GRID SERVER CONFIGURATION

DU-06817-001_v01 | October 2018

Getting Started Guide

Chapter 1. GRID SERVER USE CASES

There are two basic use cases for NVIDIA GRID servers.

- Bare metal:

  This consists of a single OS which can be either Linux or Windows. The OS owns and manages all the GPUs and one or more applications simultaneously.

- Xen/NMOS:

  There is one OS that typically manages all Virtual Machines running in that server configuration. Each VM runs an OS which manages one single GPU at a time.

Both use case scenarios are dramatically different from each other and serve different purposes. A third use case involves a mix of the two and takes methods from the bare metal and Xen/NMOS use cases.

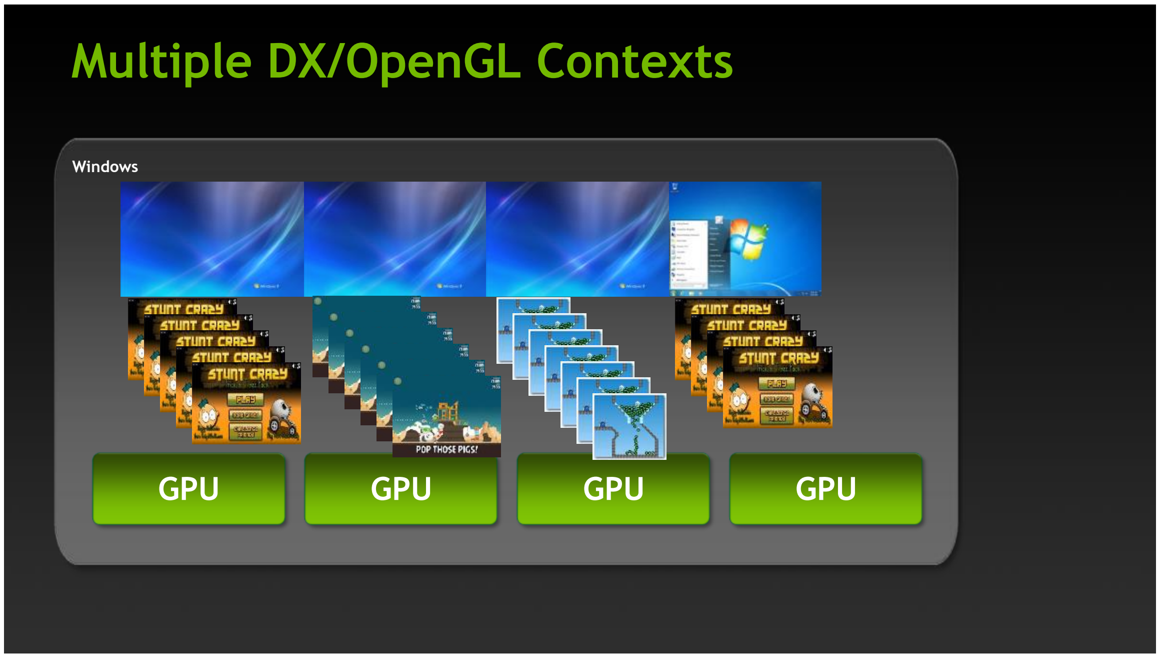
In all cases, the NVIDIA Capture SDK is used to fully accelerate pixel capture and compression for streaming while requiring very low overhead to accomplish this.

This document focuses on the bare metal use case.

1.1 BARE METAL

The bare-metal setup is best suited for running several applications per GPU simultaneously. The advantage of this approach is that it adds very little overhead to the game, as the OS is required to manage all the CPU and GPU resources. To use this approach, a software framework needs to be developed to be able to share the GPU resource among all applications. Proper sandboxing is required such that each game instance will not interfere with other game instances.

