit)
Test Win7-64
Test Win7-64 vGPU types
Win7-64 updates + tools
win7-64 XD 7.6
Win7-64 XD 7.6 testing
win7 xd 5.6 test
Win8-64 base
DVD drives
Local storage
NFS ISO library
Removable storage
acurrid-vgpu-8 (VM IPs 10.31.223.50-95,
Infrastructure
Objects
Organization Views
Saved Searches
Notifications 10
acurrid-vgpu-2 (VM IPs 10.31.223.0 - 10.31.223.19)
Logged in as: Local root account
General Memory Storage Networking NICs GPU Console Performance Users Search
GPU
Edit...
GK104GL [GRID K2]
Edit Selected GPUs...

Placement policy: Maximum density: put as many VMs as possible on the same GPU

Allowed vGPU types:
- ✓ Pass-through whole GPU
- ✓ GRID K280Q vGPU (1 per GPU)
- ✓ GRID K260Q vGPU (2 per GPU)
- ✓ GRID K240Q vGPU (4 per GPU)
- ✓ GRID K200 vGPU (8 per GPU)
- ✓ GRID K220Q vGPU (8 per GPU)

Figure 18 Physical GPU display in XenCenter

5.1.2 vgpu-type

A `vgpu-type` represents a type of virtual GPU, such as GRID K100, K140Q, K200, etc. An additional, *passthrough* vGPU type is defined to represent a physical GPU that is directly assignable to a single guest VM.

XenServer automatically creates `vgpu-type` objects at startup to represent each virtual type supported by the physical GPUs present on the platform.

To list the `vgpu-type` objects present on a platform, use `xe vgpu-type-list`. For example, this platform contains multiple GRID K2 cards, therefore the vGPU types reported are solely those supported by GRID K2:

```
[root@xenserver ~]# xe vgpu-type-list
uuid ( RO)                : 7cd190db-e4fe-e824-cf4a-ff1604882323
       vendor-name ( RO): NVIDIA Corporation
        model-name ( RO): GRID K240Q
         max-heads ( RO): 2
    max-resolution ( RO): 2560x1600


uuid ( RO)                : 2c210411-7de3-37f5-c58c-9635b40d50f6
       vendor-name ( RO): NVIDIA Corporation
        model-name ( RO): GRID K220Q
         max-heads ( RO): 2
    max-resolution ( RO): 2560x1600


uuid ( RO)                : 24a7baa3-a70a-8c7b-ee7d-f437e0857eca
       vendor-name ( RO): NVIDIA Corporation
        model-name ( RO): GRID K260Q
         max-heads ( RO): 4
    max-resolution ( RO): 2560x1600


uuid ( RO)                : 0d581f02-c601-a9b1-f440-f852f31f583e
       vendor-name ( RO): NVIDIA Corporation
        model-name ( RO): GRID K200
         max-heads ( RO): 2
    max-resolution ( RO): 1920x1200


uuid ( RO)                : c18ab767-ba72-b286-9350-d8010bab4f30
       vendor-name ( RO):
        model-name ( RO): passthrough
         max-heads ( RO): 0
    max-resolution ( RO): 0x0


uuid ( RO)                : bfcfb8cd-c01b-2691-272c-8e908937922d
       vendor-name ( RO): NVIDIA Corporation
        model-name ( RO): GRID K280Q
         max-heads ( RO): 4
    max-resolution ( RO): 2560x1600

[root@xenserver ~]#
```

To see detailed information about a `vgpu-type`, use `xe vgpu-type-param-list`:

```
[root@xenserver ~]# xe vgpu-type-param-list uuid=7cd190db-e4fe-e824-cf4a-
ff1604882323
uuid ( RO)                         : 7cd190db-e4fe-e824-cf4a-ff1604882323
                vendor-name ( RO): NVIDIA Corporation
                 model-name ( RO): GRID K240Q
           framebuffer-size ( RO): 939524096
                  max-heads ( RO): 2
             max-resolution ( RO): 2560x1600
         supported-on-PGPUs ( RO): d72b9b0d-ae86-a1fa-4432-a46bcef4a4ab;
f17f00fc-dff2-ecb0-5bdb-8f050da2fd8b; 13cfa311-93fe-79e5-f94f-1e8c38a88486;
a9474d47-ddba-ab3a-8f44-58163ffa49f8; 8d147385-40a5-7305-95ea-de92ed4bcfc8;
d3984345-f8e1-c5fe-c5fc-78d2225f0382; 50493ce6-f3b1-1bd9-c012-2457472f2a92;
4778208a-97a9-cbf0-cedf-a20cd28f91f3
           enabled-on-PGPUs ( RO): d72b9b0d-ae86-a1fa-4432-a46bcef4a4ab;
f17f00fc-dff2-ecb0-5bdb-8f050da2fd8b; 13cfa311-93fe-79e5-f94f-1e8c38a88486;
a9474d47-ddba-ab3a-8f44-58163ffa49f8; 8d147385-40a5-7305-95ea-de92ed4bcfc8;
d3984345-f8e1-c5fe-c5fc-78d2225f0382; 50493ce6-f3b1-1bd9-c012-2457472f2a92;
4778208a-97a9-cbf0-cedf-a20cd28f91f3
    supported-on-GPU-groups ( RO): 315a1e1e-6d0c-1cb3-7903-1602d236a33a
      enabled-on-GPU-groups ( RO): 315a1e1e-6d0c-1cb3-7903-1602d236a33a
                 VGPU-uuids ( RO): b6242c9c-87ad-92e9-5a24-a6bd1a3d8950
[root@xenserver ~]#
```

5.1.3 gpu-group

A `gpu-group` is a collection of physical GPUs, all of the same type. XenServer automatically creates `gpu-group` objects at startup to represent the distinct types of physical GPU present on the platform.

To list the `gpu-group` objects present on a platform, use `xe gpu-group-list`. For example, a system with a single GRID K2 card contains a single GPU group of type GRID K2:

```
[root@xenserver ~]# xe gpu-group-list
uuid ( RO)                : be825ba2-01d7-8d51-9780-f82cfaa64924
          name-label ( RW): Group of NVIDIA Corporation GK104GL [GRID K2] GPUs
    name-description ( RW):

[root@xenserver ~]#
```

To see detailed information about a `gpu-group`, use `xe gpu-group-param-list`:

```
[root@xenserver ~]# xe gpu-group-param-list uuid=be825ba2-01d7-8d51-9780-
f82cfaa64924
uuid ( RO)                    : be825ba2-01d7-8d51-9780-f82cfaa64924
              name-label ( RW): Group of NVIDIA Corporation GK104GL [GRID K2]
GPUs
        name-description ( RW):
              VGPU-uuids (SRO): 6493ff45-d895-764c-58d8-96f1bc0307aa; 8481cb68-
66e5-25e6-a0c0-bd691df682b3; b73cbd30-096f-8a9a-523e-a800062f4ca7
              PGPU-uuids (SRO): a4a4df34-4e5f-de9f-82d6-2134d9e339dc; 84c76e93-
555c-5ffa-e9a9-0d6fcb9ff48d; d07fa627-7dc9-f625-92be-ce5d2655e830; 7c1e3cff-
1429-0544-df3d-bf8a086fb70a
            other-config (MRW):
      enabled-VGPU-types ( RO): d1fb00dd-02e6-e7df-ccd5-1944965ece55; 1a312df9-
5397-bd44-c447-c6da804d2fe7; fa50b0f0-9705-6c59-689e-ea62a3d35237; 3f318889-
7508-c9fd-7134-003d4d05ae56
    supported-VGPU-types ( RO): d1fb00dd-02e6-e7df-ccd5-1944965ece55; 1a312df9-
5397-bd44-c447-c6da804d2fe7; fa50b0f0-9705-6c59-689e-ea62a3d35237; 3f318889-
7508-c9fd-7134-003d4d05ae56
    allocation-algorithm ( RW): depth-first

[root@xenserver ~]
```

5.1.4 vgpu

A `vgpu` object represents a virtual GPU. Unlike the other GPU management objects, vGPUs are not created automatically by XenServer; they are created when a VM is configured via XenCenter or via `xe` to use a vGPU, or by cloning a VM that is configured to use vGPU (see section 5.4).

5.2 CREATING A VGPU USING XE

Use `xe vgpu-create` to create a `vgpu` object, specifying the type of vGPU required, the GPU group it will be allocated from, and the VM it is associated with:

```
[root@xenserver ~]# xe vgpu-create vm-uuid=e71afda4-53f4-3a1b-6c92-a364a7f619c2
gpu-group-uuid=be825ba2-01d7-8d51-9780-f82cfaa64924 vgpu-type-uuid=3f318889-
7508-c9fd-7134-003d4d05ae56
b73cbd30-096f-8a9a-523e-a800062f4ca7
[root@xenserver ~]#
```

Creating the vGPU object for a VM does not immediately cause a virtual GPU to be created on a physical GPU. Instead, the vGPU is created whenever its associated VM is started. For more details on how vGPUs are created at VM startup, see section 5.3.

Note: the owning VM must be in the powered-off state in order for the `vgpu-create` command to succeed.

A `vgpu` object's owning VM, associated GPU group, and vGPU type are fixed at creation and cannot be subsequently changed. To change the type of vGPU allocated to a VM, delete the existing `vgpu` object and create another one.

