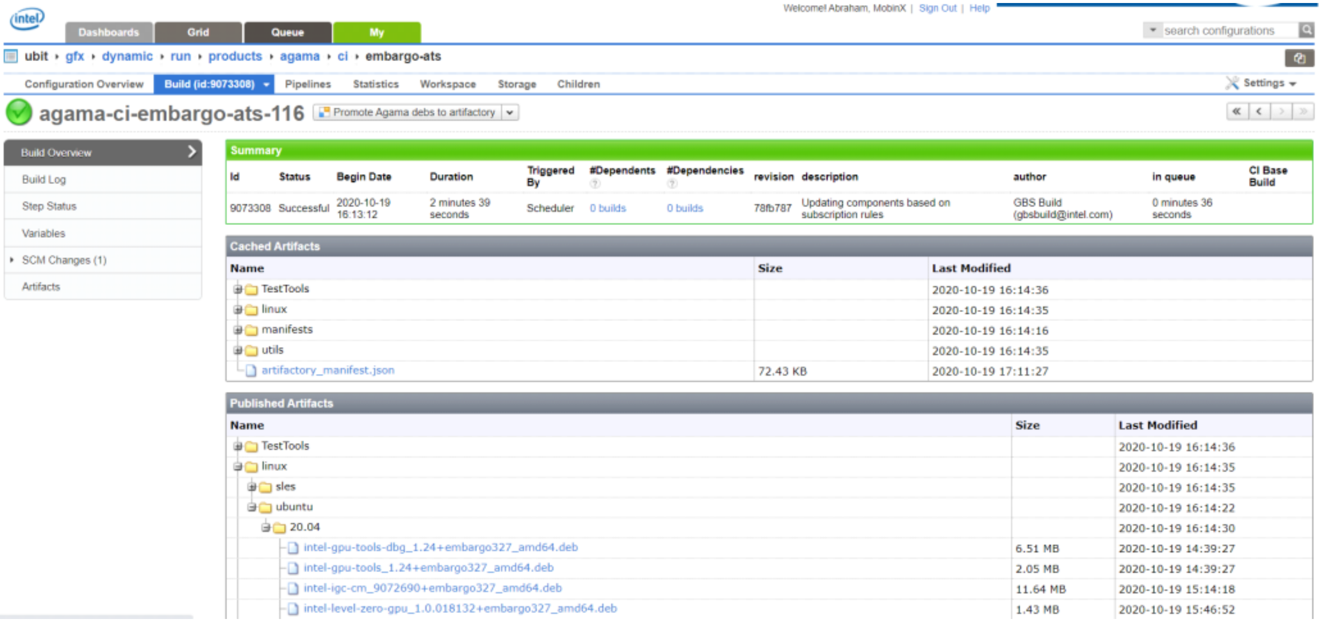
BKM

Friday, November 6, 2020 06:25 PM

BASIC Steps after Ubuntu OS Installation -

- Agama Image (Kernel Upgrade)
    - Agama Package can be downloaded from link: https://ubit-gfx.intel.com/overview/25408
    - *Published Artifacts -> linux -> ubuntu -> 20.04*



Summary

Id	Status	Begin Date	Duration	Triggered By	#Dependents	#Dependencies	revision	description	author	in queue	CI Base Build
9073308	Successful	2020-10-19 16:13:12	2 minutes 39 seconds	Scheduler	0 builds	0 builds	78fb787	Updating components based on subscription rules	GBS Build (gbsbuild@intel.com)	0 minutes 36 seconds	

Cached Artifacts

Name	Size	Last Modified
TestTools		2020-10-19 16:14:36
linux		2020-10-19 16:14:35
manifests		2020-10-19 16:14:16
utils		2020-10-19 16:14:35
artifactory_manifest.json	72.43 KB	2020-10-19 17:11:27

Published Artifacts

Name	Size	Last Modified
TestTools		2020-10-19 16:14:36
linux		2020-10-19 16:14:35
sles		2020-10-19 16:14:35
ubuntu		2020-10-19 16:14:22
20.04		2020-10-19 16:14:30
intel-gpu-tools-dbg_1.24+embargo327_amd64.deb	6.51 MB	2020-10-19 14:39:27
intel-gpu-tools_1.24+embargo327_amd64.deb	2.05 MB	2020-10-19 14:39:27
intel-igc-cm_9072690+embargo327_amd64.deb	11.64 MB	2020-10-19 15:14:18
intel-level-zero-gpu_1.0.018132+embargo327_amd64.deb	1.43 MB	2020-10-19 15:46:52

-
    - Download all the deb packages to system
    - $ ***sudo dpkg -i *.deb*** -> Will install all the packages
    - Restart the system ones installation completes
    - System is now upgraded & ready to use

    - $ ***uname -r*** -> will display current kernel version
    - $ ***dpkg -l | grep intel-opencl*** -> will display current driver version

- IP Change Issue
    - Change hostname (edit */etc/hostname* to update)
    - Remove the following files:
        i. $ ***sudo rm /etc/machine-id***
        ii. $ ***sudo rm /var/lib/dbus/machine-id***
    - Run the following two commands:
        i. $ ***sudo dbus-uuidgen --ensure=/etc/machine-id***
        ii. $ ***sudo dbus-uuidgen --ensure***
    - Run the following command to renew the IP
        i. $ ***sudo dhclient -r;sudo dhclient***
    - Reboot the system and check if there is a new IP generated

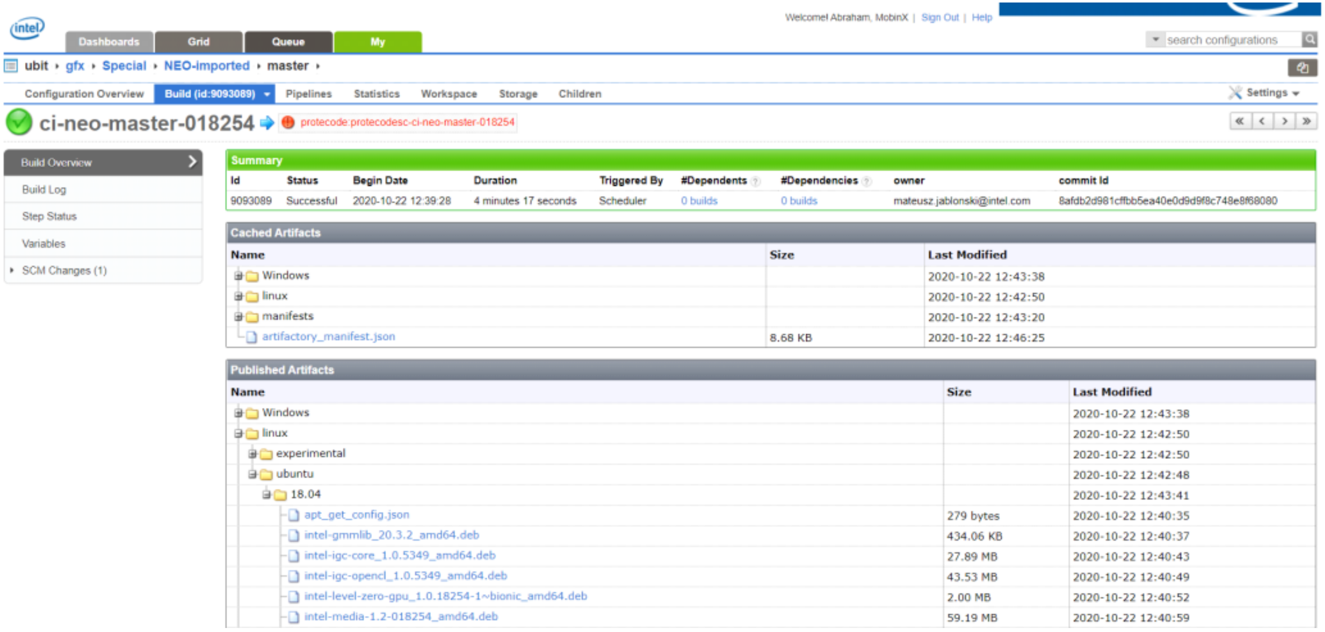
    Once we make sure the IP has changed, reboot a few times to ensure the change is consistent

- KMD/UMD Parameters
    - KMD parameters can be checked in */etc/modprobe.d/i915.conf*
    - Default parameters used: (Workaround level 3)
      options i915 force_probe=* enable_guc=3 enable_rc6=1 disable_uc_auth=1 enable_hangcheck=0 enable_guc_hangcheck=0 smem_access_control=2 enable_hw_throttle_blt=0
    - When WA level 3 is used no need of UMD Env variables. But if WA level 2 is used, needs to export Env variables.