n apps, m GPUs, n streams, 1 OS instance



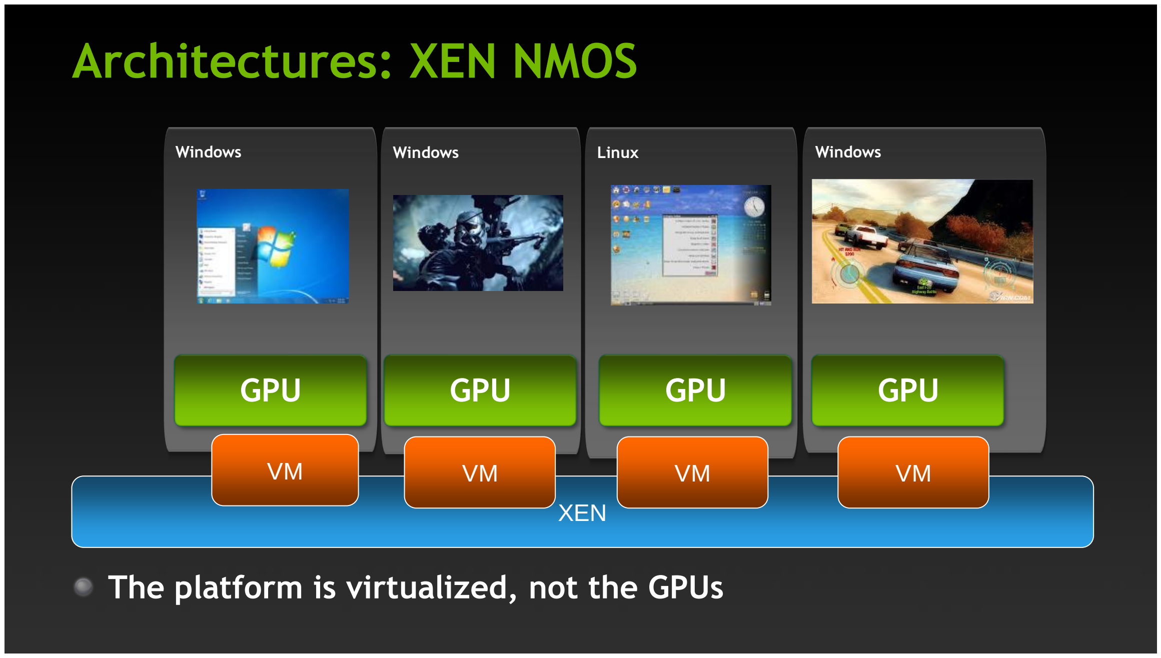
With only one OS, several "full-screen" applications cannot run concurrently because of inherent restrictions placed on applications running under Windows OS. Fortunately, it is possible to have multiple "non full-screen exclusive" applications run concurrently. These applications are run within a sandbox and shimmed at the DirectX layer, so that instead of rendering to full screen, it is rendered in a window. By adhering to this model, more than one application can run per GPU, and each application is able to take full advantage of GPU acceleration for all GPUs.

More than one graphics application can run on a single GPU, provided enough resources (memory) and visual computing power is available for each GPU. The SHIM redirects the graphics rendering calls from the application to the SHIM. For device creation, the SHIM chooses a specific GPU. For Direct3D handling, the full-screen exclusive control mode desktop application is created as a windowed non-exclusive mode Direct3D mode. This allows several windows and games to run on a single GPU, or multiple GPUs, and also enables capturing the rendered frames and compressing them using functions from the NVIDIA Video SDK.

1.2 XEN/NMOS

The Xen/NMOS setup is better suited for running one full screen application for each OS and GPU.

m Full-screen apps, m GPUs, m streams, m OS instances



Because exactly one OS manages only one GPU at a time, a single application can take over full-screen and render freely. For this scenario, a shim layer is not needed. The asynchronous "full-screen" grabbing feature of the **NvFBC** functions can be used for this case.

Chapter 2. SETTING UP GRID SERVER ON BARE METAL

2.1 SETTING UP GRID SERVER ON WINDOWS 8 / WINDOWS 2012

1. Install GRID PCI-e boards for a server system with 4 PCI-e slots.

   Up to 4 GRID PCI-e boards can be installed.

2. Boot up the machine into the system BIOS and configure IPMI (Intelligent Platform Management Interface).

   IPMI is a standard computer system level interface used by system administrators for remote management of data centers.

   a) Configure IPMI settings with a static IP address (xxx.yyy.zzz.aaa).

   b) Verify Connectivity by using http://www.yyy.zzz.aaa

      User: ADMIN / Password: ADMIN

3. Install Windows 8 or Windows Server 2012 on the GRID server.

   Network drivers are automatically included and installed automatically.