5.3 CONTROLLING VGPU ALLOCATION

Configuring a VM to use a vGPU in XenCenter, or creating a `vgpu` object for a VM using `xe`, does not immediately cause a virtual GPU to be created; rather, the virtual GPU is created at the time the VM is next booted, using the following steps:

- The GPU group that the `vgpu` object is associated with is checked for a physical GPU that can host a vGPU of the required type (i.e. the `vgpu` object's associated `vgpu-type`). Because vGPU types cannot be mixed on a single physical GPU, the new vGPU can only be created on a physical GPU that has no vGPUs resident on it, or only vGPUs of the same type, and less than the limit of vGPUs of that type that the physical GPU can support.
- If no such physical GPUs exist in the group, the `vgpu` creation fails and the VM startup is aborted.
- Otherwise, if more than one such physical GPU exists in the group, a physical GPU is selected according to the GPU group's *allocation policy*, as described here:

5.3.1 GPU allocation policy

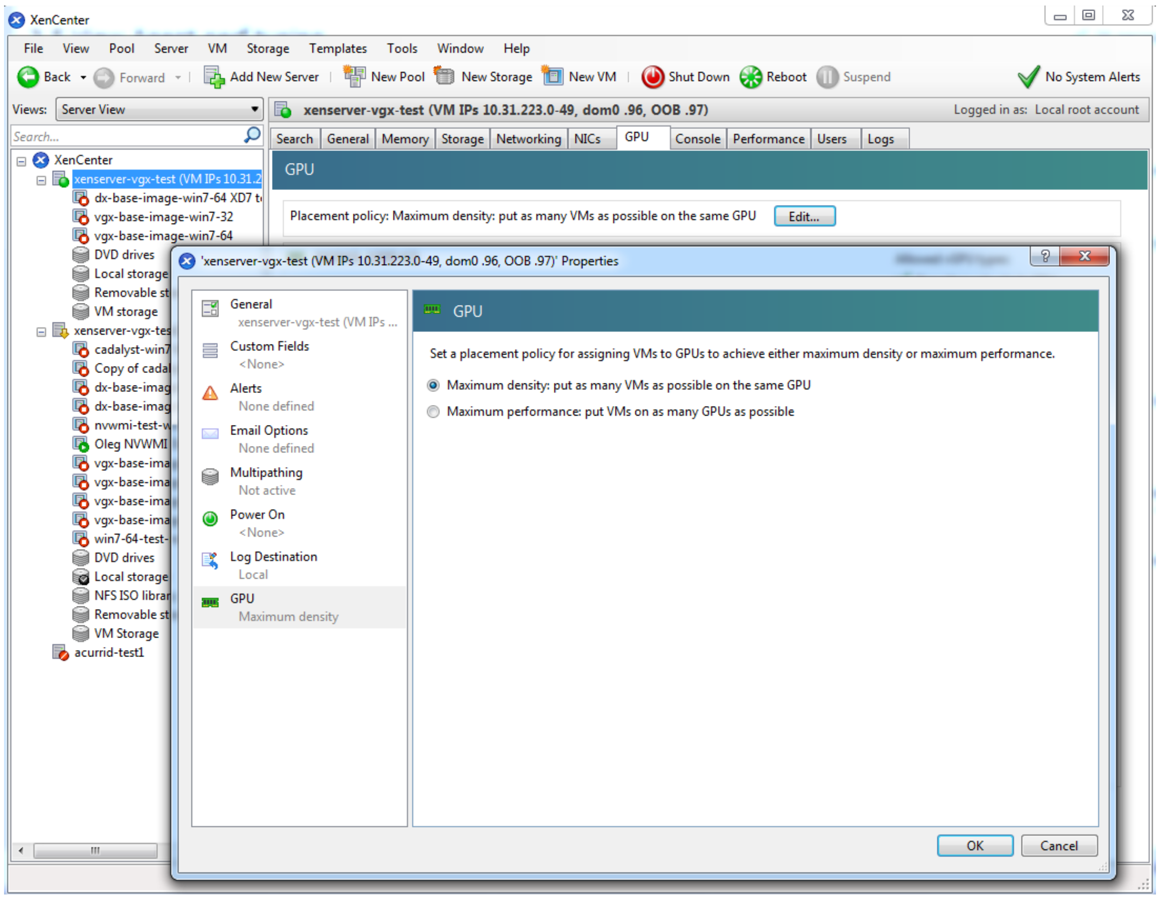
XenServer creates GPU groups with a default allocation policy of *depth-first*. The depth-allocation policy attempts to maximize the number of vGPUs running on each physical GPU within the group, by placing newly-created vGPUs on the physical GPU that can support the new vGPU and that has the most number of vGPUs already resident. This policy generally leads to higher density of vGPUs, particularly when different types of vGPUs are being run, but may result in lower performance because it attempts to maximize sharing of physical GPUs.

Conversely, a *breadth-first* allocation policy attempts to minimize the number of vGPUs running on each physical GPU within the group, by placing newly-created vGPUs on the physical GPU that can support the new vGPU and that has the least number of vGPUs already resident. This policy generally leads to higher performance because it attempts to minimize sharing of physical GPUs, but in doing so it may artificially limit the total number of vGPUs that can run.

The allocation policy of a GPU group is stored in the `allocation-algorithm` parameter of the `gpu-group` object, and can be changed using `gpu-group-param-set`:

```
[root@xenserver ~]# xe gpu-group-param-get uuid=be825ba2-01d7-8d51-9780-
f82cfaa64924 param-name=allocation-algorithm
depth-first
[root@xenserver ~]# xe gpu-group-param-set uuid=be825ba2-01d7-8d51-9780-
f82cfaa64924 allocation-algorithm=breadth-first
[root@xenserver ~]#
```

Allocation policy can also be controlled from the GPU tab in XenCenter:


XenCenter
File View Pool Server VM Storage Templates Tools Window Help
Back Forward Add New Server New Pool New Storage New VM Shut Down Reboot Suspend No System Alerts
Views: Server View
Search Search General Memory Storage Networking NICs GPU Console Performance Users Logs
Logged in as: Local root account
General
Custom Fields
Alerts
Email Options
Multipathing
Power On
Log Destination
GPU
OK
Cancel
Edit...

xenserver-vgx-test (VM IPs 10.31.223.0-49, dom0 .96, OOB .97)

GPU

Placement policy: Maximum density: put as many VMs as possible on the same GPU

'xenserver-vgx-test (VM IPs 10.31.223.0-49, dom0 .96, OOB .97)' Properties

General: xenserver-vgx-test (VM IPs ...
Custom Fields: <None>
Alerts: None defined
Email Options: None defined
Multipathing: Not active
Power On: <None>
Log Destination: Local
GPU: Maximum density

GPU

Set a placement policy for assigning VMs to GPUs to achieve either maximum density or maximum performance.

[✓ Maximum density: put as many VMs as possible on the same GPU]

Maximum performance: put VMs on as many GPUs as possible

Figure 19 Modifying GPU placement policy in XenCenter

5.3.2 Determining the physical GPU that a virtual GPU is resident on

The `vgpu` object's `resident-on` parameter returns the UUID of the `pgpu` object for the physical GPU the vGPU is resident on:

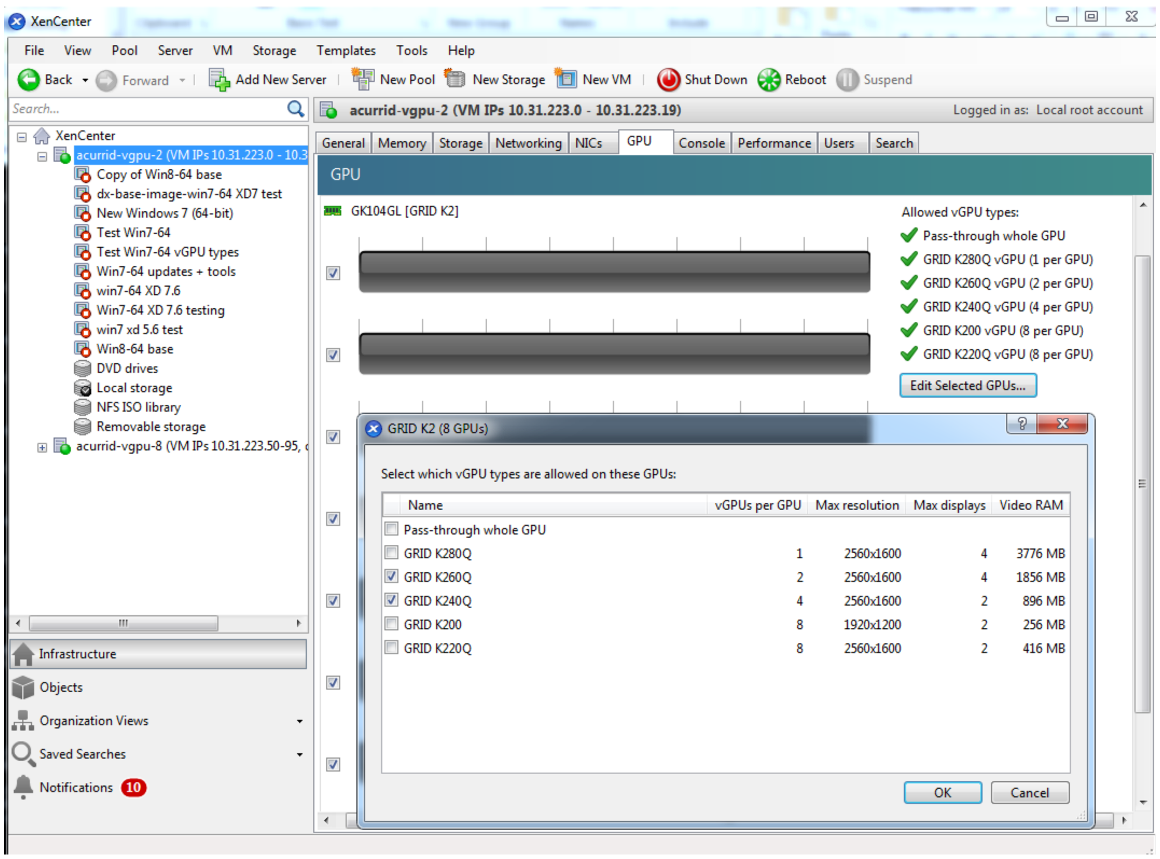
```
[root@xenserver ~]# xe vgpu-param-get uuid=8481cb68-66e5-25e6-a0c0-bd691df682b3
param-name=resident-on
a4a4df34-4e5f-de9f-82d6-2134d9e339dc

[root@xenserver ~]# xe pgpu-param-list uuid=a4a4df34-4e5f-de9f-82d6-
2134d9e339dc
uuid ( RO)                    : a4a4df34-4e5f-de9f-82d6-2134d9e339dc
             vendor-name ( RO): NVIDIA Corporation
             device-name ( RO): GK104GL [GRID K2]
          gpu-group-uuid ( RW): be825ba2-01d7-8d51-9780-f82cfaa64924
    gpu-group-name-label ( RO): Group of NVIDIA Corporation GK104GL [GRID K2]
GPUs
               host-uuid ( RO): 6f6209a6-0f11-4c51-b12d-2bce361e9639
```

```
         host-name-label ( RO): xenserver (VM IPs 10.31.213.50-95, dom0 .98,
OOB .99)
                  pci-id ( RO): 0000:09:00.0
            dependencies (SRO):
            other-config (MRW):
    supported-VGPU-types ( RO): fa50b0f0-9705-6c59-689e-ea62a3d35237; 1a312df9-
5397-bd44-c447-c6da804d2fe7; d1fb00dd-02e6-e7df-ccd5-1944965ece55; 3f318889-
7508-c9fd-7134-003d4d05ae56
      enabled-VGPU-types (SRW): fa50b0f0-9705-6c59-689e-ea62a3d35237; 1a312df9-
5397-bd44-c447-c6da804d2fe7; d1fb00dd-02e6-e7df-ccd5-1944965ece55; 3f318889-
7508-c9fd-7134-003d4d05ae56
          resident-VGPUs ( RO): 8481cb68-66e5-25e6-a0c0-bd691df682b3

[root@xenserver ~]#
```

Note: If the vGPU is not currently running, the `resident-on` parameter is not instantiated for the vGPU, and the `vgpu-param-get` operation returns `<not in database>`.

