HDDL-S installation guide

Hardware: CoffeeLake, Intel(R) Xeon(R) E-2176G CPU @ 3.70GHz

OS: Ubuntu 16.04.3

1. **Install OpenVINO R5**

Download OpenVINO R5:

<https://software.intel.com/en-us/openvino-toolkit/choose-download/free-download-linux>

Install OpenVINO:

<https://software.intel.com/en-us/articles/OpenVINO-Install-Linux>

*#source /opt/intel/computer\_vision\_sdk/bin/setupvars.sh*

For Intel® Vision Accelerator Design with Intel® Movidius™ VPUs, the following additional installation steps are required.

Install dependencies

*#sudo apt install libusb-1.0-0 libboost-program-options1.58.0 libboost-thread1.58.0 libboost-filesystem1.58.0 libssl1.0.0 libudev1 libjson-c2*

Add the current Linux user to the users group:

*#sudo usermod -a -G users "$(whoami)"*

Logout and login to make it take into effect

If your Intel® Vision Accelerator Design with Intel® Movidius™ VPUs card requires SMBUS connection to PCIe slot (Raw video data card with HW version Fab-B and before), generate rules for SMBUS controller and enable the i2c\_i801 driver (Intel® SMBUS controller driver):

*#cd ${HDDL\_INSTALL\_DIR}*

*#sudo chmod +x ./generate\_udev\_rules.sh*

*#sudo ./generate\_udev\_rules.sh /etc/udev/rules.d/98-hddlbsl.rules*

Check if the file /etc/modprobe.d/blacklist.conf contains the line "blacklist i2c\_i801" and comment it if so:

*#sudo sed -i "s/\(.\*i2c\_i801$\)/#\1/g" /etc/modprobe.d/blacklist.conf*

#sudo modprobe i2c\_i801 # unblocking will take effect at next reboot. To avoid reboot, this time we still insmod manually

Before you start installing the drivers, make sure there is no inference application is running. Use the following command to make sure there is no HDDL service is running:

*#kill -9 $(pidof hddldaemon autoboot)*

Install the drivers:

*#cd ${HDDL\_INSTALL\_DIR}/drivers*

*#sudo chmod +x ./setup.sh*

*#sudo ./setup.sh install*

*#sudo cp -av ${HDDL\_INSTALL\_DIR}/../97-myriad-usbboot.rules /etc/udev/rules.d/*

*#sudo cp -av ${HDDL\_INSTALL\_DIR}/etc /*

*#sudo udevadm control --reload-rules*

*#sudo udevadm trigger*

*#sudo ldconfig*

Now, the drivers are installed.

1. **Install OpenCL**

*#cd /opt/intel/computer\_vision\_sdk/install\_dependencies*

*#sudo ./install\_NEO\_OCL\_driver.sh*

Add OpenCL users to the video group:

*#sudo usermod -a -G video USERNAME*

e.g. if the user running OpenCL host applications is foo, run: sudo usermod -a -G video foo

Install 4.14 kernel using install\_4\_14\_kernel.sh script and reboot into this kernel

*#sudo ./install\_4\_14\_kernel.sh*

If you use 8th Generation Intel processor, you will need to add:

i915.alpha\_support=1

to the 4.14 kernel command line, in order to enable OpenCL functionality for this platform.

1. **Install OpenCL SDK**

*#sudo apt-get install dkms*

*#tar -xvf intel\_sdk\_for\_opencl\_2017\_7.0.0.2568\_x64.gz*

*#cd intel\_sdk\_for\_opencl\_2017\_7.0.0.2568\_x64*

*#./install\_GUI.sh*

1. **Install**

**4.1 dependency package**

*#sudo apt-get install libelf-dev*

*#sudo apt-get install libusb-1.0-0-dev libudev-dev libssl-dev rpm cmake libboost-program-options1.58-dev libboost-thread1.58 libboost-filesystem1.58 git*

**4.2 Install json-c**

*#git clone https://github.com/json-c/json-c.git*

*#cd json-c*

*#git checkout f8c632f579c71012f9aca81543b880a579f634fc*

*#sudo apt-get install autoconf libtool*

*#sh autogen.sh*

*#./configure*

*#make*

*#sudo make install*

1. **Install gstreamer**

*#sudo apt-get install gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstreamer1.0-plugins-ugly gstreamer1.0-plugins-bad libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev*

*#export PKG\_CONFIG\_PATH=$PKG\_CONFIG\_PATH:/opt/intel/mediasdk/lib64/pkgconfig*

1. **Install uWebSocket**

*#sudo apt-get install libssl-dev*

*#git clone https://github.com/uNetworking/uWebSockets.git*

*#make && sudo make install*

1. **Install OpenCV**

*#sudo apt-get install libgtk2.0-dev pkg-config libgtkglext1-dev*

*#export CPLUS\_INCLUDE\_PATH=/opt/intel/mediasdk/include:$CPLUS\_INCLUDE\_PATH*

*#git clone https://github.com/opencv/opencv.git*

*#cd opencv && git checkout 6ffc48769ac60d53c4bd1913eac15117c9b1c9f7*

*#mkdir build && cd build*

*#cmake -DWITH\_VA\_INTEL=ON -DWITH\_CUDA=OFF ..*

*#make -j8*

*#sudo make install*

Note: OpenVINO has provided OpenCV libraries, but HDDL-S need VA support in OpenCV, so we must rebuild it.