4. Assign a static IP address for the GRID server (remember this IP address).

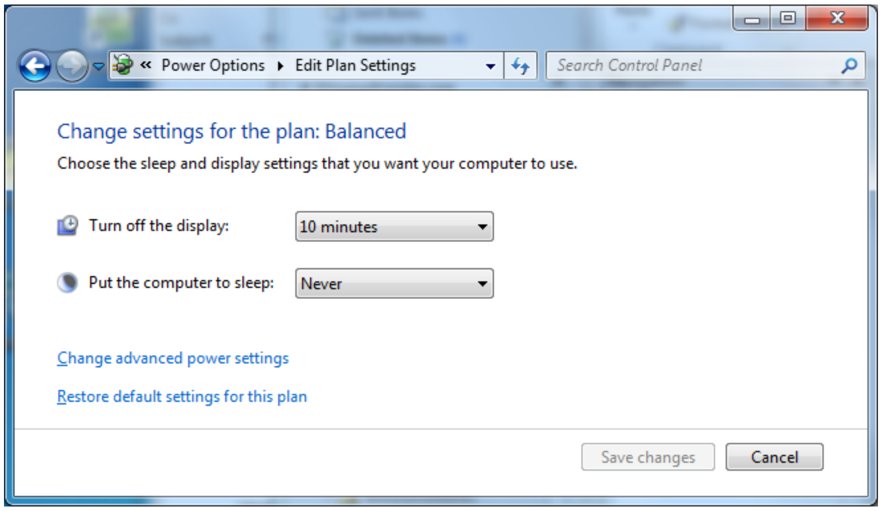
5. Reboot and verify that networking is working.

6. Install VNC for remote desktop, then set up the VNC server password.

   a) Click the checkbox to start VNC service on system boot.

   b) Verify VNC is working.

7. Go into the Control Panel and Power Options, change the setting for "Put the computer to sleep:" to **Never**.



Change settings for the plan: Balanced

Choose the sleep and display settings that you want your computer to use.

Turn off the display: 10 minutes

Put the computer to sleep: Never

Change advanced power settings
Restore default settings for this plan
Save changes
Cancel

8. Install the NVIDIA GRID Drivers (now 12 GPU for 3 GRID K340) devices and verify success through the Windows Device Manager.

   The onboard VGA is also active.

9. Set Primary to one of the GRID K340 boards, and disable virtual monitor on the onboard VGA.

10. Boot up the system.

    You can now run DirectX or OpenGL accelerated games.

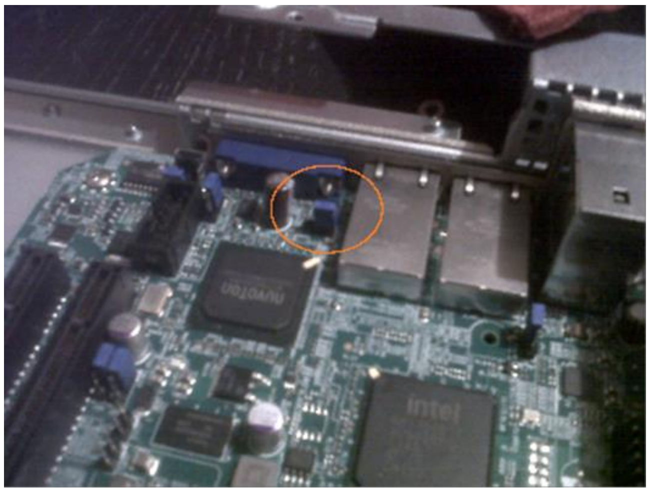
Note: For DirectX9 Games, up to 12 GPUs can be used, but for Direct3D10 and Direct3D11 does not have this limitation. Now jump to section 2.3 to continue.

2.2 SETTING UP GRID SERVER ON WINDOWS 7 / WINDOWS 2008R2

1. Make sure the onboard VGA controller is enabled.

   This is accomplished by installing jumper JPG1 in the enabled position, which is typically the default position.

   Refer to the server motherboard manual for the location of jumper JPG1.

2. Install the GRID PCI-e boards into the server system.

   For a server system with four PCI-e slots, first install only three GRID boards.