5.3.3 Controlling the vGPU types enabled on specific physical GPUs

Physical GPUs support several vGPU types, as defined in Table 1 on page 3, and the "passthrough" type that is used to assign an entire physical GPU to a VM (see section 5.5). To limit the types of vGPU that may be created on a specific vGPU, open the server's GPU tab in XenCenter, check the box beside one or more GPUs, and select Edit Selected GPUs:


XenCenter
File View Pool Server VM Storage Templates Tools Help
Back Forward Add New Server New Pool New Storage New VM Shut Down Reboot Suspend
acurrid-vgpu-2 (VM IPs 10.31.223.0 - 10.31.223.19)
Logged in as: Local root account
General Memory Storage Networking NICs GPU Console Performance Users Search
Infrastructure
Objects
Organization Views
Saved Searches
Notifications 10
Edit Selected GPUs...
OK
Cancel

GPU

GK104GL [GRID K2]

Allowed vGPU types:

- ✓ Pass-through whole GPU
- ✓ GRID K280Q vGPU (1 per GPU)
- ✓ GRID K260Q vGPU (2 per GPU)
- ✓ GRID K240Q vGPU (4 per GPU)
- ✓ GRID K200 vGPU (8 per GPU)
- ✓ GRID K220Q vGPU (8 per GPU)

GRID K2 (8 GPUs)

Select which vGPU types are allowed on these GPUs:

Name	vGPUs per GPU	Max resolution	Max displays	Video RAM
☐ Pass-through whole GPU				
☐ GRID K280Q	1	2560x1600	4	3776 MB
☑ GRID K260Q	2	2560x1600	4	1856 MB
☑ GRID K240Q	4	2560x1600	2	896 MB
☐ GRID K200	8	1920x1200	2	256 MB
☐ GRID K220Q	8	2560x1600	2	416 MB

Figure 20 Editing a GPU’s enabled vGPU types using XenCenter

Alternatively, modify the physical GPU’s `pgpu` object’s `enabled-vGPU-types` parameter use `xe pgpu-param-set`:

```
[root@xenserver ~]# xe pgpu-param-list uuid=f2607117-5b4c-d6cc-3900-
00bf712e33f4
uuid ( RO)                    : f2607117-5b4c-d6cc-3900-00bf712e33f4
             vendor-name ( RO): NVIDIA Corporation
             device-name ( RO): GK104GL [GRID K2]
          gpu-group-uuid ( RW): f4662c69-412c-abc5-6d02-f74b7703cccd
    gpu-group-name-label ( RO): GRID K2 Socket 0
               host-uuid ( RO): d9eb9118-a5c5-49fb-970e-80e6a8f7ff98
         host-name-label ( RO): xenserver-vgx-test (VM IPs 10.31.223.0-49, dom0
.96, OOB .97)
                  pci-id ( RO): 0000:08:00.0
            dependencies (SRO):
            other-config (MRW):
    supported-VGPU-types ( RO): a724b756-d108-4c9f-0ea3-8f3a1553bfbc; 63d9d912-
3454-b020-8519-58dedb3b0117; 0bdf4715-e035-19c3-a57d-5ead20b3e5cd; a7838abe-
0d73-1918-7d29-fd361d3e411f
      enabled-VGPU-types (SRW): a724b756-d108-4c9f-0ea3-8f3a1553bfbc; 63d9d912-
3454-b020-8519-58dedb3b0117; 0bdf4715-e035-19c3-a57d-5ead20b3e5cd; a7838abe-
0d73-1918-7d29-fd361d3e411f
          resident-VGPUs ( RO):
```

```
[root@xenserver-vgx-test ~]# xe pgpu-param-set uuid=f2607117-5b4c-d6cc-3900-
00bf712e33f4 enabled-VGPU-types=a724b756-d108-4c9f-0ea3-8f3a1553bfbc
```

5.3.4 Creating vGPUs on specific physical GPUs

To precisely control allocation of vGPUs on specific physical GPUs, create separate GPU groups for the physical GPUs you wish to allocate vGPUs on. When creating a virtual GPU, create it on the GPU group containing the physical GPU you want it to be allocated on.

For example, to create a new GPU group for the physical GPU at PCI bus ID `0000:05:0.0`, start by creating the new GPU group with an appropriate name:

```
[root@xenserver ~]# xe gpu-group-create name-label="GRID K2 5:0.0"
585877ef-5a6c-66af-fc56-7bd525bdc2f6
[root@xenserver ~]#
```

Next, find the UUID of the physical GPU at `0000:05:0.0` that you wish to assign to the new GPU group

```
[root@xenserver ~]# xe pgpu-list pci-id=0000:05:00.0
uuid ( RO)               : 7c1e3cff-1429-0544-df3d-bf8a086fb70a
       vendor-name  ( RO): NVIDIA Corporation
       device-name  ( RO): GK104GL [GRID K2]
    gpu-group-uuid  ( RW): be825ba2-01d7-8d51-9780-f82cfaa64924

[root@xenserver ~]
```

Note: the pci-id parameter passed to the pgpu-list command must be in the exact format shown, with the PCI domain fully specified (e.g. `0000`) and the PCI bus and devices numbers each being two digits (e.g. `05:00.0`).

Ensure that no vGPUs are currently operating on the physical GPU by checking the `resident-VGPUs` parameter:

```
[root@xenserver ~]# xe pgpu-param-get uuid=7c1e3cff-1429-0544-df3d-bf8a086fb70a
param-name=resident-VGPUs

[root@xenserver ~]#
```

If any vGPUs are listed, shut down the VMs associated with them.

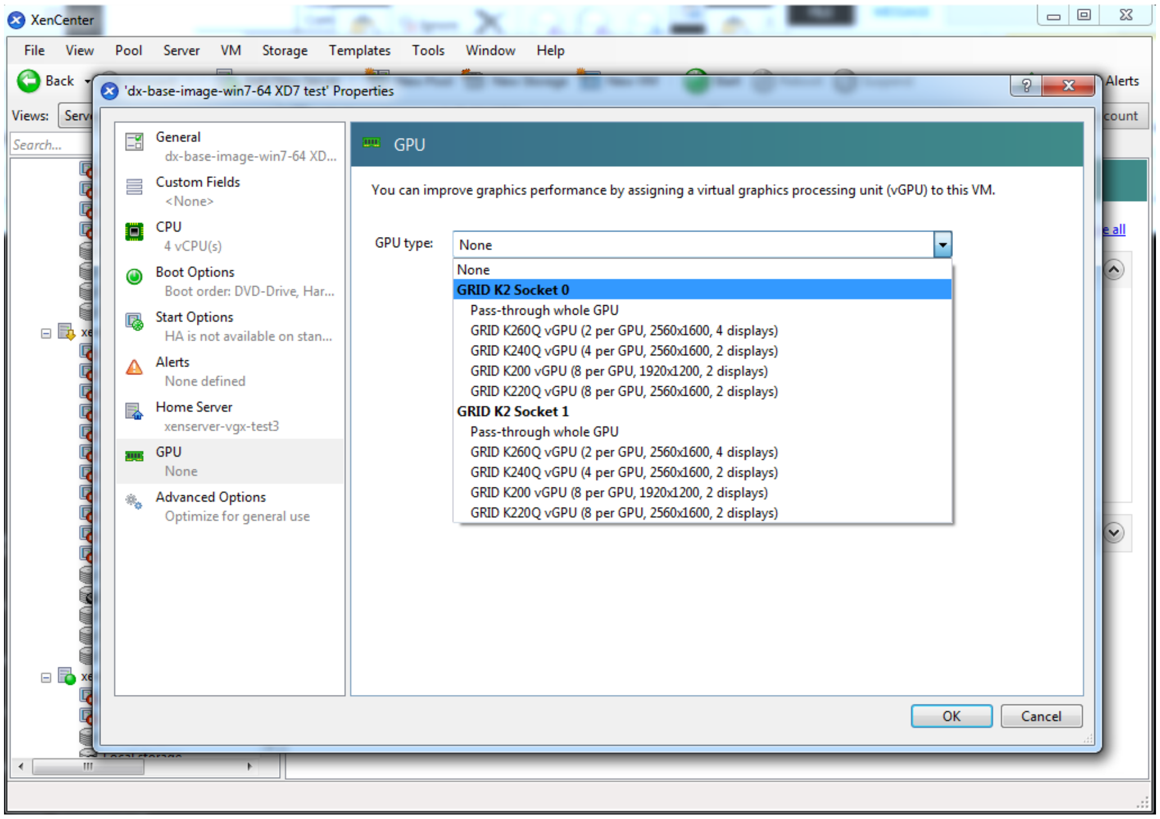
Now change the `gpu-group-uuid` parameter of the physical GPU to the UUID of the newly-created GPU group:

```
[root@xenserver ~]# xe pgpu-param-set uuid=7c1e3cff-1429-0544-df3d-bf8a086fb70a
gpu-group-uuid=585877ef-5a6c-66af-fc56-7bd525bdc2f6
[root@xenserver ~]#
```

Any `vgpu` object now created that specifies this GPU group UUID will always have its vGPUs created on the GPU at PCI bus ID `0000:05:0.0`.