Linux Kernel Parameters	Compute UMD Env Variables
$ cat << EOF | sudo tee /etc/modprobe.d/i915.conf options i915 smem_access_control=0 enable_hw_throttle_blt=1	export ForceLocalMemoryAccessMode=1 export ForceNonSystemMemoryPlacement= 53248
Linux Kernel Parameters	**Compute UMD Env Variables**
$ cat << EOF | sudo tee /etc/modprobe.d/i915.conf options i915 smem_access_control=2 enable_hw_throttle_blt=0	Default Config – No change needed if Level 3 WA is used

- Driver loading
  - Needs to run $ ***sudo modprobe i915***
  - This command will load the driver
  - It can be verified by checking *clinfo* log or dmesg log
  - Ones, driver loaded successfully, system is ready to use.

- Neo Driver Package Install (if needed)
  - Package can be downloaded from link: https://ubit-gfx.intel.com/overview/19379
  - Select the neo driver package required
  - *Published Artifacts -> linux -> ubuntu -> 18.04*



Summary

Id	Status	Begin Date	Duration	Triggered By	#Dependents	#Dependencies	owner	commit Id
9093089	Successful	2020-10-22 12:39:28	4 minutes 17 seconds	Scheduler	0 builds	0 builds	mateusz.jablonski@intel.com	8afdb2d981cffbb5ea40e0d9d9f8c748e8f68080

Cached Artifacts

Name	Size	Last Modified
Windows		2020-10-22 12:43:38
linux		2020-10-22 12:42:50
manifests		2020-10-22 12:43:20
artifactory_manifest.json	8.68 KB	2020-10-22 12:46:25

Published Artifacts

Name	Size	Last Modified
Windows		2020-10-22 12:43:38
linux		2020-10-22 12:42:50
experimental		2020-10-22 12:42:50
ubuntu		2020-10-22 12:42:48
18.04		2020-10-22 12:43:41
apt_get_config.json	279 bytes	2020-10-22 12:40:35
intel-gmmlib_20.3.2_amd64.deb	434.06 KB	2020-10-22 12:40:37
intel-igc-core_1.0.5349_amd64.deb	27.89 MB	2020-10-22 12:40:43
intel-igc-opencl_1.0.5349_amd64.deb	43.53 MB	2020-10-22 12:40:49
intel-level-zero-gpu_1.0.18254-1~bionic_amd64.deb	2.00 MB	2020-10-22 12:40:52
intel-media-1.2-018254_amd64.deb	59.19 MB	2020-10-22 12:40:59

  - Download the deb packages to system (*intel-gmmlib, intel-igc-core, intel-igc-opencl, intel-opencl*)
  - $ ***sudo dpkg -i *.deb*** -> Will install all the packages
  - Restart the system ones installation completes
  - System is now upgraded to neo driver
  - Verify the driver version by $ ***dpkg -l | grep intel-opencl***

Note : For installing the neo driver packages, all the dependencies should be removed. While trying installing deb packages, error will be shown. Needs to remove the specific package shown in error.

Python SV

GT Frequency-

Use below to dump the GT frequency

- ***import articsound.debug.domains.gfx.gt.gtPmStatus as pms***
- ***pms.dumpGtFrequency ()*** → use this to dump the frequency in a loop

- ***Pms.dumpGtFrequency(None)*** → use this to dump the frequency in only once

Code path :C:\PythonSv\articsound\debug\domains\gfx\gt\gtPmStatus.py
Note: If you are using harasser script you might see frequency change

GT Frequency-
Harasser-

- Before WL run please run this on pysv console
    - ***import articsound.debug.domains.gfx.gt.gtPmStatus as pms***
    - ***pms.gtPmHarassers(n_tiles=2)*** [By default script will take tile0 if you want to enable it for 2T pass argument as n_tiles=2 and for 4T n_tiles=4 so that harasser will be enabled for desired number of tiles]
- Start initiating the WL on Target
- After end of WL run please press Ctrl +C on pysv console
- While WL is running on target please Keep an eye on pysv console and see frequency switch and clock gating enable/disable happening

Path: C:\PythonSv\articsound\debug\domains\gfx\gt\gtPmStatus.py
Note: Make sure we issued itp.unlock() before doing this

- Gtstatus
    - ***import articsound.debug.domains.gfx.gt.gtStatus as gs***
    - ***gs.status()***

- Scandump
    - ***import articsound.debug.domains.gfx.tools.scandump.atsgfxscandumpAFD as scan***
    - ***scan.gtScandump()***

Compute WL Run Procedure

Repository :
Below test content [***Test 1-4***] are available at: https://gitlab.devtools.intel.com/ccallawa/compute-wl-packaging.git.
Use GIT to clone the repository to the target machine.
$ ***git clone https://gitlab.devtools.intel.com/ccallawa/compute-wl-packaging.git***

1. **Stream Traid**
   Directory: *STREAM/OCL/ATS*
   This currently has 4 different tests - Read-Only (RO), Write-Only (WO), 1 Read - 1 Write (Scale), 2 Read - 1 Write (Triad).
   1. Go to folder 'gen' and execute the following commands:
      a. $ ***./stream-ro-dp-cache -b1g***
      b. $ ***./stream-wo-dp-cache -b1g***
      c. $ ***./stream-scale-dp-cache -b1g***
      d. $ ***./stream-triad-dp-cache -b1g***

   Note:
   1. -b1g denoted a 1G buffer. Use this argument to change the input buffer for different operations
   2. All tests run 10 iterations by default; use -i as an additional argument to change the number of iterations

2. **SPMV**
   Directory: *SPMV/OCL/ATS*
   Pre-requisites: Add ~/SPMV/OCL/ATS/libs to LD_LIBRARY_PATH variable before running this workload by
   export LD_LIBRARY_PATH=<location>/SPMV/OCL/ATS/libs
   $ ***./spmv.ocl -t 1 -f d -k S -T 1024 --A-cacheable=false --threads-per-group 2 -m 32 band27-128m.mtx***
   $ ***./spmv.ocl -t 1 -f d -k S -T 1024 --A-cacheable=false --threads-per-group 2 -m 32 band27-256m.mtx***

3. **Resnet Training**

   Directory: *Resnet/OCL/ATS*

   ```
   $ training/examples_training64 --model=resnet50_train --input=training/imagenet --batch=32 --image_number=64 --lr=0.01 --image_set=imagenet --force_ats
   ```

4. **Resnet Inference**

   Directory: *Resnet/OCL/ATS/inference*

   Run the following commands for either normal or performance mode.

   Full Quantization mode:

   ```
   $ ./examples64 --model=resnet50-i8 --batch=64 --input=images/224/64/ --force_no_padding_for_first_convolution --force_ats –-loop=N --profiling
   ```

   Light Quantization (performance) mode:

   ```
   $ ./examples64 --model=resnet50-i8 --batch=64 --input=images/224/64/ --force_no_padding_for_first_convolution --force_ats –-loop=N --profiling --use_lightweight_quantization
   ```

   where
   loop="N" - is number of RN-50 Inference iterations (full ML stack from top to bottom with 64 images each time)
   FPS based on GPU PROFILING (END_TIMESTAMP - START_TIMESTAMP): 670.2
   profiling is optional but mandatory if you are looking for performance numbers without clIntercept. GPU active clocks are reported with signature:
   force_ats – mandatory to activate DPAS code path, otherwise it works in Gen9 mode
   Note: Increase batch sizes to 128, 256, 512 (make sure to change batch size and input images folder accordingly) for different batch experiments.

Repository :
All below test content ***[Test 5-9]*** are available at: \\bassvlab03.gar.corp.intel.com\GfxReports\Tools\Compute_Linux

1. **BlackScholes**

   export LD_LIBRARY_PATH=<path>/blackscholes
   For 1 Tile system -

   ```
   $ ./bs_ocl_1_tile  -i:500
   ```

   For 2 Tile system -

   ```
   $ ./bs_ocl_2_tile  -i:500
   ```

For GEMM Workloads install package *intel-igc-cmfe*

1. **SGEMM**

   From repository, copy GEMM directory & extract SGEMM.tar.gz ($ ***tar -xvf SGEMM.tar.gz***)
   For 1 Tile system -

   ```
   $ ./SGEMM.x --M < > --K < > --N < >--profiled_timing --eu_count < > --ocl_online_mode  --atomic_poll_interval 16 --REP_COUNT 1  --MTile_selection123 1 --outloop_per_tg_sync 255 --mtile_height 32
   ```

   For eg : $ *./SGEMM.x --M 4096 --K 4096 --N 4096 --profiled_timing --eu_count 1024 --ocl_online_mode --atomic_poll_interval 16 --REP_COUNT 1 --MTile_selection123 1 --outloop_per_tg_sync 255 --mtile_height 32*

   For 2 Tile system -

   ```
   $ ./SGEMM_2tile.x --M < > --K < > --N < > --profiled_timing --eu_count < > --ocl_online_mode  --atomic_poll_interval 16 --REP_COUNT 1  --MTile_selection123 1 --outloop_per_tg_sync 255 --mtile_height 32
   ```