3. Install Windows 7 (64-bit) or Windows Server 2008 R2.

   The network driver either be found from this FTP location:

   ftp://ftp.supermicro.com/CDR-X9-UP_1.22_for_Intel_X9_UP_platform/Intel/LAN/

4. Assign a static IP address for the GRID server (remember this IP address).

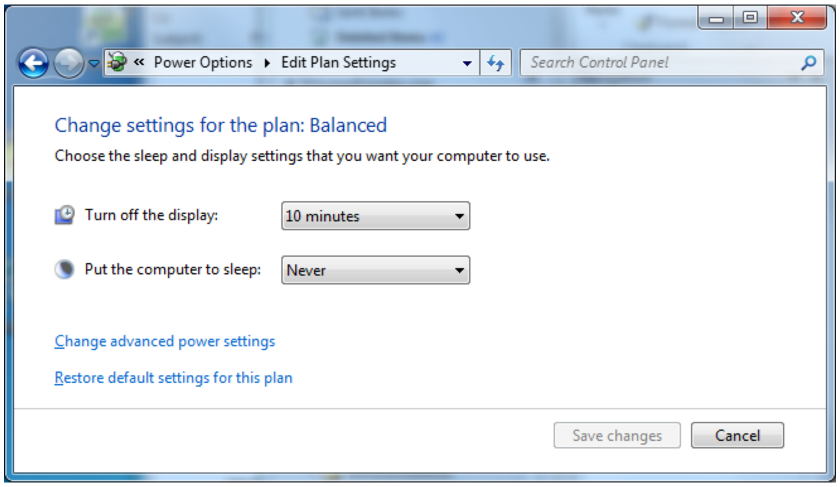
5. Reboot and verify that networking is working.

6. Install VNC for remote desktop, and set up the VNC server password.

   a) Click the checkbox to start VNC service on system boot.

   b) Verify VNC is working.

7. Go into the Control Panel and Power Options, change the setting for "Put the computer to sleep:" to **Never**.



Change settings for the plan: Balanced

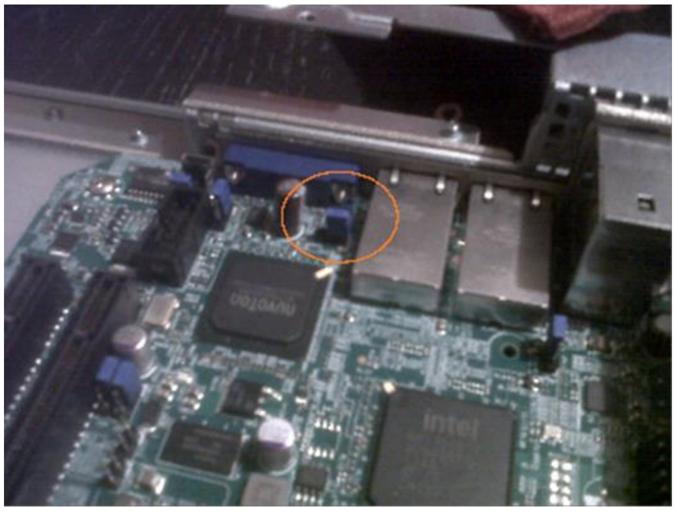
Choose the sleep and display settings that you want your computer to use.

Turn off the display: 10 minutes

Put the computer to sleep: Never

Change advanced power settings
Restore default settings for this plan
Save changes
Cancel

8. Shut down the system.
9. Move jumper JPG1 in the back of the system to the Disabled position. This will disable the onboard VGA, so no display can be driven through the VGA port.



10. Power on the system.
11. From another PC, connect to the server via VNC.
12. Install the NVIDIA GRID Drivers.

    Now you should see 12 GPU devices installed in the Windows Device Manager.
13. Boot up the system.

    It will be running in headless mode, and you can now run DirectX or OpenGL accelerated games.

Note: Do not use Microsoft Remote Desktop Protocol (RDP) to connect to the server. Running a RDP session will cause the NVIDIA drivers to be unloaded and use non-accelerated GDI for rendering, effectively disabling the NVIDIA driver. Go to Section 2.3 to continue installation.