Note: you can add more than one physical GPU to a manually-created GPU group - for example, to represent all the GPUs attached to the same CPU socket in a multi-socket server platform - but as for automatically-created GPU groups, all the physical GPUs in the group must of the same type.

In XenCenter, manually-created GPU groups appear in the GPU type listing in a VM's GPU Properties. Select a GPU type within the group from which you wish the vGPU to be allocated:


XenCenter
File View Pool Server VM Storage Templates Tools Window Help
Back
'dx-base-image-win7-64 XD7 test' Properties
General
dx-base-image-win7-64 XD...
Custom Fields
<None>
CPU
4 vCPU(s)
Boot Options
Boot order: DVD-Drive, Har...
Start Options
HA is not available on stan...
Alerts
None defined
Home Server
xenserver-vgx-test3
GPU
None
Advanced Options
Optimize for general use
GPU
OK
Cancel

You can improve graphics performance by assigning a virtual graphics processing unit (vGPU) to this VM.

GPU type: None

- None
- **GRID K2 Socket 0**
  - Pass-through whole GPU
  - GRID K260Q vGPU (2 per GPU, 2560x1600, 4 displays)
  - GRID K240Q vGPU (4 per GPU, 2560x1600, 2 displays)
  - GRID K200 vGPU (8 per GPU, 1920x1200, 2 displays)
  - GRID K220Q vGPU (8 per GPU, 2560x1600, 2 displays)
- **GRID K2 Socket 1**
  - Pass-through whole GPU
  - GRID K260Q vGPU (2 per GPU, 2560x1600, 4 displays)
  - GRID K240Q vGPU (4 per GPU, 2560x1600, 2 displays)
  - GRID K200 vGPU (8 per GPU, 1920x1200, 2 displays)
  - GRID K220Q vGPU (8 per GPU, 2560x1600, 2 displays)

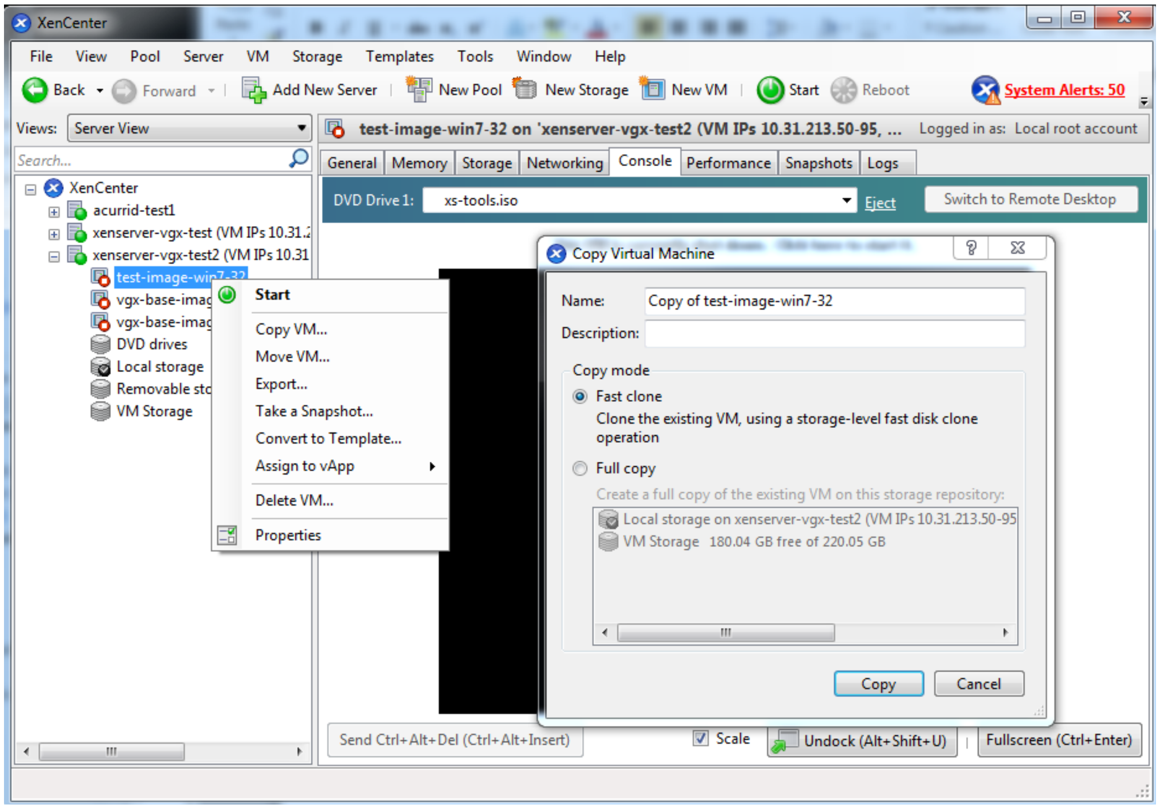
Figure 21 Using a custom GPU group within XenCenter

5.4 CLONING VGPU-ENABLED VMS

XenServer's fast-clone or copying feature can be used to rapidly create new VMs from a "golden" base VM image that has been configured with GRID vGPU, the NVIDIA driver, applications, and remote graphics software. Cloning/copying can be initiated via XenCenter (see Figure 22) or from the dom0 shell:

```
[root@xenserver ~]# xe vm-clone new-name-label="new-vm" vm="base-vm-name"
7f7035cb-388d-1537-1465-1857fb6498e7
[root@xenserver ~]#
```

When a VM is cloned, any vGPU configuration associated with the base VM is copied to the cloned VM. Starting the cloned VM will create a vGPU instance of the same type as the original VM, from the same GPU group as the original vGPU.


XenCenter
File View Pool Server VM Storage Templates Tools Window Help
Back Forward Add New Server New Pool New Storage New VM Start Reboot
System Alerts: 50
Views: Server View
Search...
XenCenter
acurrid-test1
xenserver-vgx-test (VM IPs 10.31.2
xenserver-vgx-test2 (VM IPs 10.31
test-image-win7-32
vgx-base-imag
vgx-base-imag
DVD drives
Local storage
Removable sto
VM Storage
Start
Copy VM...
Move VM...
Export...
Take a Snapshot...
Convert to Template...
Assign to vApp
Delete VM...
Properties
test-image-win7-32 on 'xenserver-vgx-test2 (VM IPs 10.31.213.50-95, ...
Logged in as: Local root account
General Memory Storage Networking Console Performance Snapshots Logs
DVD Drive 1: xs-tools.iso
Eject
Switch to Remote Desktop
Copy Virtual Machine
Send Ctrl+Alt+Del (Ctrl+Alt+Insert)
Scale
Undock (Alt+Shift+U)
Fullscreen (Ctrl+Enter)
Copy
Cancel

Name: Copy of test-image-win7-32

Description:

Copy mode

[✓ Fast clone]

Clone the existing VM, using a storage-level fast disk clone operation

Full copy

Create a full copy of the existing VM on this storage repository:

Local storage on xenserver-vgx-test2 (VM IPs 10.31.213.50-95

VM Storage 180.04 GB free of 220.05 GB

Figure 22 Cloning a VM using XenCenter

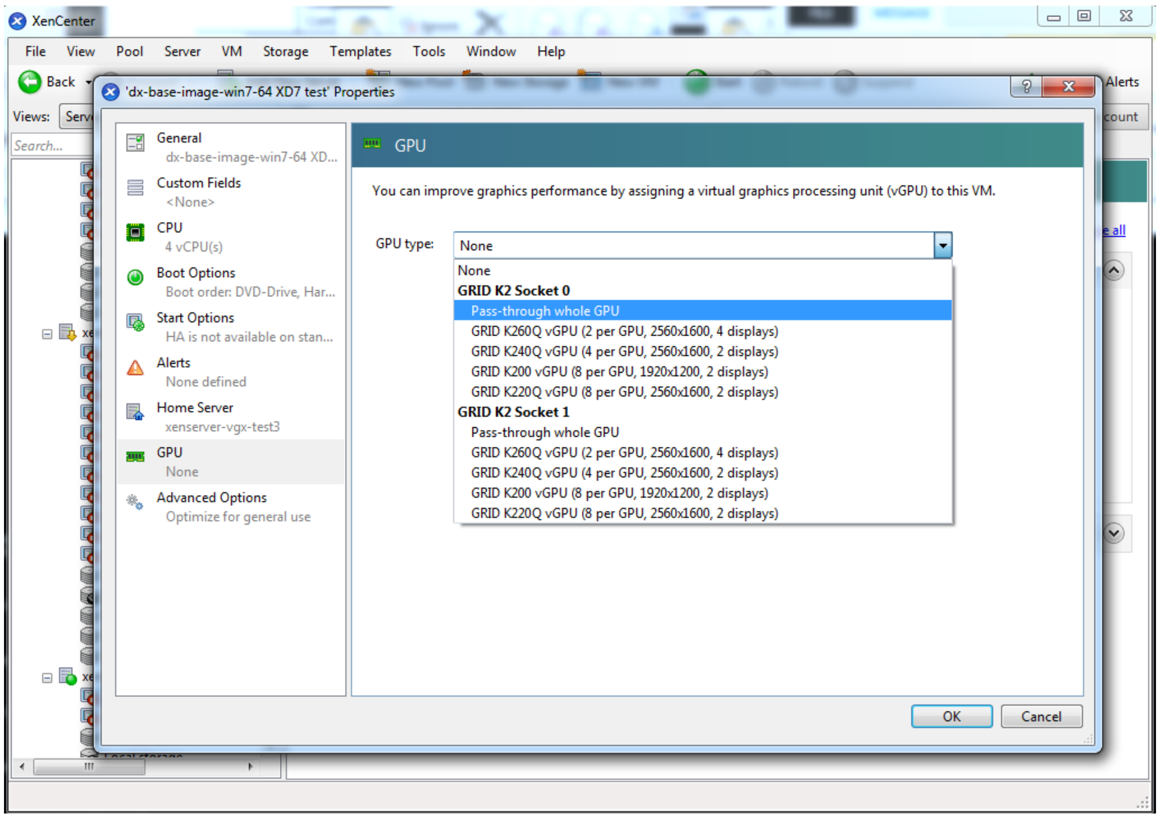
5.5 USING GPU PASS-THROUGH

GPU pass-through is used to directly assign an entire physical GPU to one VM, bypassing the GRID Virtual GPU Manager. In this mode of operation, the GPU is accessed exclusively by the NVIDIA driver running in the VM to which it is assigned; the GPU is not shared among VMs.