   For Eg : $ *./SGEMM_2tile.x --M 4096 --K 4096 --N 4096 --profiled_timing --eu_count 1024 --ocl_online_mode --atomic_poll_interval 16 --REP_COUNT 1 --MTile_selection123 1 --outloop_per_tg_sync 255 --mtile_height 32*

i. EU_COUNT can be used as per the SKU
ii. For higher performance EU_COUNT needs to be varied per MxN and it will range from 4096 to 16384. Please contact FF Perf team for more details on this.
iii. Other MxKxN matrix combinations can be run

2. DGEMM

From repository, copy GEMM directory & extract DGEMM.tar.gz ($ ***tar -xvf DGEMM.tar.gz***)

For 1 Tile system -

```
$ ./DGEMM.x --M < > --K < > --N < > --profiled_timing --eu_count < > --ocl_online_mode --atomic_poll_interval 16 --REP_COUNT 1  --MTile_selection123 1 --outloop_per_tg_sync 255 --mtile_height 32
```

For Eg :

```
$ ./DGEMM.x --M 4096 --K 4096 --N 4096 --profiled_timing --eu_count 1024 --ocl_online_mode  --atomic_poll_interval 16 --REP_COUNT 1  --MTile_selection123 1 --outloop_per_tg_sync 255 --mtile_height 32
```

For 2 Tile system -

```
$ ./DGEMM_2tile.x --M < > --K < > --N < > --profiled_timing --eu_count < > --ocl_online_mode  --atomic_poll_interval 16 --REP_COUNT 1  --MTile_selection123 1 --outloop_per_tg_sync 255 --mtile_height 32
```

For Eg :

```
$ ./DGEMM_2tile.x --M 4096 --K 4096 --N 4096 --profiled_timing --eu_count 1024 --ocl_online_mode  --atomic_poll_interval 16 --REP_COUNT 1  --MTile_selection123 1 --outloop_per_tg_sync 255 --mtile_height 32
```

i. EU_COUNT can be used as per the SKU
ii. For higher performance EU_COUNT needs to be varied per MxN and it will range from 8192 to 32768. Please contact FF Perf team for more details on this.
iii. Other MxKxN matrix combinations can be run

3. IGEMM

From repository, copy GEMM directory & extract IGEMM.tar.gz ($ ***tar -xvf IGEMM.tar.gz***)

For 1 Tile system -

```
$ ./ATSIGEMM_online_WAR_gold_sim.x --M < > --K < > --N < > --RX < > --RY < > --profiled_timing --REP_COUNT 1 --WARMUP_COUNT 2
```

For Eg :

```
$ ./ATSIGEMM_online_WAR_gold_sim.x --M 5120 --K 4096 --N 4096 --RX 2 --RY 4  --profiled_timing --REP_COUNT 1 --WARMUP_COUNT 2
```

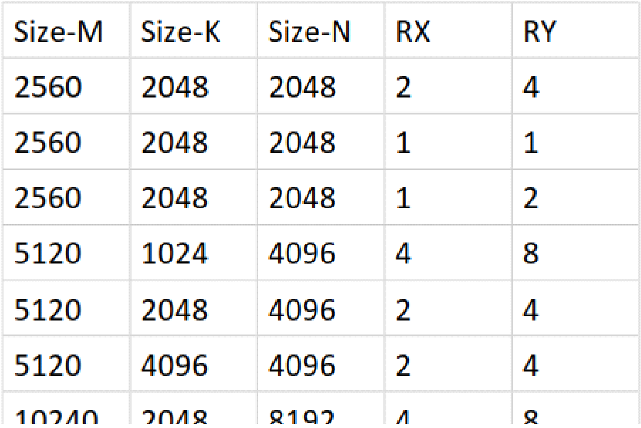
For 2 Tile system -

```
$ ./ATSIGEMM_online_WAR_gold_sim_mt.x --M < > --K < > --N < > --RX < > --RY < >  --profiled_timing --REP_COUNT 1 --WARMUP_COUNT 2
```

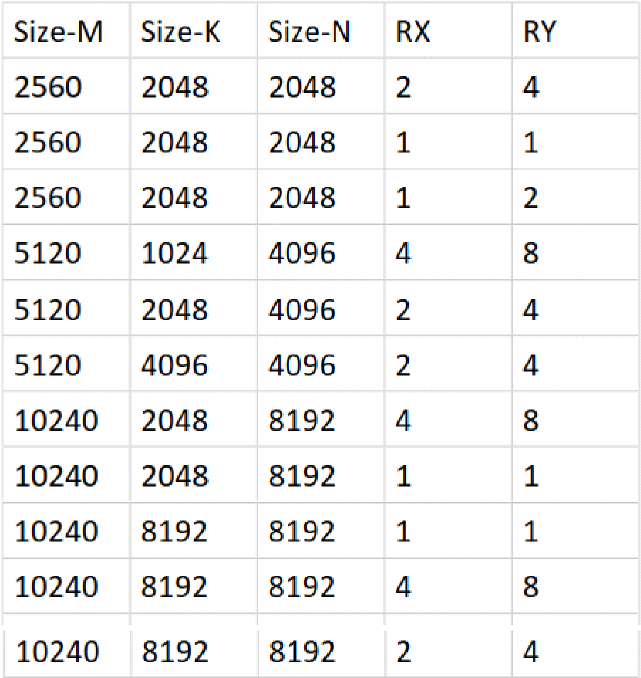
For Eg :

```
$ ./ATSIGEMM_online_WAR_gold_sim_mt.x --M 5120 --K 4096 --N 4096 --RX 2 --RY 4  --profiled_timing --REP_COUNT 1 --WARMUP_COUNT 2
```

where the matrices, sizes of RX, RY are as follows:



Size-M	Size-K	Size-N	RX	RY
2560	2048	2048	2	4
2560	2048	2048	1	1
2560	2048	2048	1	2
5120	1024	4096	4	8
5120	2048	4096	2	4
5120	4096	4096	2	4
10240	2048	8192	4	8



Size-M	Size-K	Size-N	RX	RY
2560	2048	2048	2	4
2560	2048	2048	1	1
2560	2048	2048	1	2
5120	1024	4096	4	8
5120	2048	4096	2	4
5120	4096	4096	2	4
10240	2048	8192	4	8
10240	2048	8192	1	1
10240	8192	8192	1	1
10240	8192	8192	4	8
10240	8192	8192	2	4

Note:
1. The higher matrices 5K/10K take anywhere from 30m to over an hour to run

4. BGEMM

From repository, copy GEMM directory & extract BGEMM.tar.gz ($ ***tar -xvf BGEMM.tar.gz***)

For 1 Tile system -

```
$ ./ATSBGEMM_online_WAR_gold_sim.x --M < > --K < > --N < > --RX < > --RY < >  --profiled_timing --REP_COUNT 1 --WARMUP_COUNT 2
```

For Eg :

```
$ ./ATSBGEMM_online_WAR_gold_sim.x --M 5120 --K 4096 --N 4096 --RX 2 --RY 4  --profiled_timing --REP_COUNT 1 --WARMUP_COUNT 2
```

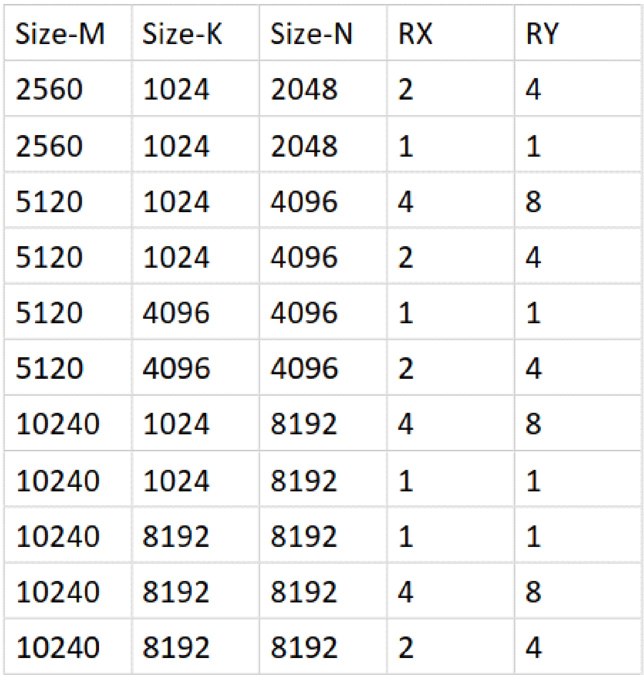
For 2 Tile system -

```
$ ./ATSBGEMM_online_WAR_gold_sim_mt.x --M < > --K < > --N < > --RX < > --RY < >  --profiled_timing --REP_COUNT 1 --WARMUP_COUNT 2
```

For Eg :

```
$ ./ATSBGEMM_online_WAR_gold_sim_mt.x --M 5120 --K 4096 --N 4096 --RX 2 --RY 4  --profiled_timing --REP_COUNT 1 --WARMUP_COUNT 2
```

where the matrices, sizes of RX, RY are as follows:



Size-M	Size-K	Size-N	RX	RY
2560	1024	2048	2	4
2560	1024	2048	1	1
5120	1024	4096	4	8
5120	1024	4096	2	4
5120	4096	4096	1	1
5120	4096	4096	2	4
10240	1024	8192	4	8
10240	1024	8192	1	1
10240	8192	8192	1	1
10240	8192	8192	4	8
10240	8192	8192	2	4

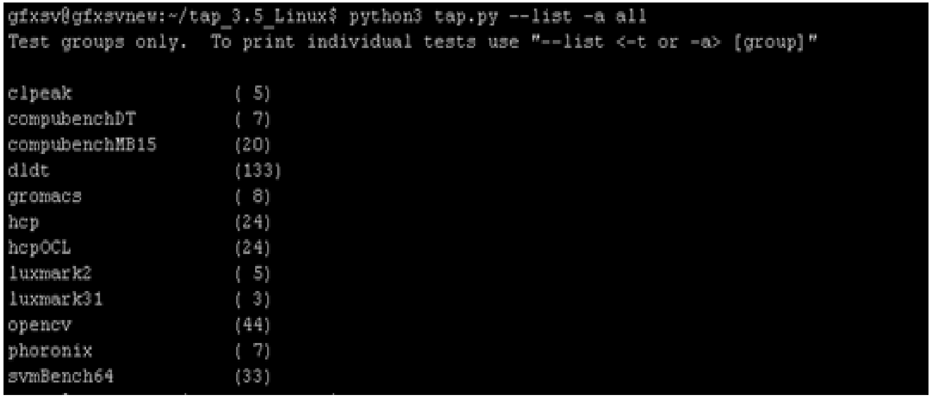
Note:
1. The higher matrices 5K/10K take anywhere from 30m to over an hour to run

5. TAP WLs

Pre-requisites: Python3

1. Copy the .tar.gz Linux WL file from \\amr.corp.intel.com\EC\proj\fm\VPG\ComputeTraces\tap
   Also copied the tar file to location: \\bassvlab03.gar.corp.intel.com\GfxReports\Tools\Compute_Linux : TAP_3.5_Linux.tar.gz

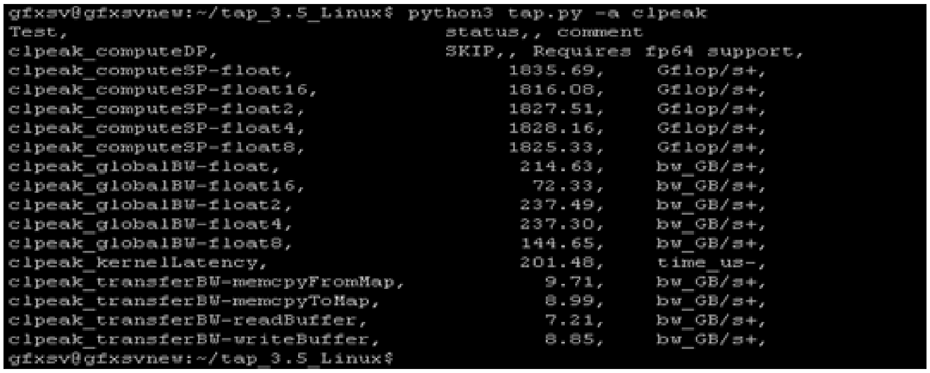
2. Extract to a local folder (use $ ***tar -xvf <filename.tar.gz>***)
   $ ***python3 tap.py --list -a all*** [to list all workload groups]



```
gfxsv@gfxsvnew:~/tap_3.5_Linux$ python3 tap.py --list -a all
Test groups only.  To print individual tests use "--list <-t or -a> [group]"

clpeak                 ( 5)
compubenchDT           ( 7)
compubenchMB15         (20)
dldt                   (133)
gromacs                ( 8)
hcp                    (24)
hcpOCL                 (24)
luxmark2               ( 5)
luxmark31              ( 3)
opencv                 (44)
phoronix               ( 7)
svmBench64             (33)
```

1. For running a specific WL set, the below command can be used
   $ ***python3 tap.py -a <WL group name>*** [to run specific WL set]

   For Eg: $ ***python3 tap.py -a clpeak*** (execute all sub-tests within clpeak)



```
gfxsv@gfxsvnew:~/tap_3.5_Linux$ python3 tap.py -a clpeak
Test,                                   status,, comment
clpeak_computeDP,                       SKIP,, Requires fp64 support,
clpeak_computeSP-float,                        1835.69,     Gflop/s+,
clpeak_computeSP-float16,                      1816.08,     Gflop/s+,
clpeak_computeSP-float2,                       1827.51,     Gflop/s+,
clpeak_computeSP-float4,                       1828.16,     Gflop/s+,
clpeak_computeSP-float8,                       1825.33,     Gflop/s+,
clpeak_globalBW-float,                          214.63,     bw_GB/s+,
clpeak_globalBW-float16,                         72.33,     bw_GB/s+,
clpeak_globalBW-float2,                         237.49,     bw_GB/s+,
clpeak_globalBW-float4,                         237.30,     bw_GB/s+,
clpeak_globalBW-float8,                         144.65,     bw_GB/s+,
clpeak_kernelLatency,                           201.48,     time_us-,
clpeak_transferBW-memcpyFromMap,                  9.71,     bw_GB/s+,
clpeak_transferBW-memcpyToMap,                    8.99,     bw_GB/s+,
clpeak_transferBW-readBuffer,                     7.21,     bw_GB/s+,
clpeak_transferBW-writeBuffer,                    8.85,     bw_GB/s+,
gfxsv@gfxsvnew:~/tap_3.5_Linux$
```

2. If any particular sub-test needs to be initiated within a test, the below command can be used

   $ ***python3 tap.py -a <subtestname>*** (execute specific sub-test within test)

   For Eg: $ ***python3 tap.py -a clpeak_kernelLatency*** (execute only specific sub-test)

3. Above methods are applicable for all the workload groups in the tap based test.
4. Exception for Gromacs workload -
   1) Add "*reset=0*" in KMD parameter.
   2) Use below command for running gromacs WL -

      $ ***python3 tap.py -a gromacs --timeout=10380***

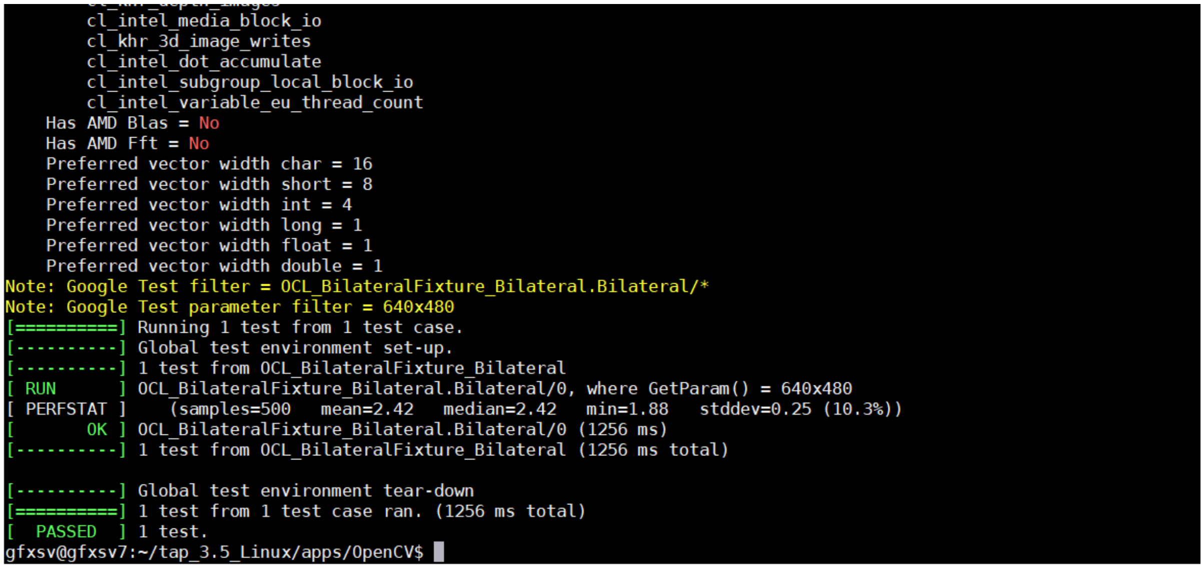
If getting any value as 'NONE' or hang during execution with tap, needs to check the test individually running via apps.

a. <u>Opencv</u>

1) Goto directory *~/tap_3.5_Linux/apps/OpenCV*
2) $ ***export LD_LIBRARY_PATH=/home/gfxsv/tap_3.5_Linux/apps/OpenCV/***
3) For running the test use below command