2.3 CONFIGURING BOARDS FOR WINDOWS

Connection via VNC will show a desktop with all resolutions as well as the NVIDIA Control Panel. In order to access all of the extra GPUs, it is necessary to extend the desktop to enable assigning GPUs to an extended desktop. This configuration allows GPU accelerated applications to run for each desktop and enables the application to take full advantage of the multiple GPUs by using the GPUs as DirectX devices.

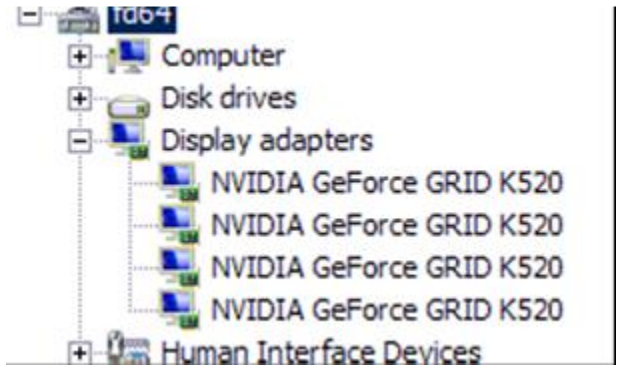


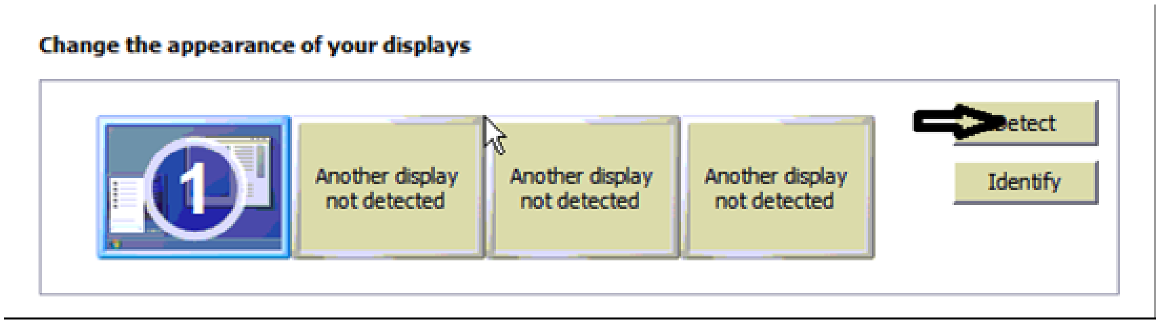
Figure 1. Example System with Two GRID K520 Boards Showing Four GPUs

For information on how to enable this automatically, refer to the NVIDIA Capture SDK example at

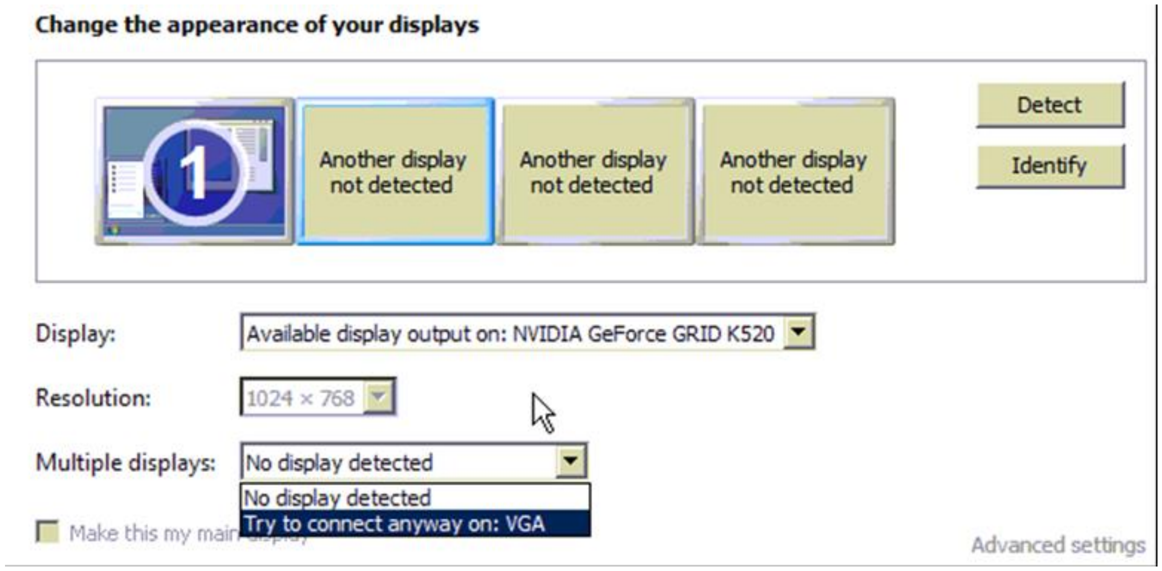
> "\NVIDIA Corporation\Capture SDK\docs\Displayless Multi-GPU on Windows 7.pdf".