GPU pass-through can be used in a server platform alongside GRID vGPU, with some restrictions:

- A physical GPU can host GRID vGPUs, or can be used for pass-through, but cannot do both at the same time.
- A physical GPU passed through to a VM cannot be performance-monitored via XenCenter or nvidia-smi (see Chapter 4)
- Passthrough GPUs do not provide console output via XenCenter's VM Console tab. Use a remote graphics connection directly into the VM to access the VM's OS.

To configure a VM for GPU pass-through, select the "Pass-through whole GPU" option as the GPU type in the VM's Properties:


XenCenter
File View Pool Server VM Storage Templates Tools Window Help
Back
Views:
Search...
Alerts
OK
Cancel

'dx-base-image-win7-64 XD7 test' Properties

General: dx-base-image-win7-64 XD...
Custom Fields: <None>
CPU: 4 vCPU(s)
Boot Options: Boot order: DVD-Drive, Har...
Start Options: HA is not available on stan...
Alerts: None defined
Home Server: xenserver-vgx-test3
GPU: None
Advanced Options: Optimize for general use

GPU

You can improve graphics performance by assigning a virtual graphics processing unit (vGPU) to this VM.

GPU type: None

None
GRID K2 Socket 0
Pass-through whole GPU
GRID K260Q vGPU (2 per GPU, 2560x1600, 4 displays)
GRID K240Q vGPU (4 per GPU, 2560x1600, 2 displays)
GRID K200 vGPU (8 per GPU, 1920x1200, 2 displays)
GRID K220Q vGPU (8 per GPU, 2560x1600, 2 displays)
GRID K2 Socket 1
Pass-through whole GPU
GRID K260Q vGPU (2 per GPU, 2560x1600, 4 displays)
GRID K240Q vGPU (4 per GPU, 2560x1600, 2 displays)
GRID K200 vGPU (8 per GPU, 1920x1200, 2 displays)
GRID K220Q vGPU (8 per GPU, 2560x1600, 2 displays)

Figure 23 Using XenCenter to configure a passthrough GPU

Alternatively, create a `vgpu` object with the "passthrough" vGPU type:

```
[root@xenserver ~]# xe vgpu-type-list model-name="passthrough"
uuid ( RO)                  : fa50b0f0-9705-6c59-689e-ea62a3d35237
         vendor-name ( RO):
          model-name ( RO): passthrough
```

```
    framebuffer-size ( RO): 0

[root@xenserver ~]# xe vgpu-create vm-uuid=753e77a9-e10d-7679-f674-65c078abb2eb
vgpu-type-uuid=fa50b0f0-9705-6c59-689e-ea62a3d35237 gpu-group-uuid=585877ef-
5a6c-66af-fc56-7bd525bdc2f6
6aa530ec-8f27-86bd-b8e4-fe4fde8f08f9
[root@xenserver ~]#
```

! **CAUTION:** Do not assign passthrough GPUs using the legacy `other-config:pci` parameter setting. This mechanism is not supported alongside the XenCenter UI and `xe vgpu` mechanisms, and attempts to use it may lead to undefined results.

Chapter 6. XENSERVER PERFORMANCE TUNING

This chapter provides recommendations on optimizing performance for VMs running with GRID vGPU on Citrix XenServer.

6.1 XENSERVER TOOLS

To get maximum performance out of a VM running on Citrix XenServer, regardless of whether you are using GRID vGPU, you must install Citrix XenServer tools within the VM. Without the optimized networking and storage drivers that the XenServer tools provide, remote graphics applications running on GRID vGPU will not deliver maximum performance.

6.2 USING REMOTE GRAPHICS

GRID vGPU implements a console VGA interface that permits the VM's graphics output to be viewed via XenCenter's console tab. This feature allows the desktop of a vGPU-enabled VM to be visible in XenCenter before any NVIDIA graphics driver is loaded in the virtual machine, but it is intended solely as a management convenience; it only supports output of vGPU's primary display and isn't designed or optimized to deliver high frame rates.

To deliver high frames from multiple heads on vGPU, we recommend installation of a high-performance remote graphics stack such as Citrix XenDesktop® with HDX 3D Pro remote graphics and, once this is done, disable vGPU's console VGA.

! **CAUTION:** Using Windows Remote Desktop (RDP) to access Windows 7 / Windows Server 2008 VMs running GRID vGPU will cause the NVIDIA driver in the VM to be unloaded. GPU-accelerated DirectX, OpenGL, and the NVIDIA control panel will be unavailable whenever RDP is active. Installing a VNC server in the VM will allow for basic, low-performance remote access while leaving the NVIDIA driver loaded and vGPU active, but for high performance remote accesses, use an accelerated stack such as XenDesktop.

6.2.1 Disabling console VGA

The console VGA interface in vGPU is optimized to consume minimal resources, but when a system is loaded with a high number of VMs, disabling the console VGA interface entirely may yield some performance benefit.

Once you have installed an alternate means of accessing a VM (such as XenDesktop or a VNC server), its vGPU console VGA interface can be disabled by specifying `disable_vnc=1` in the VM's `platform:vgpu_extra_args` parameter:

```
[root@xenserver ~]# xe vm-param-set uuid=e71afda4-53f4-3a1b-6c92-a364a7f619c2
platform:vgpu_extra_args="disable_vnc=1"
[root@xenserver ~]#
```

The new console VGA setting takes effect the next time the VM is started or rebooted. With console VGA disabled, the XenCenter console will display the Windows boot splash screen for the VM, but nothing beyond that.

! **CAUTION:** If you disable console VGA before you have installed/enabled an alternate mechanism to access the VM (such as XenDesktop), you will not be able to interact with the VM once it has booted.

You can recover console VGA access by removing the `vgpu_extra_args` key from the `platform` parameter, or by removing `disable_vnc=1` from the `vgpu_extra_args` key, or by setting `disable_vnc=0`. For example:

```
[root@xenserver ~]# xe vm-param-set uuid=e71afda4-53f4-3a1b-6c92-a364a7f619c2
platform:vgpu_extra_args="disable_vnc=0"
```

6.4 ALLOCATION STRATEGIES

6.4.1 NUMA considerations

Server platforms typically implement multiple CPU sockets, with system memory and PCI Express expansion slots local to each CPU socket, as illustrated in Figure 24:

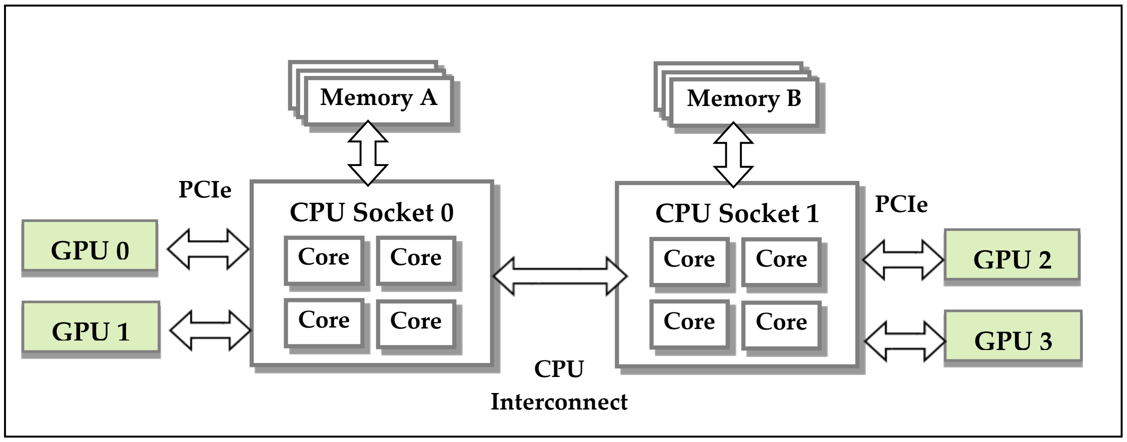


Figure 24 A NUMA server platform

These platforms are typically configured to operate in Non-Uniform Memory Access (NUMA) mode; physical memory is arranged sequentially in the address space, with all the memory attached to each socket appearing in a single contiguous block of addresses. The cost of accessing a range of memory from a CPU or GPU varies; memory attached to the same socket as the CPU or GPU is accessible at lower latency than memory on another CPU socket, because accesses to remote memory must additionally traverse the interconnect between CPU sockets.

To obtain best performance on a NUMA platform, we recommend pinning VM vCPU cores to physical cores on the same CPU socket to which the physical GPU hosting the VM's vGPU is attached. For example, using as a reference, a VM with a vGPU allocated on physical GPU 0 or 1 should have its vCPUs pinned to CPU cores on CPU socket 0. Similarly, a VM with a vGPU allocated on physical GPU 2 or 3 should have its vCPUs pinned to CPU cores on socket 1.

See Appendix A.5 for guidance on pinning vCPUs, and A.7 for guidance on determining which CPU socket a GPU is connected to. Section 5.3.3 describes how to precisely control which physical GPU is used to host a vGPU, by creating GPU groups for specific physical GPUs.

6.4.2 Maximizing performance

To maximize performance as the number of vGPU-enabled VMs on the platform increases, we recommend adopting a *breadth-first* allocation: allocate new VMs on the least-loaded CPU socket, and allocate the VM's vGPU on an available, least-loaded, physical GPU connected via that socket.

XenServer's creates GPU groups with a default allocation policy of *depth-first*. See section 5.3.1 for details on switching the allocation policy to breadth-first.

Note: Due to vGPU's requirement that only one type of vGPU can run on a physical GPU at any given time, not all physical GPUs may be available to host the vGPU type required by the new VM.

Chapter 7. TROUBLESHOOTING

This chapter describes basic troubleshooting steps for GRID vGPU on Citrix XenServer and VMware vSphere, and how to collect debug information when filing a bug report.

7.1 KNOWN ISSUES

Before troubleshooting or filing a bug report, review the release notes that accompany each driver release, for information about known issues with the current release, and potential workarounds.