   $ ***./<testname>.sh***

   For eg: $ *./bilateralFilter1.sh*



```
        cl_intel_media_block_io
        cl_khr_3d_image_writes
        cl_intel_dot_accumulate
        cl_intel_subgroup_local_block_io
        cl_intel_variable_eu_thread_count
    Has AMD Blas = No
    Has AMD Fft = No
    Preferred vector width char = 16
    Preferred vector width short = 8
    Preferred vector width int = 4
    Preferred vector width long = 1
    Preferred vector width float = 1
    Preferred vector width double = 1
Note: Google Test filter = OCL_BilateralFixture_Bilateral.Bilateral/*
Note: Google Test parameter filter = 640x480
[==========] Running 1 test from 1 test case.
[----------] Global test environment set-up.
[----------] 1 test from OCL_BilateralFixture_Bilateral
[ RUN      ] OCL_BilateralFixture_Bilateral.Bilateral/0, where GetParam() = 640x480
[ PERFSTAT ]    (samples=500   mean=2.42   median=2.42   min=1.88   stddev=0.25 (10.3%))
[       OK ] OCL_BilateralFixture_Bilateral.Bilateral/0 (1256 ms)
[----------] 1 test from OCL_BilateralFixture_Bilateral (1256 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test case ran. (1256 ms total)
[  PASSED  ] 1 test.
gfxsv@gfxsv7:~/tap_3.5_Linux/apps/OpenCV$
```

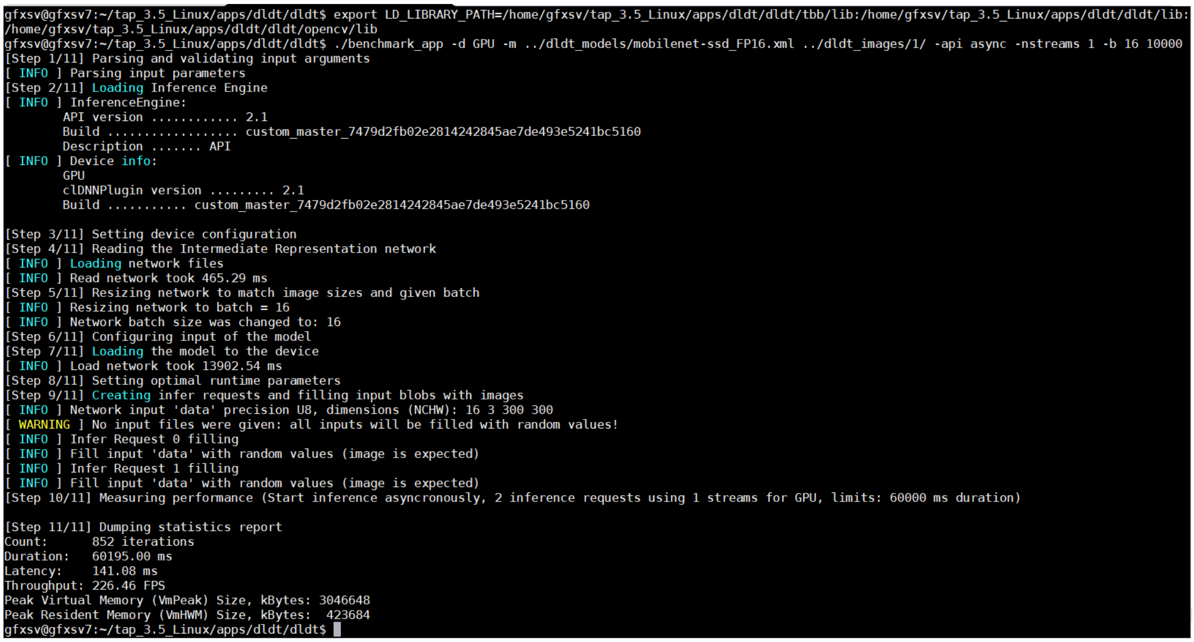
b. <u>dldt</u>

1) Goto directory *~/tap_3.5_Linux/apps/dldt/dldt*
2) $ ***export LD_LIBRARY_PATH=/home/gfxsv/tap_3.5_Linux/apps/dldt/dldt/tbb/lib:/home/gfxsv/tap_3.5_Linux/apps/dldt/dldt/lib:/home/gfxsv/tap_3.5_Linux/apps/dldt/dldt/opencv/lib***
3) For running the test, use below command

$ ***./benchmark_app -d GPU -m ../dldt_models/<testname.xml> ../dldt_images/1/ -api async -nstreams 1 -b <value> 10000*** (test name in highlighted in yellow color and value of b in blue color)

Testname can be obtained from *~/tap_3.5 _Linux/apps/dldt/dldt_models/ -> <testname>.xml*

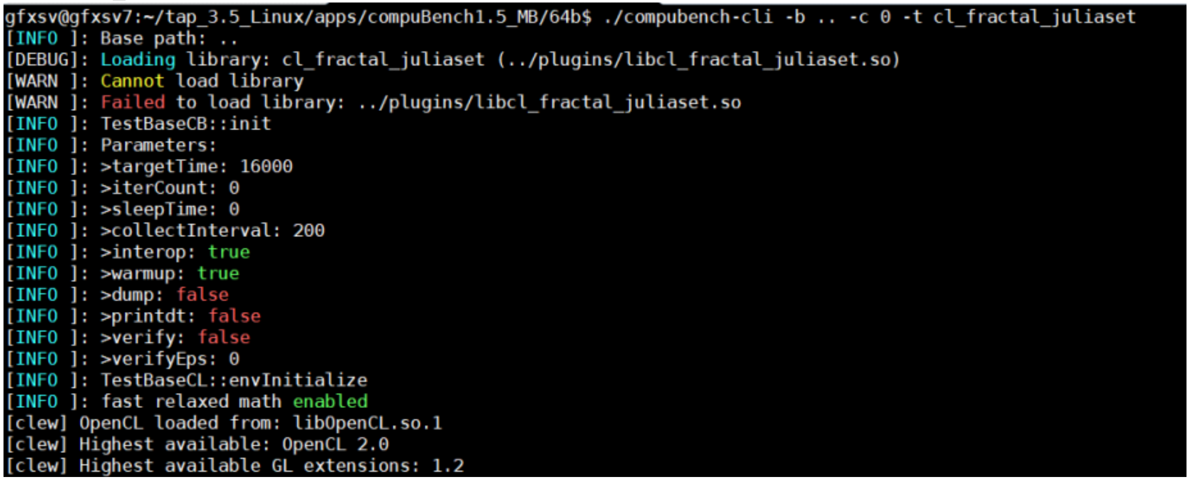
For Eg: $ *./benchmark_app -d GPU -m ../dldt_models/mobilenet-ssd_FP16.xml ../dldt_images/1/ -api async -nstreams 1 -b 16 10000*



```
gfxsv@gfxsv7:~/tap_3.5_Linux/apps/dldt/dldt$ export LD_LIBRARY_PATH=/home/gfxsv/tap_3.5_Linux/apps/dldt/dldt/tbb/lib:/home/gfxsv/tap_3.5_Linux/apps/dldt/dldt/lib:
/home/gfxsv/tap_3.5_Linux/apps/dldt/dldt/opencv/lib
gfxsv@gfxsv7:~/tap_3.5_Linux/apps/dldt/dldt$ ./benchmark_app -d GPU -m ../dldt_models/mobilenet-ssd_FP16.xml ../dldt_images/1/ -api async -nstreams 1 -b 16 10000
[Step 1/11] Parsing and validating input arguments
[ INFO ] Parsing input parameters
[Step 2/11] Loading Inference Engine
[ INFO ] InferenceEngine:
        API version ............ 2.1
        Build .................. custom_master_7479d2fb02e2814242845ae7de493e5241bc5160
        Description ....... API
[ INFO ] Device info:
        GPU
        clDNNPlugin version ......... 2.1
        Build ........... custom_master_7479d2fb02e2814242845ae7de493e5241bc5160

[Step 3/11] Setting device configuration
[Step 4/11] Reading the Intermediate Representation network
[ INFO ] Loading network files
[ INFO ] Read network took 465.29 ms
[Step 5/11] Resizing network to match image sizes and given batch
[ INFO ] Resizing network to batch = 16
[ INFO ] Network batch size was changed to: 16
[Step 6/11] Configuring input of the model
[Step 7/11] Loading the model to the device
[ INFO ] Load network took 13902.54 ms
[Step 8/11] Setting optimal runtime parameters
[Step 9/11] Creating infer requests and filling input blobs with images
[ INFO ] Network input 'data' precision U8, dimensions (NCHW): 16 3 300 300
[ WARNING ] No input files were given: all inputs will be filled with random values!
[ INFO ] Infer Request 0 filling
[ INFO ] Fill input 'data' with random values (image is expected)
[ INFO ] Infer Request 1 filling
[ INFO ] Fill input 'data' with random values (image is expected)
[Step 10/11] Measuring performance (Start inference asyncronously, 2 inference requests using 1 streams for GPU, limits: 60000 ms duration)

[Step 11/11] Dumping statistics report
Count:      852 iterations
Duration:   60195.00 ms
Latency:    141.08 ms
Throughput: 226.46 FPS
Peak Virtual Memory (VmPeak) Size, kBytes: 3046648
Peak Resident Memory (VmHWM) Size, kBytes:  423684
gfxsv@gfxsv7:~/tap_3.5_Linux/apps/dldt/dldt$
```

c. **CompubenchMB15**

1) Goto *~/tap_3.5_Linux/apps/compuBench1.5_MB/64b*
2) Open README file steps are written there for execution
    a) $ ***./compubench-cli -b . -c 0 -t <testID>***
    b) Test_id can be obtained from README file