To do enable this manually, perform the following steps.

1. Right click the desktop, then select Personalize-> Display->Adjust resolution.


Change the appearance of your displays
1
Another display not detected
Another display not detected
Another display not detected
Detect
Identify

2. Select each of these "displays", then select "Try to connect anyway on: VGA", and then click "Apply". Repeat for each GPU.



Change the appearance of your displays

1
Another display not detected
Another display not detected
Another display not detected
Detect
Identify

Display: Available display output on: NVIDIA GeForce GRID K520

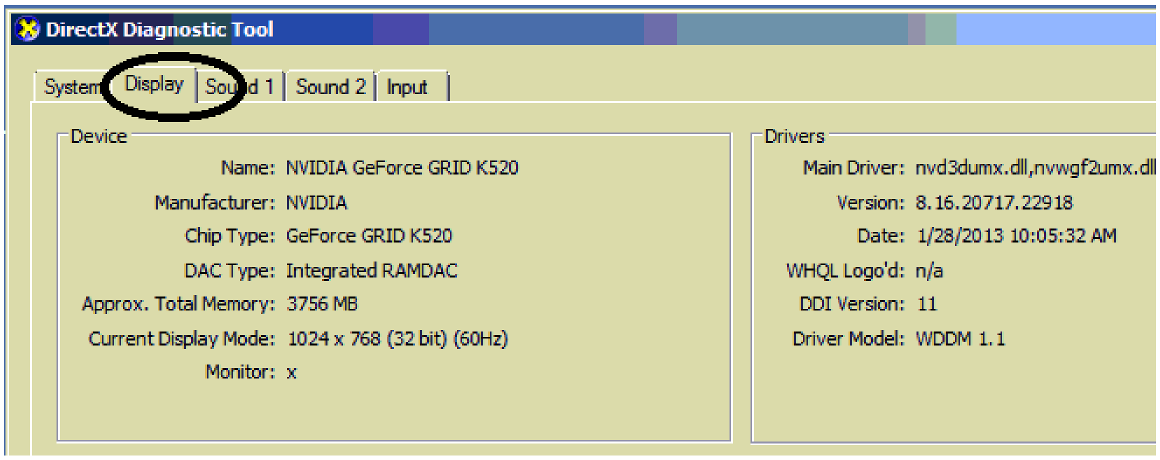
Resolution: 1024 × 768

Multiple displays: No display detected

No display detected
Try to connect anyway on: VGA

Make this my main display

At this point, all GPUs have been "connected" to GDI, and DxDiag.exe (from the Microsoft DirectX SDK) will report as shown here:



DirectX Diagnostic Tool

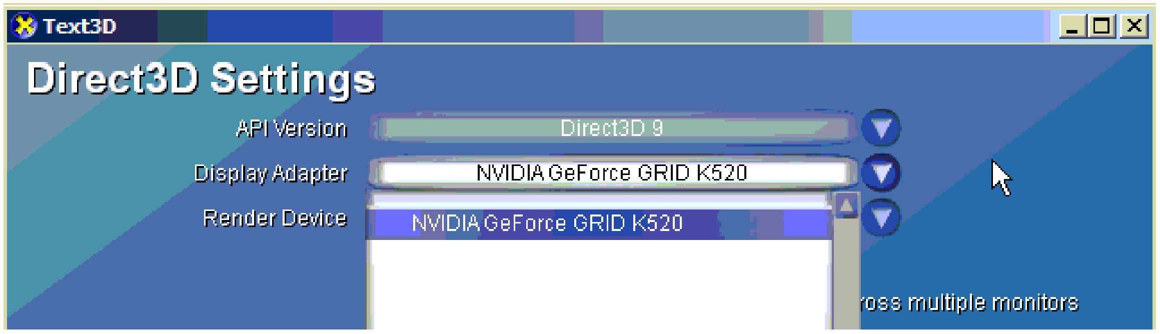
Device

Name:	NVIDIA GeForce GRID K520
Manufacturer:	NVIDIA
Chip Type:	GeForce GRID K520
DAC Type:	Integrated RAMDAC
Approx. Total Memory:	3756 MB
Current Display Mode:	1024 x 768 (32 bit) (60Hz)
Monitor:	x

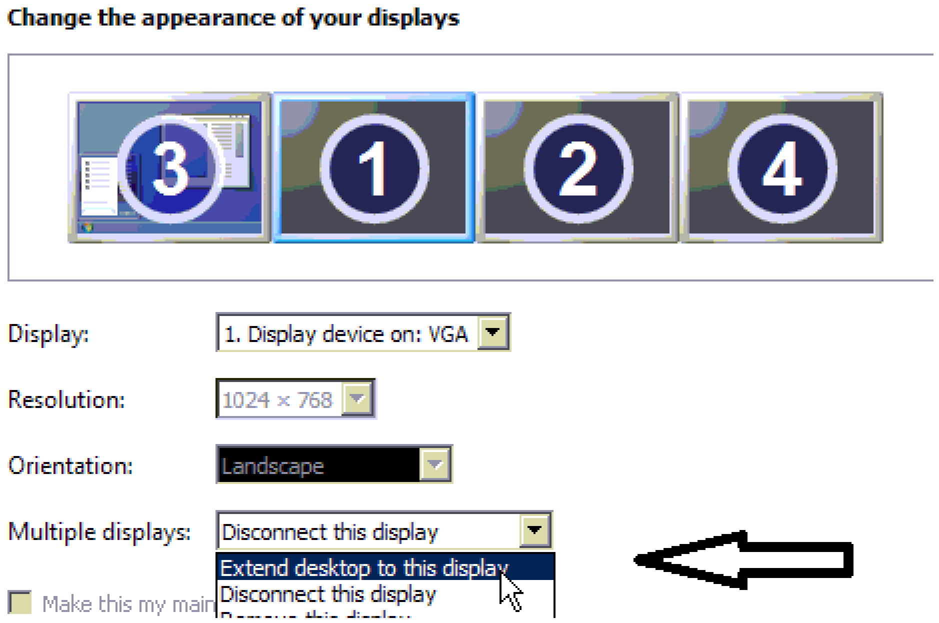
Drivers

Main Driver:	nvd3dumx.dll,nvwgf2umx.dll
Version:	8.16.20717.22918
Date:	1/28/2013 10:05:32 AM
WHQL Logo'd:	n/a
DDI Version:	11
Driver Model:	WDDM 1.1

The most basic DirectX SDK sample will report only one usable GPU:



By extending the desktop to the "connected" GPU, we end up with fully addressable DirectX renderers.


Change the appearance of your displays
3
1
2
4
Extend desktop to this display
Disconnect this display
Make this my main