7.2 TROUBLESHOOTING STEPS

If a vGPU-enabled VM fails to start, or doesn't display any output when it does start, follow these steps to narrow down the probable cause.

7.2.1 Verify the NVIDIA kernel driver is loaded

On Citrix XenServer, use `lsmod` to verify that the kernel driver is loaded:

```
[root@xenserver ~]# lsmod|grep nvidia
nvidia               9604895  84
i2c_core               20294  2 nvidia,i2c_i801
[root@xenserver ~]#
```

On VMware vSphere, use `vmkload_mod`:

```
[root@esxi:~] vmkload_mod -l | grep nvidia
nvidia                     5    8420
```

If the `nvidia` driver is not listed in the output, check `dmesg` for any load-time errors reported by the driver (see section 7.2.3). On XenServer, also use the '`rpm -q`' command to verify that the NVIDIA GPU Manager package is correctly installed (see section 2.1.3.2).

7.2.2 Verify that nvidia-smi works

If the NVIDIA kernel driver is correctly loaded on the physical GPU, run `nvidia-smi` and verify that all physical GPUs not currently being used for GPU passthrough are listed in the output. For details on expected output, see section 4.1.

If `nvidia-smi` fails to report the expected output, check `dmesg` for NVIDIA kernel driver messages.

7.2.3 NVIDIA kernel driver output

Information and debug messages from the NVIDIA kernel driver are logged in kernel logs, prefixed with "`NVRM`" or '`nvidia`'. Run `dmesg` on both Citrix XenServer and VMware vSphere to check for these:

```
[root@xenserver ~]# dmesg | grep -E "NVRM|nvidia"
[   22.054928] nvidia: module license 'NVIDIA' taints kernel.
[   22.390414] NVRM: loading
[   22.829226] nvidia 0000:04:00.0: enabling device (0000 -> 0003)
[   22.829236] nvidia 0000:04:00.0: PCI INT A -> GSI 32 (level, low) -> IRQ 32
[   22.829240] NVRM: This PCI I/O region assigned to your NVIDIA device is
invalid:
[   22.829241] NVRM: BAR0 is 0M @ 0x0 (PCI:0000:00:04.0)
[   22.829243] NVRM: The system BIOS may have misconfigured your GPU.
```

7.2.4 GRID Virtual GPU Manager messages

Information and debug messages from the GRID Virtual GPU Manager are logged to the hypervisor's log files, prefixed with '`vmiop`':

7.2.4.1 Citrix XenServer

GRID Virtual GPU Manager are written to `/var/log/messages`:

```
[root@xenserver ~]# grep vmiop /var/log/messages
```

```
Nov  8 09:17:44 xenserver-vgx-test2 fe: vgpu-2[14901]: vmiop_log: notice: vmiop-env:
guest_max_gpfn:0x10efff
Nov  8 09:17:44 xenserver-vgx-test2 fe: vgpu-2[14901]: vmiop_log: notice: pluginconfig:
/usr/share/nvidia/vgx/grid_k100.conf,disable_vnc=0,gpu-pci-id=0000:88:00.0
Nov  8 09:17:44 xenserver-vgx-test2 fe: vgpu-2[14901]: vmiop_log: notice: Loading Plugin0: libnvidia-
vgx
Nov  8 09:17:45 xenserver-vgx-test2 fe: vgpu-2[14901]: vmiop_log: notice: vgpu_type : vdi
Nov  8 09:17:45 xenserver-vgx-test2 fe: vgpu-2[14901]: vmiop_log: notice: Framebuffer: 0x10000000
Nov  8 09:17:45 xenserver-vgx-test2 fe: vgpu-2[14901]: vmiop_log: notice: Virtual Device Id:
0x0FE7:0x101E
Nov  8 09:17:45 xenserver-vgx-test2 fe: vgpu-2[14901]: vmiop_log: notice: ######## vGPU Manager
Information: ########
Nov  8 09:17:45 xenserver-vgx-test2 fe: vgpu-2[14901]: vmiop_log: notice: Driver Version: 331.59
Nov  8 09:17:45 xenserver-vgx-test2 fe: vgpu-2[14901]: vmiop_log: notice: VGX Version: 1.1
Nov  8 09:17:45 xenserver-vgx-test2 fe: vgpu-2[14901]: vmiop_log: notice: Init frame copy engine:
syncing...
Nov  8 09:18:03 xenserver-vgx-test2 fe: vgpu-2[14901]: vmiop_log: notice: ######## Guest NVIDIA
Driver Information: ########
Nov  8 09:18:03 xenserver-vgx-test2 fe: vgpu-2[14901]: vmiop_log: notice: Driver Version: 331.82
Nov  8 09:18:03 xenserver-vgx-test2 fe: vgpu-2[14901]: vmiop_log: notice: VGX Version: 0.5
[root@xenserver ~]#
```

7.2.4.2 VMware vSphere

GRID Virtual GPU Manager are written to `vmware.log` in the guest VM's storage directory:

```
[root@esxi:~] grep vmiop /vmfs/volumes/datastore1/win7-vgpu-test1/vmware.log
2015-04-28T14:02:21.275Z| vmx| I120: DICT   pciPassthru0.virtualDev = "vmiop"
2015-04-28T14:02:21.344Z| vmx| I120: GetPluginPath testing /usr/lib64/vmware/plugin/libvmx-vmiop.so
2015-04-28T14:02:21.344Z| vmx| I120: PluginLdr_LoadShared: Loaded shared plugin libvmx-vmiop.so from
/usr/lib64/vmware/plugin/libvmx-vmiop.so
2015-04-28T14:02:21.344Z| vmx| I120: VMIOP: Loaded plugin libvmx-vmiop.so:VMIOP_InitModule
2015-04-28T14:02:21.359Z| vmx| I120: VMIOP: Initializing plugin vmiop-display
2015-04-28T14:02:21.365Z| vmx| I120: vmiop_log: gpu-pci-id : 0000:04:00.0
2015-04-28T14:02:21.365Z| vmx| I120: vmiop_log: vgpu_type : quadro
2015-04-28T14:02:21.365Z| vmx| I120: vmiop_log: Framebuffer: 0x74000000
2015-04-28T14:02:21.365Z| vmx| I120: vmiop_log: Virtual Device Id: 0x11B0:0x101B
2015-04-28T14:02:21.365Z| vmx| I120: vmiop_log: ######## vGPU Manager Information: ########
2015-04-28T14:02:21.365Z| vmx| I120: vmiop_log: Driver Version: 346.42
2015-04-28T14:02:21.365Z| vmx| I120: vmiop_log: VGX Version: 2.0
2015-04-28T14:02:21.445Z| vmx| I120: vmiop_log: Init frame copy engine: syncing...
2015-04-28T14:02:37.031Z| vthread-12| I120: vmiop_log: ######## Guest NVIDIA Driver Information:
########
2015-04-28T14:02:37.031Z| vthread-12| I120: vmiop_log: Driver Version: 347.52
2015-04-28T14:02:37.031Z| vthread-12| I120: vmiop_log: VGX Version: 2.0
2015-04-28T14:02:37.093Z| vthread-12| I120: vmiop_log: Clearing BAR1 mapping
2015-07-06T23:39:55.726Z| vmx| I120: VMIOP: Shutting down plugin vmiop-display
[root@esxi:~]
```

7.3 FILING A BUG REPORT

When filing a bug report with NVIDIA, capture relevant configuration data from the platform exhibiting the bug, using `nvidia-bug-report.sh`. On Citrix XenServer, a server status report may be used as an alternative.

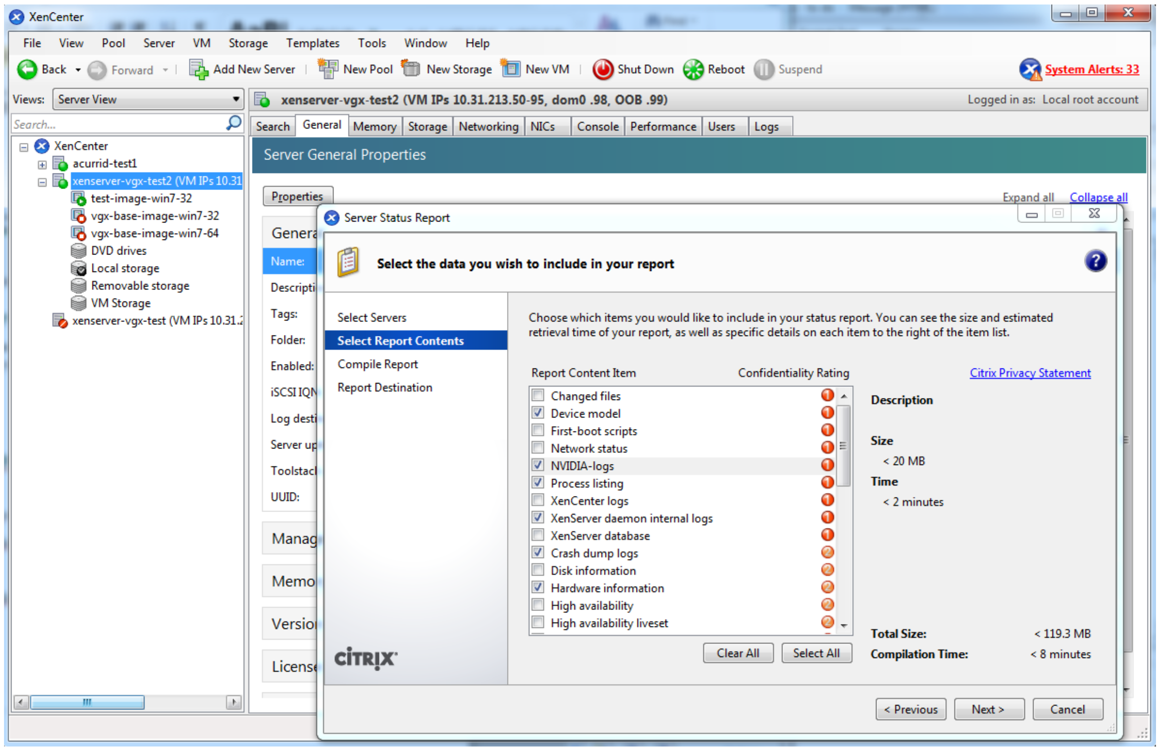
7.3.1 nvidia-bug-report.sh

Run `nvidia-bug-report.sh` from Citrix XenServer's dom0 shell or VMware ESXi's host shell to capture debug information into a gzip'd log file on the server. The example shows execution on Citrix XenServer, but the procedure is the same on vSphere ESXi:

```
[root@xenserver ~]# nvidia-bug-report.sh

nvidia-bug-report.sh will now collect information about your
system and create the file 'nvidia-bug-report.log.gz' in the current
directory.  It may take several seconds to run.  In some
cases, it may hang trying to capture data generated dynamically
by the Linux kernel and/or the NVIDIA kernel module.  While
the bug report log file will be incomplete if this happens, it
may still contain enough data to diagnose your problem.

For Xen open source/XCP users, if you are reporting a domain issue,
please run: nvidia-bug-report.sh --domain-name <"domain_name">

Please include the 'nvidia-bug-report.log.gz' log file when reporting
your bug via the NVIDIA Linux forum (see devtalk.nvidia.com)
or by sending email to 'linux-bugs@nvidia.com'.

Running nvidia-bug-report.sh...

If the bug report script hangs after this point consider running with
--safe-mode command line argument.

 complete.

[root@xenserver ~]#
```

7.3.2 XenServer status report

From XenCenter, select the Tools menu, Server Status Report, then select the XenServer instance from which you wish to collect a status report. Select the data to include in the report, check "NVIDIA-logs" to include GRID vGPU debug information, then generate the report.