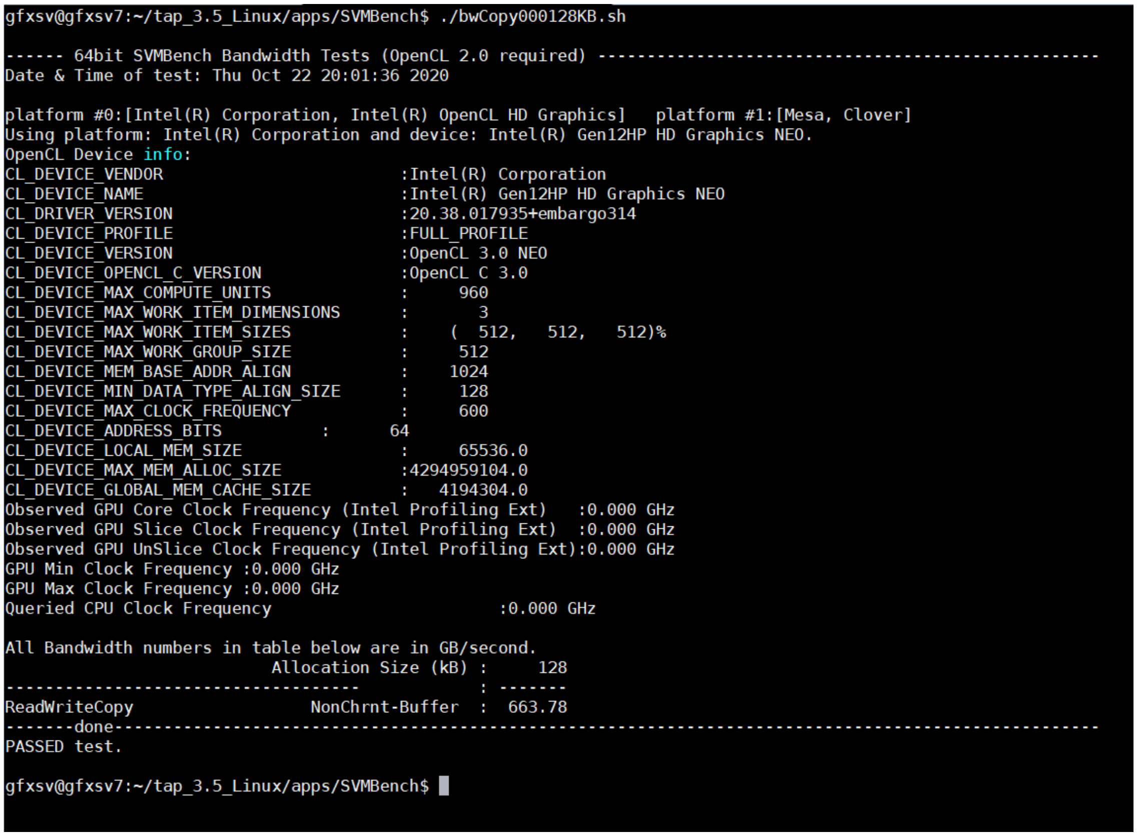
```
gfxsv@gfxsv7:~/tap_3.5_Linux/apps/compuBench1.5_MB/64b$ ./compubench-cli -b .. -c 0 -t cl_fractal_juliaset
[INFO ]: Base path: ..
[DEBUG]: Loading library: cl_fractal_juliaset (../plugins/libcl_fractal_juliaset.so)
[WARN ]: Cannot load library
[WARN ]: Failed to load library: ../plugins/libcl_fractal_juliaset.so
[INFO ]: TestBaseCB::init
[INFO ]: Parameters:
[INFO ]: >targetTime: 16000
[INFO ]: >iterCount: 0
[INFO ]: >sleepTime: 0
[INFO ]: >collectInterval: 200
[INFO ]: >interop: true
[INFO ]: >warmup: true
[INFO ]: >dump: false
[INFO ]: >printdt: false
[INFO ]: >verify: false
[INFO ]: >verifyEps: 0
[INFO ]: TestBaseCL::envInitialize
[INFO ]: fast relaxed math enabled
[clew] OpenCL loaded from: libOpenCL.so.1
[clew] Highest available: OpenCL 2.0
[clew] Highest available GL extensions: 1.2
```

a. **CompubenchDT**

1) Same as that of CompubenchMB
2) Goto *~/tap_3.5_Linux/apps/CompubenchCL157Desktop/*
3) Open README file. Steps of execution is mentioned there.
4) Use the below command for executing
    a) $ ***./compubench-cli-glfw3 -b . 0 -t <testID>***

b. **Svmbench64**

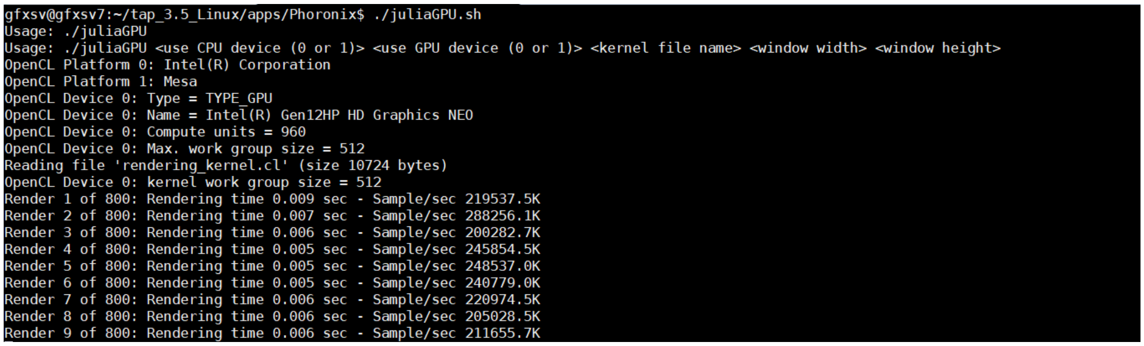
1) Goto *~/tap_3.5_Linux/apps/SVMBench/*
2) Use the below command for test execution
    a) $ ***./<testname>.sh***
    For Eg: $ *./bwCopy000128.sh*



```
gfxsv@gfxsv7:~/tap_3.5_Linux/apps/SVMBench$ ./bwCopy000128KB.sh

------ 64bit SVMBench Bandwidth Tests (OpenCL 2.0 required) -------------------------------------------------
Date & Time of test: Thu Oct 22 20:01:36 2020

platform #0:[Intel(R) Corporation, Intel(R) OpenCL HD Graphics]   platform #1:[Mesa, Clover]
Using platform: Intel(R) Corporation and device: Intel(R) Gen12HP HD Graphics NEO.
OpenCL Device info:
CL_DEVICE_VENDOR                        :Intel(R) Corporation
CL_DEVICE_NAME                          :Intel(R) Gen12HP HD Graphics NEO
CL_DRIVER_VERSION                       :20.38.017935+embargo314
CL_DEVICE_PROFILE                       :FULL_PROFILE
CL_DEVICE_VERSION                       :OpenCL 3.0 NEO
CL_DEVICE_OPENCL_C_VERSION              :OpenCL C 3.0
CL_DEVICE_MAX_COMPUTE_UNITS             :     960
CL_DEVICE_MAX_WORK_ITEM_DIMENSIONS      :       3
CL_DEVICE_MAX_WORK_ITEM_SIZES           :    (  512,   512,   512)%
CL_DEVICE_MAX_WORK_GROUP_SIZE           :     512
CL_DEVICE_MEM_BASE_ADDR_ALIGN           :    1024
CL_DEVICE_MIN_DATA_TYPE_ALIGN_SIZE      :     128
CL_DEVICE_MAX_CLOCK_FREQUENCY           :     600
CL_DEVICE_ADDRESS_BITS          :      64
CL_DEVICE_LOCAL_MEM_SIZE                :     65536.0
CL_DEVICE_MAX_MEM_ALLOC_SIZE            :4294959104.0
CL_DEVICE_GLOBAL_MEM_CACHE_SIZE         :   4194304.0
Observed GPU Core Clock Frequency (Intel Profiling Ext)   :0.000 GHz
Observed GPU Slice Clock Frequency (Intel Profiling Ext)  :0.000 GHz
Observed GPU UnSlice Clock Frequency (Intel Profiling Ext):0.000 GHz
GPU Min Clock Frequency :0.000 GHz
GPU Max Clock Frequency :0.000 GHz
Queried CPU Clock Frequency                       :0.000 GHz

All Bandwidth numbers in table below are in GB/second.
                           Allocation Size (kB) :     128
------------------------------------            : -------
ReadWriteCopy                  NonChrnt-Buffer  :  663.78
-------done--------------------------------------------------------------------------------------------------
PASSED test.

gfxsv@gfxsv7:~/tap_3.5_Linux/apps/SVMBench$
```