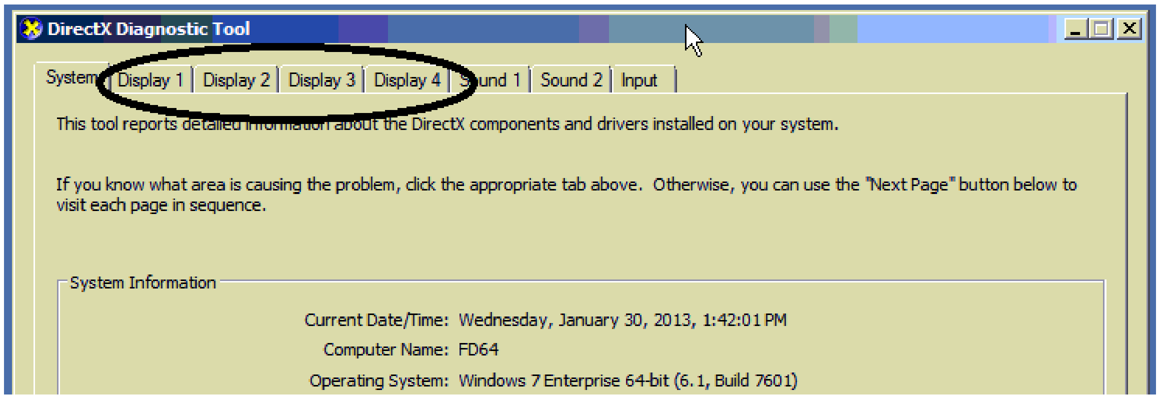
Display: 1. Display device on: VGA

Resolution: 1024 × 768

Orientation: Landscape

Multiple displays: Disconnect this display

Here is a screenshot of DxDiag.exe



This tool reports detailed information about the DirectX components and drivers installed on your system.

If you know what area is causing the problem, click the appropriate tab above. Otherwise, you can use the "Next Page" button below to visit each page in sequence.

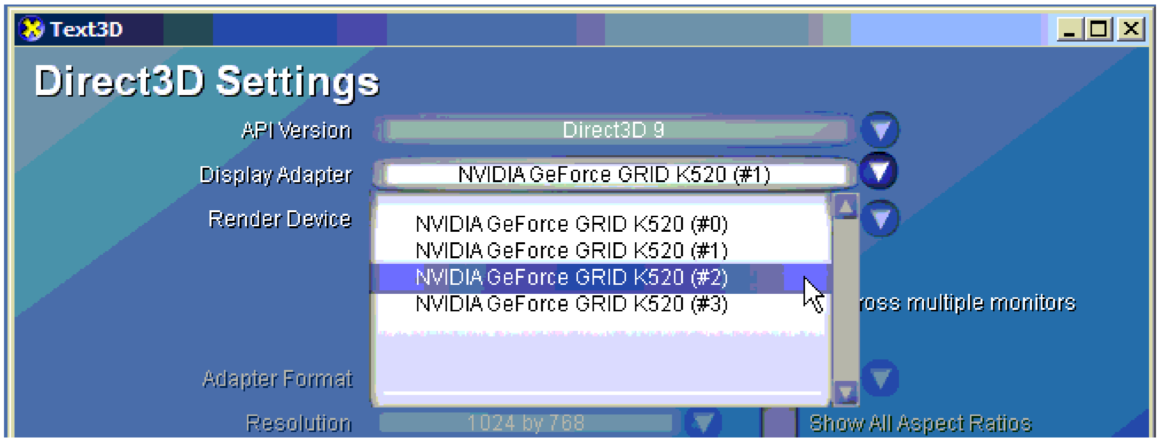
System Information

Current Date/Time: Wednesday, January 30, 2013, 1:42:01 PM

Computer Name: FD64

Operating System: Windows 7 Enterprise 64-bit (6.1, Build 7601)

And from the Microsoft Text3D.exe SDK sample.



Each DirectX renderer can now be accessed by creating a device with the right "DX Ordinal".

m_pIDirect3D9->CreateDevice(**dwDXOrdinal**, DeviceType, hFocusWindow...

Once this device has been created, it is best to hide the window that uses the DX device to ensure the GDI won't trigger any cross adapter blits as this will hurt overall performance.

Reminder: The location of a window does not determine which GPU renders it in DirectX; it is set one time at CreateDevice() time.

2.4 OTHER WINDOWS OS BARE-METAL USE CASES

This section describes a method to address DirectX adapters where one application has access and visibility to all GPUs available in the system.

1 application or game, m GPUs, n streams, 1 instance of OS

This configuration type enables new game server and engine architectures. Application developers would directly manage multiple-GPUs. This may include added performance available for photo realistic rendering, which can include several GPUs working in parallel in the same rendering. Or it could be a single shared-world with multi-viewports for each user in the game world.

2.5 LINUX BARE METAL

Linux can be installed. The main difference compared to section 2.1 is that this installation does not allow applications to access each GPU independently. The advantage with this approach is that any OpenGL program can utilize all GPUs in the system at the same time, as has been described for the use case in section 2.2.