XenCenter
File View Pool Server VM Storage Templates Tools Window Help
Back Forward Add New Server New Pool New Storage New VM Shut Down Reboot Suspend
System Alerts: 33
Views: Server View
xenserver-vgx-test2 (VM IPs 10.31.213.50-95, dom0 .98, OOB .99)
Logged in as: Local root account
Search General Memory Storage Networking NICs Console Performance Users Logs
XenCenter
acurrid-test1
xenserver-vgx-test2 (VM IPs 10.31
test-image-win7-32
vgx-base-image-win7-32
vgx-base-image-win7-64
DVD drives
Local storage
Removable storage
VM Storage
xenserver-vgx-test (VM IPs 10.31.2
Server General Properties
Properties
Expand all Collapse all
Server Status Report
Select Servers
Select Report Contents
Compile Report
Report Destination
Clear All
Select All
< Previous
Next >
Cancel

Select the data you wish to include in your report

Choose which items you would like to include in your status report. You can see the size and estimated retrieval time of your report, as well as specific details on each item to the right of the item list.

Report Content Item	Confidentiality Rating
☐ Changed files	●
☑ Device model	●
☐ First-boot scripts	●
☐ Network status	●
☑ NVIDIA-logs	●
☑ Process listing	●
☐ XenCenter logs	●
☑ XenServer daemon internal logs	●
☐ XenServer database	●
☑ Crash dump logs	●
☐ Disk information	●
☑ Hardware information	●
☐ High availability	●
☐ High availability liveset	●

Citrix Privacy Statement

Description

Size
< 20 MB

Time
< 2 minutes

Total Size: < 119.3 MB
Compilation Time: < 8 minutes

Figure 25 Including NVIDIA logs in server status report

Appendix A. XENSERVER BASICS

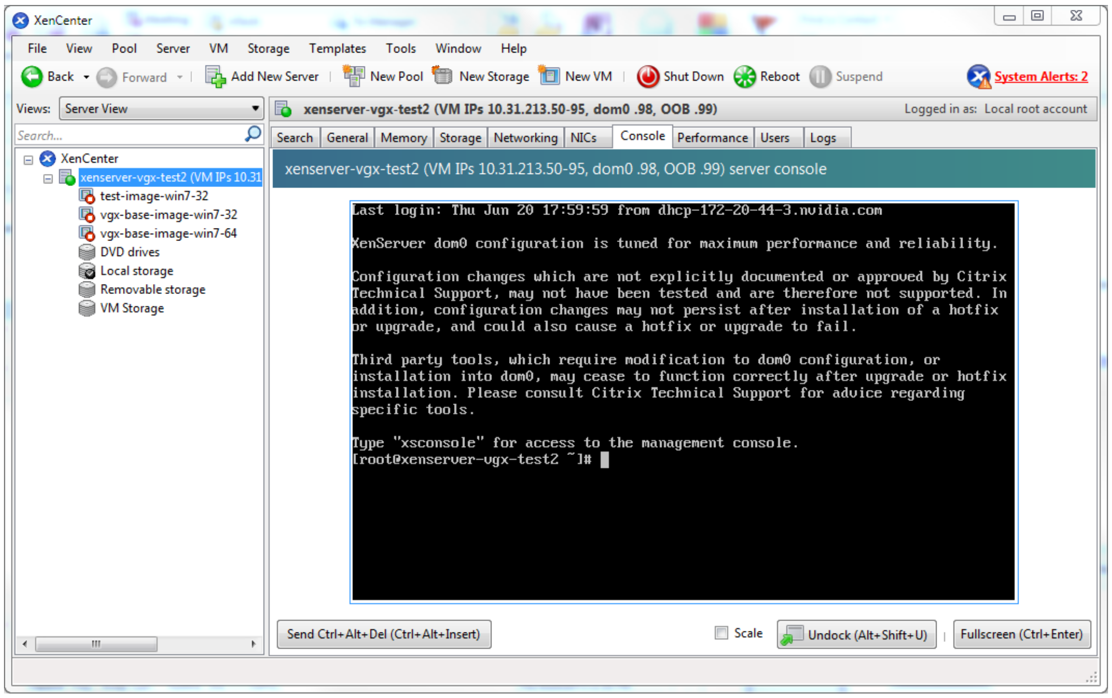
This appendix outlines basic operations on XenServer that are needed in order to install and configure GRID vGPU, and optimize XenServer operation with vGPU.

A.1 OPENING A DOM0 SHELL

Most configuration commands must be run in a command shell on XenServer's dom0. There are two ways to open a shell on XenServer's dom0; using the console window in XenCenter, or using a standalone secure shell (ssh) client:

A.1.1 Accessing the dom0 shell via XenCenter

To access the dom0 shell via XenCenter, in the left-hand pane click on the XenServer host you wish to connect to. Then click on the Console tab to open the XenServer's console, and press enter to start a shell prompt:


XenCenter
File View Pool Server VM Storage Templates Tools Window Help
Back Forward Add New Server New Pool New Storage New VM Shut Down Reboot Suspend
System Alerts: 2
Views: Server View
xenserver-vgx-test2 (VM IPs 10.31.213.50-95, dom0 .98, OOB .99)
Logged in as: Local root account
Search...
XenCenter
xenserver-vgx-test2 (VM IPs 10.31
test-image-win7-32
vgx-base-image-win7-32
vgx-base-image-win7-64
DVD drives
Local storage
Removable storage
VM Storage
Search General Memory Storage Networking NICs Console Performance Users Logs
xenserver-vgx-test2 (VM IPs 10.31.213.50-95, dom0 .98, OOB .99) server console
Send Ctrl+Alt+Del (Ctrl+Alt+Insert)
Scale
Undock (Alt+Shift+U)
Fullscreen (Ctrl+Enter)

```
Last login: Thu Jun 20 17:59:59 from dhcp-172-20-44-3.nvidia.com

XenServer dom0 configuration is tuned for maximum performance and reliability.

Configuration changes which are not explicitly documented or approved by Citrix
Technical Support, may not have been tested and are therefore not supported. In
addition, configuration changes may not persist after installation of a hotfix
or upgrade, and could also cause a hotfix or upgrade to fail.

Third party tools, which require modification to dom0 configuration, or
installation into dom0, may cease to function correctly after upgrade or hotfix
installation. Please consult Citrix Technical Support for advice regarding
specific tools.

Type "xsconsole" for access to the management console.
[root@xenserver-vgx-test2 ~]#
```

Figure 26 Connecting to the dom0 shell via XenCenter

A.1.2 Accessing the dom0 shell using ssh

To access the dom0 shell via an ssh client, you will need an ssh client suite such as *putty* on Windows, or the ssh client from OpenSSH on Linux.

Connect your ssh client to the management IP address of the XenServer, and log in as the root user.

A.2 COPYING FILES TO DOM0

Files can be easily copied to/from XenServer dom0 using an scp client or using a network-mounted filesystem.

A.2.1 Copying files using scp

Copying files using scp

scp is a secure copy program that is part of the ssh suite of applications. scp is implemented in dom0 and can be used to copy from a remote ssh-enabled server:

```
[root@xenserver ~]# scp root@10.31.213.96:/tmp/somefile .
The authenticity of host '10.31.213.96 (10.31.213.96)' can't be established.
RSA key fingerprint is 26:2d:9b:b9:bf:6c:81:70:36:76:13:02:c1:82:3d:3c.
```

```
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.31.213.96' (RSA) to the list of known hosts.
root@10.31.213.96's password:
somefile                                     100%  532     0.5KB/s   00:00
[root@xenserver ~]#
```

Alternatively, scp can be used to copy files from a remote system to XenServer. Using the `pscp` program from the putty suite on Windows:

```
C:\Users\nvidia>pscp somefile root@10.31.213.98:/tmp
root@10.31.213.98's password:
somefile               | 80 kB |  80.1 kB/s | ETA: 00:00:00 | 100%

C:\Users\nvidia>
```

A.2.2 Copying files via a CIFS-mounted filesystem

Files can be copied to/from a CIFS/SMB file share by mounting the share from dom0.