a. **Phoronix**

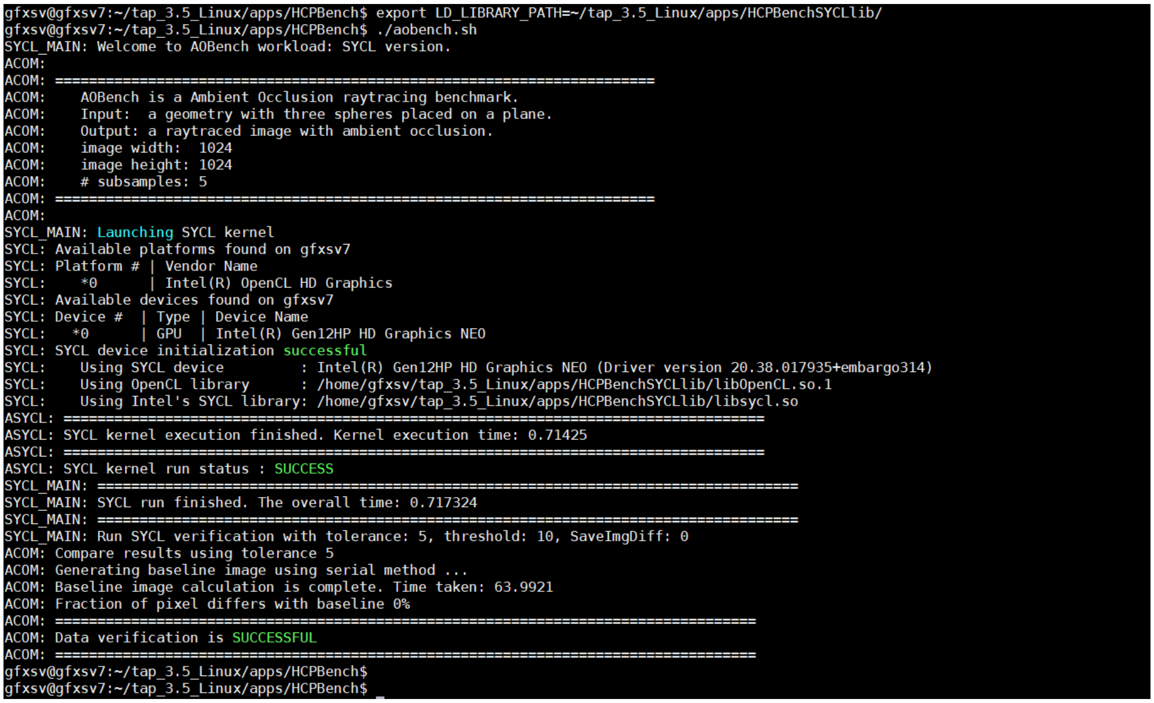
1) Goto *~/tap_3.5_Linux/apps/Phoronix/*
2) Use the below command for test execution
    a) $ ./<testname>.sh
    For Eg: $ *./juliaGPU.sh*



```
gfxsv@gfxsv7:~/tap_3.5_Linux/apps/Phoronix$ ./juliaGPU.sh
Usage: ./juliaGPU
Usage: ./juliaGPU <use CPU device (0 or 1)> <use GPU device (0 or 1)> <kernel file name> <window width> <window height>
OpenCL Platform 0: Intel(R) Corporation
OpenCL Platform 1: Mesa
OpenCL Device 0: Type = TYPE_GPU
OpenCL Device 0: Name = Intel(R) Gen12HP HD Graphics NEO
OpenCL Device 0: Compute units = 960
OpenCL Device 0: Max. work group size = 512
Reading file 'rendering_kernel.cl' (size 10724 bytes)
OpenCL Device 0: kernel work group size = 512
Render 1 of 800: Rendering time 0.009 sec - Sample/sec 219537.5K
Render 2 of 800: Rendering time 0.007 sec - Sample/sec 288256.1K
Render 3 of 800: Rendering time 0.006 sec - Sample/sec 200282.7K
Render 4 of 800: Rendering time 0.005 sec - Sample/sec 245854.5K
Render 5 of 800: Rendering time 0.005 sec - Sample/sec 248537.0K
Render 6 of 800: Rendering time 0.005 sec - Sample/sec 240779.0K
Render 7 of 800: Rendering time 0.006 sec - Sample/sec 220974.5K
Render 8 of 800: Rendering time 0.006 sec - Sample/sec 205028.5K
Render 9 of 800: Rendering time 0.006 sec - Sample/sec 211655.7K
```

b. **HCPBench**

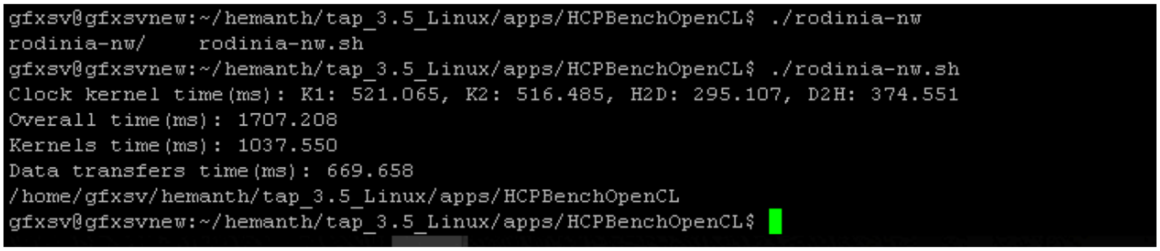
1) Goto *~/tap_3.5_Linux/apps/HCPBench/*
2) $ ***export LD_LIBRARY_PATH=~/tap_3.5_Linux/apps/HCPBenchSYCLlib/***
3) Use the below command for test execution
    a) $ ***./<testcase.sh>***
    For Eg: $ *./aobench.sh*



```
gfxsv@gfxsv7:~/tap_3.5_Linux/apps/HCPBench$ export LD_LIBRARY_PATH=~/tap_3.5_Linux/apps/HCPBenchSYCLlib/
gfxsv@gfxsv7:~/tap_3.5_Linux/apps/HCPBench$ ./aobench.sh
SYCL_MAIN: Welcome to AOBench workload: SYCL version.
ACOM:
ACOM: =======================================================================
ACOM:    AOBench is a Ambient Occlusion raytracing benchmark.
ACOM:    Input:  a geometry with three spheres placed on a plane.
ACOM:    Output: a raytraced image with ambient occlusion.
ACOM:    image width:  1024
ACOM:    image height: 1024
ACOM:    # subsamples: 5
ACOM: =======================================================================
ACOM:
SYCL_MAIN: Launching SYCL kernel
SYCL: Available platforms found on gfxsv7
SYCL: Platform # | Vendor Name
SYCL:    *0      | Intel(R) OpenCL HD Graphics
SYCL: Available devices found on gfxsv7
SYCL: Device #  | Type | Device Name
SYCL:   *0      | GPU  | Intel(R) Gen12HP HD Graphics NEO
SYCL: SYCL device initialization successful
SYCL:    Using SYCL device         : Intel(R) Gen12HP HD Graphics NEO (Driver version 20.38.017935+embargo314)
SYCL:    Using OpenCL library      : /home/gfxsv/tap_3.5_Linux/apps/HCPBenchSYCLlib/libOpenCL.so.1
SYCL:    Using Intel's SYCL library: /home/gfxsv/tap_3.5_Linux/apps/HCPBenchSYCLlib/libsycl.so
ASYCL: ===================================================================================
ASYCL: SYCL kernel execution finished. Kernel execution time: 0.71425
ASYCL: ===================================================================================
ASYCL: SYCL kernel run status : SUCCESS
SYCL_MAIN: ===================================================================================
SYCL_MAIN: SYCL run finished. The overall time: 0.717324
SYCL_MAIN: ===================================================================================
SYCL_MAIN: Run SYCL verification with tolerance: 5, threshold: 10, SaveImgDiff: 0
ACOM: Compare results using tolerance 5
ACOM: Generating baseline image using serial method ...
ACOM: Baseline image calculation is complete. Time taken: 63.9921
ACOM: Fraction of pixel differs with baseline 0%
ACOM: ===================================================================================
ACOM: Data verification is SUCCESSFUL
ACOM: ===================================================================================
gfxsv@gfxsv7:~/tap_3.5_Linux/apps/HCPBench$
gfxsv@gfxsv7:~/tap_3.5_Linux/apps/HCPBench$ _
```

c. **HCPBenchOCL**

1) Goto *~/tap_3.5_Linux/apps/HCPBenchOpenCL/*
2) $ ***export LD_LIBRARY_PATH=~/tap_3.5_Linux/apps/HCPBenchSYCLlib/***
3) Use the below command for test execution
    a) $ ***./<testcase.sh>***
    For Eg: $ *./rodinia-nw.sh*