1. **Install HDDL-S software**

**8.1 download source code**

*#git clone* [*git@gitlab-icv.inn.intel.com:hddl/s\_framework.git*](mailto:git@gitlab-icv.inn.intel.com:hddl/s_framework.git)

**8.2 build MSDK gst-plugin**

*#sudo apt-get install libdrm-dev libudev-dev libgstreamer-plugins-bad1.0-dev libx11-xcb-dev libgles2-mesa-dev libgl1-mesa-dev*

*#sudo ln -sf /opt/intel/mediasdk/lib64/libva.so.2 /usr/lib/libva.so*

*#sudo ln -sf /opt/intel/mediasdk/lib64/libva-drm.so.2 /usr/lib/libva-drm.so*

*#cd gstreamer-media-SDK && mkdir build && cd build && cmake ..*

*#make*

*#sudo cp build/lib/release/libgstmfx.so /usr/lib/x86\_64-linux-gnu/gstreamer-1.0/libgstmfx.so*

* 1. **build OpenVINO gst-plugin**

*#**sudo apt-get install libeigen3-dev libopenblas-dev liblapack-dev libdlib-dev*

*#cd gstreamer\_plugin\_openVINO*

*#make*

*#sudo make install*

Run below command to check if it has been installed successfully:

*# gst-inspect-1.0 cvdlfilter*

*# gst-inspect-1.0 resconvert*

*# gst-inspect-1.0 wssink*

* 1. **Install hddlspipe**

*#cd gstreamer\_pipeline*

*#make && sudo make install*

1. **Setup HDDL-S Server**

*#sudo apt-get install nodejs-legacy npm*

*#npm config set proxy http://child-prc.intel.com:913*

*#sudo npm install -g n*

*#sudo n stable*

*#npm install ws@6.0.0*

*#npm install child\_process arraybuffer-to-string rimraf commander tree-kill*

*#chmod a+x \*.js*

In client side:

*#echo <server\_ip> > path.txt*

*#echo <server\_host\_name> hostname.txt*

1. **Setup rtsp server as input stream source(option)**

*#wget* [*https://gstreamer.freedesktop.org/src/gst-rtsp-server/gst-rtsp-server-1.8.3.tar.xz*](https://gstreamer.freedesktop.org/src/gst-rtsp-server/gst-rtsp-server-1.8.3.tar.xz)

*#tar -xvf gst-rtsp-server-1.8.3.tar.xz*

*#cd gst-rtsp-server-1.8.3*

*#sudo apt-get install gtk-doc-tools*

*#cp s\_framework/test/rtsp\_serve/test-launch.c examples/.*

*#./autogen.sh && ./configure && make*

*#cd example*

*#./test-launch --gst-debug=3 "( filesrc location=/home/lijunjie/1600x1200.mp4 ! qtdemux ! rtph264pay name=pay0 pt=96 )"*

Note: in rtsp receiver side, we need set

1). echo 800000 > /proc/sys/net/core/rmem\_max

2). Add udp-buff-size=800000 into rtspsrc

For example:

gst-launch-1.0 -v rtspsrc location=rtsp://10.239.85.64:8554/test udp-buff-size=800000 ! rtph264depay ! h264parse…

1. **Run HDDL-S Software Stack**

**11.1 Set environment variable**

Modify ~/.bashrc and add below command lines into it:

*-----------------------------------------------------------------------------------------------------------------------------*

*source /opt/intel/computer\_vision\_sdk/bin/setupvars.sh*

*export LD\_LIBRARY\_PATH=/usr/local/lib:/opt/intel/computer\_vision\_sdk\_2018.5.445/deployment\_tools/inference\_engine/external/hddl/lib:$LD\_LIBRARY\_PATH*

*export PKG\_CONFIG\_PATH=$PKG\_CONFIG\_PATH:/opt/intel/mediasdk/lib64/pkgconfig*

*export LD\_LIBRARY\_PATH=/opt/intel/mediasdk/lib64:/usr/local/lib:/opt/intel/computer\_vision\_sdk/inference\_engine/lib/ubuntu\_16.04/intel64:/opt/intel/computer\_vision\_sdk\_2018.5.445/deployment\_tools/inference\_engine/external/omp/lib:/usr/lib/x86\_64-linux-gnu/gstreamer-1.0:$LD\_LIBRARY\_PATH*

*export HDDLS\_CVDL\_KERNEL\_PATH=/usr/lib/x86\_64-linux-gnu/libgstcvdl/kernels*

*export PATH=$PATH:/opt/intel/mediasdk/bin/*

---------------------------------------------------------------------------------------------

Run .bashrc

*#source ~/.bashrc*

**11.2 Generate certificate**

Please generate all these certificates in one pc!!!