The following example assumes that the fileshare is part of an Active Directory domain called `domain.com`, and user `myuser` has permissions to access the share. To mount a network share `\\myserver.domain.com\myshare` at `/mnt/myshare` on dom0,

```
[root@xenserver ~]# mkdir /mnt/myshare
[root@xenserver ~]# mount -t cifs -o username=myuser,workgroup=domain.com
//myserver.domain.com/myshare /mnt/myshare
Password:
[root@xenserver ~]#
```

When prompted for a password, enter the password for `myuser` in the `domain.com` domain. After completion, files can be copied to/from the fileshare by copying to/from /mnt/myshare:

```
[root@xenserver ~]# cp /mnt/myshare/NVIDIA-vgx-xenserver-6.2-331.59.i386.rpm .
[root@xenserver ~]#
```

A.3 DETERMINING A VM'S UUID

To determine a virtual machine's UUID, use the `xe vm-list` command in a dom0 shell, or XenCenter:

A.3.1 Using xe vm-list

To list all VMs and their associated UUIDs, use `xe vm-list`:

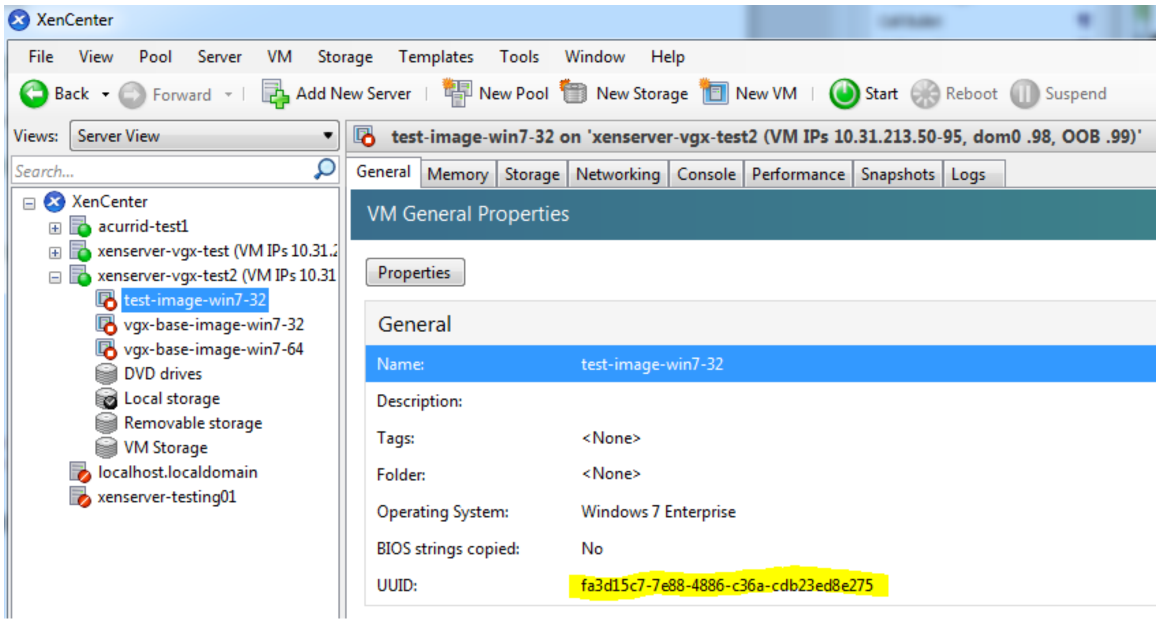
```
[root@xenserver ~]# xe vm-list
uuid ( RO)           : 6b5585f6-bd74-2e3e-0e11-03b9281c3ade
     name-label ( RW): vgx-base-image-win7-64
    power-state ( RO): halted


uuid ( RO)           : fa3d15c7-7e88-4886-c36a-cdb23ed8e275
     name-label ( RW): test-image-win7-32
    power-state ( RO): halted


uuid ( RO)           : 501bb598-a9b3-4afc-9143-ff85635d5dc3
     name-label ( RW): Control domain on host: xenserver
    power-state ( RO): running


uuid ( RO)           : 8495adf7-be9d-eee1-327f-02e4f40714fc
     name-label ( RW): vgx-base-image-win7-32
    power-state ( RO): halted
```

To find the UUID of a specific named VM, use the `name-label` parameter to `xe vm-list`:

```
[root@xenserver ~]# xe vm-list name-label=test-image-win7-32
uuid ( RO)           : fa3d15c7-7e88-4886-c36a-cdb23ed8e275
     name-label ( RW): test-image-win7-32
    power-state ( RO): halted
```

A.3.2 Using XenCenter

In the left-hand pane click on the VM, then click on the General tab . The UUID is listed in the VM's General Properties.


XenCenter
File View Pool Server VM Storage Templates Tools Window Help
Back Forward Add New Server New Pool New Storage New VM Start Reboot Suspend
Views: Server View
Search...
XenCenter
acurrid-test1
xenserver-vgx-test (VM IPs 10.31.2
xenserver-vgx-test2 (VM IPs 10.31
test-image-win7-32
vgx-base-image-win7-32
vgx-base-image-win7-64
DVD drives
Local storage
Removable storage
VM Storage
localhost.localdomain
xenserver-testing01
General Memory Storage Networking Console Performance Snapshots Logs
VM General Properties
Properties

test-image-win7-32 on 'xenserver-vgx-test2 (VM IPs 10.31.213.50-95, dom0 .98, OOB .99)'

General

Name:	test-image-win7-32
Description:	
Tags:	<None>
Folder:	<None>
Operating System:	Windows 7 Enterprise
BIOS strings copied:	No
UUID:	fa3d15c7-7e88-4886-c36a-cdb23ed8e275

Figure 27 Using XenCenter to determine a VM's UUID

A.4 USING MORE THAN TWO VCPUS WITH WINDOWS CLIENT VMS

Window client operating systems support a maximum of two CPU sockets. When allocating vCPUs to virtual sockets within a guest VM, XenServer defaults to allocating one vCPU per socket; any more than two vCPUs allocated to the VM won't be recognized by the Windows client OS.

To fix this, set `platform:cores-per-socket` to the number of vCPUs allocated to the VM:

```
[root@xenserver ~]# xe vm-param-set uuid=<vm-uuid> platform:cores-per-socket=4
VCPUs-max=4 VCPUs-at-startup=4
```

A.5 PINNING VMS TO A SPECIFIC CPU SOCKET AND CORES

Use `xe host-cpu-info` to determine the number of CPU sockets and logical CPU cores in the server platform. In this example the server implements 32 logical CPU cores across two sockets:

```
[root@xenserver ~]# xe host-cpu-info
cpu_count                 : 32
```

```
              socket_count: 2
                    vendor: GenuineIntel
                     speed: 2600.064
                 modelname: Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz
                    family: 6
                     model: 45
                  stepping: 7
                     flags: fpu de tsc msr pae mce cx8 apic sep mtrr mca cmov
pat clflush acpi mmx fxsr sse sse2 ss ht nx constant_tsc nonstop_tsc aperfmperf
pni pclmulqdq vmx est ssse3 sse4_1 sse4_2 x2apic popcnt aes hypervisor ida arat
tpr_shadow vnmi flexpriority ept vpid
                  features: 17bee3ff-bfebfbff-00000001-2c100800
      features_after_reboot: 17bee3ff-bfebfbff-00000001-2c100800
         physical_features: 17bee3ff-bfebfbff-00000001-2c100800
                  maskable: full
```

To pin a VM's vCPUs to a specific socket, set `VCPUs-params:mask`. This setting persists over VM reboots and shutdowns. In a dual socket platform with 32 total cores, cores 0-15 are on socket 0, and cores 16-31 are on socket 1. To restrict a VM to only run on socket 0:

```
[root@xenserver ~]# xe vm-param-set uuid=<vm-uuid> VCPUs-
params:mask=0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
```

Similarly, to restrict a VM to only run on socket 1:

```
[root@xenserver ~]# xe vm-param-set uuid=<vm-uuid> VCPUs-
params:mask=16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31
```

To pin vCPUs to specific cores within a socket, specify the cores directly:

```
[root@xenserver ~]# xe vm-param-set uuid=<vm-uuid> VCPUs-
params:mask=16,17,18,19
```

Use `xl vcpu-list` to list the current assignment of vCPUs to physical CPUs:

```
[root@xenserver ~]# xl vcpu-list
Name                                ID  VCPU   CPU State   Time(s) CPU Affinity
Domain-0                             0     0    25   -b-    9188.4  any cpu
Domain-0                             0     1    19   r--    8908.4  any cpu
Domain-0                             0     2    30   -b-    6815.1  any cpu
Domain-0                             0     3    17   -b-    4881.4  any cpu
Domain-0                             0     4    22   -b-    4956.9  any cpu
Domain-0                             0     5    20   -b-    4319.2  any cpu
Domain-0                             0     6    28   -b-    5720.0  any cpu
```

```
Domain-0                            0     7    26    -b-     5736.0   any cpu
test-image-win7-32                 34     0     9    -b-       17.0   4-15
test-image-win7-32                 34     1     4    -b-       13.7   4-15
```

A.6 CHANGING DOM0 VCPUS AND PINNING

The default number of vCPUs assigned to dom0 is 8. To change this number, modify the dom0_max_vcpus parameter in the Xen bootline. For example:

```
[root@xenserver ~]# /opt/xensource/libexec/xen-cmdline --set-xen
dom0_max_vcpus=4
```

Note: After applying this setting, you must reboot the system for it to take effect. Use `shutdown -r now` to reboot the server, or reboot it from XenCenter.

By default, dom0's vCPUs are unpinned, and able to run on any physical CPU in the system. To pin dom0 vCPUs to specific physical CPUs, use `xl vcpu-pin`. For example, to pin dom0's vCPU 0 to physical CPU 18, use:

```
[root@xenserver ~]# xl vcpu-pin Domain-0 0 18
```

CPU pinnings applied this way take effect immediately but do not persist over reboots. To make settings persistent, add `xl vcpu-pin` commands into `/etc/rc.local`, for example:

```
xl vcpu-pin 0 0 0-15
xl vcpu-pin 0 1 0-15
xl vcpu-pin 0 2 0-15
xl vcpu-pin 0 3 0-15
xl vcpu-pin 0 4 16-31
xl vcpu-pin 0 5 16-31
xl vcpu-pin 0 6 16-31
xl vcpu-pin 0 7 16-31
```

A.7 DETERMINING GPU LOCALITY

As noted in section 6.4.1, current multi-socket, servers typically implement PCIe expansion slots local to each CPU socket and it is advantageous to pin VMs to the same socket that their associated physical GPU is connected to.

For current Intel platforms, CPU socket 0 typically has its PCIe root ports located on bus 0, so any GPU below a root port located on bus 0 is connected to socket 0. CPU socket 1 has its root ports on a higher bus number, typically bus 0x20 or bus 0x80 depending on the specific server platform.