Enabling Vulkan* **VK_INTEL_performance_query** extension in Ubuntu*

These instructions describe how to compile and use custom Linux* kernel and Mesa* Vulkan drivers that include Intel's Vulkan **VK_INTEL_performance_query** extension which was first published in the Khronos VulkanDocs repository as part of the 1.1.109 specification. As of January 29th, 2020, their implementation/patches are now in kernel 5.5, and the Mesa branches in 20.0 and 19.3. The **VK_INTEL_performance_query** extension is designed specifically for 8th generation or newer Intel GPUs including Broadwell, Skylake, Kaby Lake and Coffee Lake.

Note
These instructions were developed on Ubuntu 18.04. They have not been tested on other Linux distributions at this point.

Prerequisites:

Install and configure Git if not already set up. Minimally configure a username and email address:

```
sudo apt-get update

sudo apt-get install git

git config –global user.name "<User Name>"

git config --global user.email "<User Name's email address>"
```

If not already installed, install key Vulkan libraries, drivers and utilities:

```
sudo apt-get update
sudo apt-get install libvulkan1 mesa-vulkan-drivers vulkan-utils
```

Run and test a known working Vulkan application on the platform to confirm that the existing Vulkan drivers function properly. If the application does not work as expected, investigate and resolve the problem before continuing with the next steps in these instructions.

Install packages required to compile the Linux kernel:

```
sudo apt-get update

sudo apt-get install build-essential kernel-package fakeroot libssl-dev
ccache flex bison libelf-dev libncurses5-dev wget
```

Warning
Some packages may prompt to make changes to the local platform and an appropriate choice should be selected based off the platform being updated.

Install packages needed by Mesa:

```
sudo apt-get install cmake python3 python-minimal python-mako
python3-mako pkg-config libexpat1-dev libxrandr-dev libdrm-dev
libxdamage-dev libxcb-glx0-dev libxcb-dri2-0-dev libxcb-dri3-dev
libxcb-present-dev libxshmfence-dev libxxf86vm-dev libx11-xcb-dev
ninja-build
```

If using Ubuntu 16.03 install additional packages needed by Mesa:

```
sudo apt-get install x11proto-gl-dev x11proto-dri2-dev gettext
```

Install or update meson to meet Mesa's dependencies (for example, a minimum of version 0.46):

```
sudo apt-get remove meson
sudo apt-get install python3-pip
pip3 install --user meson
```

If the installed version of meson is not in the PATH..

```
export PATH=~/.local/bin/:$PATH
```

Setup Steps:

1. Create the **~/custom** work area where the kernel and Mesa drivers will be built and reside:

   ```
   mkdir -p ~/custom
   ```

2. Obtain source for Linux kernel 5.5 (tag v5.5-rc2, a later release candidate or final 5.5) which contains the i915 driver changes required by the extension, build, and install it:

```
cd ~/custom

git clone
https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git

cd linux

git checkout -b 5.5-rc2 d1eef1c6

cp /boot/config-`uname -r` .config

sudo -E chown -R $USER:$USER $HOME/.ccache/ # If $HOME/.ccache
exists, avoid potential build failure.

make menuconfig # Exit immediately and save default tool changes

make clean

make -j4 # Enable a faster build by specifying the number of
cores/threads on the platform.

sudo make INSTALL_MOD_STRIP=1 modules_install install
```

3. Reboot the platform. By default, grub will automatically boot using the newly built and installed kernel. After rebooting, confirm that the newly built kernel is running with version 5.5.0-rc2 and an appropriate time stamp as follows:

```
uname -a
```

4. If LLVM is not installed or at least version 7.0.1, download LLVM 7.0.1 source (https://releases.llvm.org/download.html#7.0.1), compile and install it. You can check your version with "llvm-config –version". Compiling from the source is necessary because the pre-compiled packages do not include the needed LLVM dynamic libraries.

```
cd ~/custom

wget http://releases.llvm.org/7.0.1/llvm-7.0.1.src.tar.xz

tar xvf llvm-7.0.1.src.tar.xz

cd llvm-7.0.1.src/

mkdir build

cd build/

cmake .. -DLLVM_BUILD_LLVM_DYLIB=on -DLLVM_LINK_LLVM_DYLIB=on
```

```
make -j4 # Enable a faster build by specifying the number of
cores/threads on the platform.
sudo make install
```