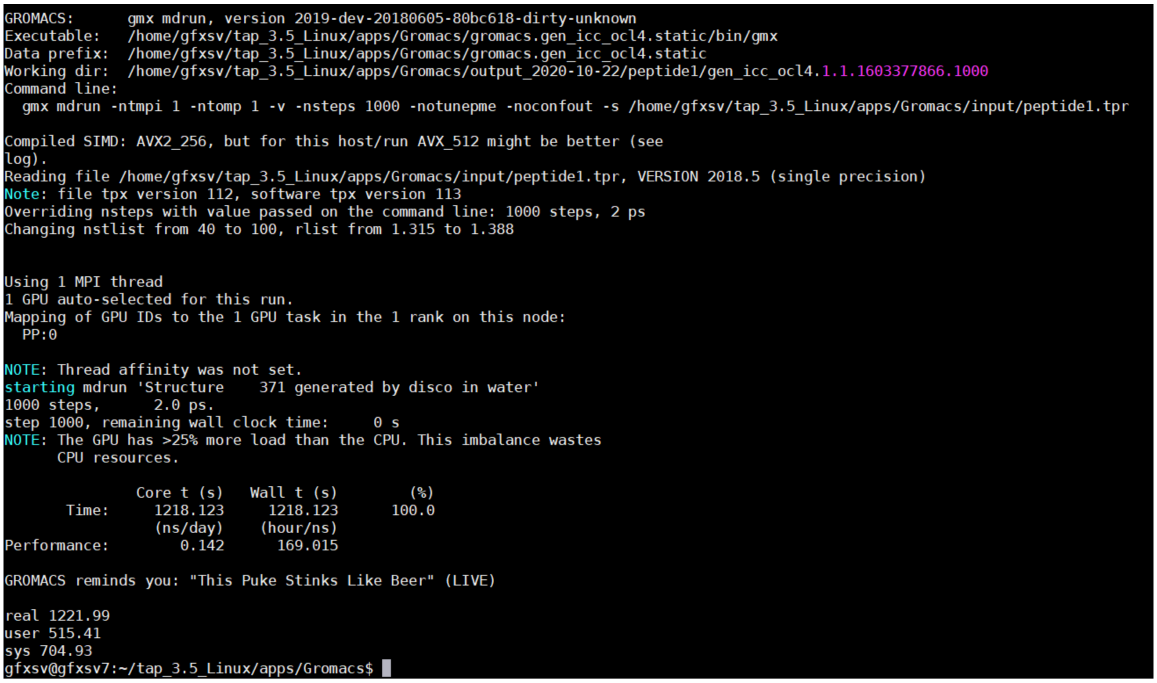
```
gfxsv@gfxsvnew:~/hemanth/tap_3.5_Linux/apps/HCPBenchOpenCL$ ./rodinia-nw
rodinia-nw/    rodinia-nw.sh
gfxsv@gfxsvnew:~/hemanth/tap_3.5_Linux/apps/HCPBenchOpenCL$ ./rodinia-nw.sh
Clock kernel time(ms): K1: 521.065, K2: 516.485, H2D: 295.107, D2H: 374.551
Overall time(ms): 1707.208
Kernels time(ms): 1037.550
Data transfers time(ms): 669.658
/home/gfxsv/hemanth/tap_3.5_Linux/apps/HCPBenchOpenCL
gfxsv@gfxsvnew:~/hemanth/tap_3.5_Linux/apps/HCPBenchOpenCL$
```

d. **Gromacs**

1) Goto *~/tap_3.5_Linux/apps/Gromacs/*
2) Use the below command to run test
    a) $ ***./run0.sh <testname> <value>***
    For Eg: $ *./run0.sh peptide1 1000*

Note:

- testname - located in input directory
- value -
    - Use 1000 for 1K
    - Use 2000 for 2K



```
GROMACS:      gmx mdrun, version 2019-dev-20180605-80bc618-dirty-unknown
Executable:   /home/gfxsv/tap_3.5_Linux/apps/Gromacs/gromacs.gen_icc_ocl4.static/bin/gmx
Data prefix:  /home/gfxsv/tap_3.5_Linux/apps/Gromacs/gromacs.gen_icc_ocl4.static
Working dir:  /home/gfxsv/tap_3.5_Linux/apps/Gromacs/output_2020-10-22/peptide1/gen_icc_ocl4.1.1.1603377866.1000
Command line:
  gmx mdrun -ntmpi 1 -ntomp 1 -v -nsteps 1000 -notunepme -noconfout -s /home/gfxsv/tap_3.5_Linux/apps/Gromacs/input/peptide1.tpr

Compiled SIMD: AVX2_256, but for this host/run AVX_512 might be better (see
log).
Reading file /home/gfxsv/tap_3.5_Linux/apps/Gromacs/input/peptide1.tpr, VERSION 2018.5 (single precision)
Note: file tpx version 112, software tpx version 113
Overriding nsteps with value passed on the command line: 1000 steps, 2 ps
Changing nstlist from 40 to 100, rlist from 1.315 to 1.388


Using 1 MPI thread
1 GPU auto-selected for this run.
Mapping of GPU IDs to the 1 GPU task in the 1 rank on this node:
  PP:0

NOTE: Thread affinity was not set.
starting mdrun 'Structure    371 generated by disco in water'
1000 steps,      2.0 ps.
step 1000, remaining wall clock time:     0 s
NOTE: The GPU has >25% more load than the CPU. This imbalance wastes
      CPU resources.

               Core t (s)   Wall t (s)        (%)
       Time:     1218.123     1218.123      100.0
                 (ns/day)    (hour/ns)
Performance:        0.142      169.015

GROMACS reminds you: "This Puke Stinks Like Beer" (LIVE)

real 1221.99
user 515.41
sys 704.93
gfxsv@gfxsv7:~/tap_3.5_Linux/apps/Gromacs$ 
```

e. **<u>Clpeak</u>**
   1) Goto *~/tap_3.5_Linux/apps/clpeak/64b*
   2) Use the below command to run test
      a) $ ***./<testname>.sh***
         For Eg: $ *./global-bandwidth.sh*

f. **<u>Luxmark2</u>**
   1) Goto *~/tap_3.5_Linux/apps/luxmark2*
   2) Open the README file & check for procedure
   3) *$* ***export LD_LIBRARY_PATH=~/tap_3.5_Linux/apps/luxmark2/***
   4) Use the below command to run test
      a) $ ***./luxmark --mode=BENCHMARK_OCL_GPU --log --scene=<testID>***
         For Eg: $ *./luxmark --mode=BENCHMARK_OCL_GPU --log --scene=SCENE_SALA*

   Note:
   - <testID> can be seen in README file
   - Use MobaXterm to run the test : since it require an X-Server to run the WL in other terminal.

g. **<u>Luxmark31</u>**
   1) Goto *~/tap_3.5_Linux/apps/luxmark31*
   2) Open the README file & check for procedure
   3) *$* ***export LD_LIBRARY_PATH=~/tap_3.5_Linux/apps/luxmark31/lib***
   4) Use the below command to run test
      a) $ ***./luxmark --mode=BENCHMARK_OCL_GPU --single-run --scene=<testID>***
         For Eg: $ *./luxmark --mode=BENCHMARK_OCL_GPU --single-run --scene=HOTEL*

   Note:
   - <testID> can be seen in README file
   - Use MobaXterm to run the test : since it require an X-Server to run the WL in other terminal.

<u>MultiCtxt+Concurrency WL Run Procedure</u>

For Compute + Compute

1. Open 1st terminal ;

    a. Run Compute WL
2. Open 2nd terminal ;
    a. Run another set of Compute WL

This will run both the tests at the same time

For Compute + Media

1. Open 1st terminal
    a. Run a compute WL
2. Open 2nd terminal
    a. Run Media WL

This will run both the Compute & media WL at same time

For Eg:

Compute + Compute -

1. Open 1st terminal
2. Initiate Opencv test

    ```
    $ python3 tap.py -a opencv
    ```

3. Open 2nd terminal in same system
4. Initiate Svmbench test

    ```
    $ python3 tap.py -a svmbench64
    ```

Both the tests should run parallelly without any Issue

Compute + Media -

1. Open 1st terminal
2. Initiate Opencv(compute) test

    ```
    $ python3 tap.py -a opencv
    ```

3. Please check the binary of Media WL with Media team
4. Open 2nd terminal & Run Media WL
5. Media WL can be Decode/Encode/VPP

Both the Compute & Media WLs should run parallelly without any Issue

MOBIN P ABRAHAM