**11.2.1 Prepare certificates**

-Generate a Certificate Authority:

*openssl req -new -x509 -days 9999 -keyout ca-key.pem -out ca-crt.pem*

-Insert a CA Password and remember it

-Specify a CA Common Name, like 'root.localhost' or 'ca.localhost'. This MUST be different from both server and client CN.

**11.2.2 Server certificate**

-Generate Server Key:

*openssl genrsa -out server-key.pem 4096*

-Generate Server certificate signing request:

*openssl req -new -key server-key.pem -out server-csr.pem*

-Specify server Common Name, run cat /etc/hosts to check valid DNS name, please don't name as'localhost'.

-For this example, do not insert the challenge password.

**11.2.3 Sign certificate using the CA**:

*openssl x509 -req -days 9999 -in server-csr.pem -CA ca-crt.pem -CAkey ca-key.pem -CAcreateserial -out server-crt.pem*

-insert CA Password

**11.2.4 Verify server certificate**:

*openssl verify -CAfile ca-crt.pem server-crt.pem*

**11.2.5 Client certificate**

-Generate Client Key:

*openssl genrsa -out client1-key.pem 4096*

-Generate Client certificate signing request:

*openssl req -new -key client1-key.pem -out client1-csr.pem*

-Specify client Common Name, like 'client.localhost'. Server should not verify this, since it should not do reverse-dns lookup.

-For this example, do not insert the challenge password.

**11.2.6 Sign certificate using the CA**:

*openssl x509 -req -days 9999 -in client1-csr.pem -CA ca-crt.pem -CAkey ca-key.pem -CAcreateserial -out client1-crt.pem*

-insert CA Password

**11.2.7 Verify client** **certificate**:

*openssl verify -CAfile ca-crt.pem client1-crt.pem*

**11.2.8 Deploy pem files**

After generated, please copy all file start with 'ca' and 'client' into 'cert\_client\_8216', and copy all file start with 'ca' and 'server' into 'cert\_server\_8216'.

**11.3 Run HDDL-S server**

*# cd s\_framework/hddls\_server\_client/hddls\_server*

*#export HDDLS\_CVDL\_MODEL\_PATH=<hddl-s\_server\_folder>/models*

*#./hddls\_server.js*

**11.3.1 How to deploy customer models**

Step 1: implement libxxxalgo.so as customer guide

Step 2: copy model IR files into <*HDDLS\_CVDL\_MODEL\_PATH*>/<model\_name>

Step 3: register this customer models

registeralgo -a <model\_name>

Step 4: edit create\_xxx.json and add <model\_name> into algopipeline property

Step 5: controller\_client send create pipeline command with create\_xxx.json

**11.4 Run HDDL-S clients**

**11.4.1 Run receiver client**

*# cd s\_framework/ hddls\_server\_client/hddls\_client*

*#./receiver\_client.js*

**11.4.2 Run controller client**

*# cd s\_framework/hddls\_server\_client/hddls\_client*

*#./controller\_client.js*

*--Please chose server by id:*

*--Please type model dictionary:*

//Choose the models directory that will be updated into hddls-server

*Note:*

1. Command format:

*-help commanders that you can use.*

*-c <create.json> create pipeslines*

*-p <property.json> <pipe\_id> set pipeslines property*

*-d <destroy.json> <pipe\_id> destroy pipeslines*

*-pipe display pipes belonging to the very client*

*-client display client ID*

*-model display model info*

*-q exit client.*

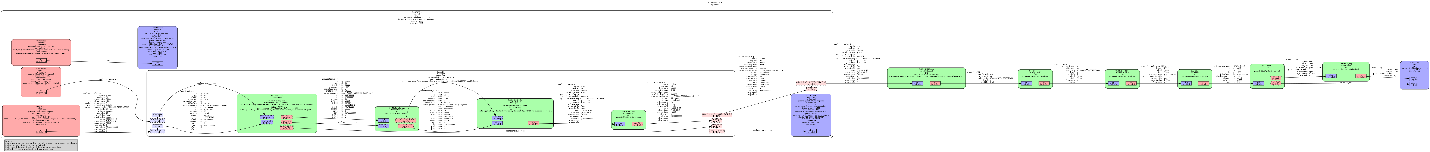
2. Json file refer to: hddls\_client/json\_file

Backup:

1. Test for cvdlfilter and resconvert plugin

*gst-launch-1.0 filesrc location=<file> ! h264parse ! mfxh264dec ! cvdlfilter ! resconvert ! mfxjpegenc ! multifilesink location=/home/lijunjie/hddls\_%d.jpeg*

1. *Gstreamer pipeline graph*

**