5. Obtain source for the Mesa Vulkan drivers with the extension and then build them:

   ```
   cd ~/custom

   git clone https://gitlab.freedesktop.org/mesa/mesa.git

   cd mesa

   git checkout mesa-19.3.3

   git log -1 # The commit should be 35203731.

   # Edit meson.build to change dependency _drm_amdgpu_ver patch
   revision to 0, thus "2.4.0"

   mkdir build

   meson build --prefix=`pwd`/install -Dplatforms=x11,drm,surfaceless
   -Ddri-drivers=i915,i965,swrast -Dgallium-drivers= -Dvulkan-
   drivers=intel

   ninja -C build install
   ```

6. Confirm that the Mesa libraries and Vulkan drivers were successfully built by checking for Mesa .so files in the following directory:

   ```
   ~/custom/mesa/install/lib/x86_64-linux-gnu
   ```

   Check if the following two files exist:

   ```
   ~/custom/mesa/install/lib/x86_64-linux-gnu/libvulkan_intel.so

   ~/custom/mesa/install/share/vulkan/icd.d/intel_icd.x86_64.json
   ```

7. On boot, the i915 kernal module enables the paranoid performance collection mode by default. To use the **VK_INTEL_performance_query** extension, this paranoid mode must be disabled.

   1. A manual approach to do this is to perform the following command:

      ```
      sudo sysctl -w dev.i915.perf_stream_paranoid=0
      ```

2. An automated approach to do this is to add a cron job that executes whenever the platform reboots, as follows:

   ```
   sudo crontab -e # Add the following line at the end or as the
   1st no comment line:

   @reboot /sbin/sysctl -w dev.i915.perf_stream_paranoid=0
   ```

3. After manually performing 7.1 or doing 7.2 and a later reboot, the “0” perf_stream_paranoid mode change can be confirmed by using the following command:

   ```
   sysctl -n dev.i915.perf_stream_paranoid
   ```

8. Lock GPU Frequency to get accurate metric values in Graphics Frame Analyzer, by:

   1. Modifying the rights to the following files to add write capabilities:

      ```
      sudo chmod +w /sys/class/drm/card0/gt_max_freq_mhz
      sudo chmod +w /sys/class/drm/card0/gt_min_freq_mhz
      sudo chmod +w /sys/class/drm/card0/gt_boost_freq_mhz
      ```

   2. Write the same GPU frequency value in mhz for the three files.

9. When running Intel® GPA Frame Analyzer, the following is necessary to obtain Vulkan metrics from the GPU. Override the installed Mesa libraries and Vulkan drivers to use the new (custom) locally built ones by defining/updating two environment variables for Intel® GPA:

   1. If starting Frame Analyzer using the Ubuntu desktop launcher icon/links, edit the associated launcher file:

      ```
      /usr/share/applications/frame-analyzer.desktop
      ```

      by replacing the "Exec=frame-analyzer" line with the following two lines (replace "<user-dir>" with the user home directory name):

      ```
      Exec=env LD_LIBRARY_PATH=/home/<user-
      dir>/custom/mesa/install/lib/x86_64-linux-gnu
      VK_ICD_FILENAMES=/home/<user-
      dir>/custom/mesa/install/share/vulkan/icd.d/intel_icd.x86_64.json
      /opt/intel/gpa/FrameAnalyzer.sh

      Path=/opt/intel/gpa
      ```

2. Otherwise, Frame Analyzer can be manually started from the command-line as follows:

   ```
   cd /opt/intel/gpa

   export LD_LIBRARY_PATH=~/custom/mesa/install/lib/x86_64-
   linuxgnu:$LD_LIBRARY_PATH

   export
   VK_ICD_FILENAMES=~/custom/mesa/install/share/vulkan/icd.d/intel_icd.
   x86_64.json

   ./FrameAnalyzer.sh
   ```