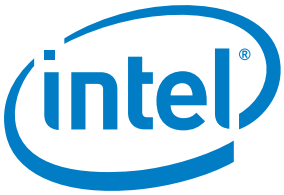

Intel® Vision Accelerator Design with Intel® Movidius™ Myriad™ VPU Scheduler

User Guide

September 2019

Document Number: 613759-0.5

Contents

1.0 Introduction 6

2.0 Squeeze Scheduler 7

2.1 Use Case 7
2.2 HAL Service Configuration File (`hddl_service.config`) Configuration Settings 7
2.3 Inference Engine API Configuration 7
2.4 Pros 7
2.5 Cons 7
2.6 Note 8

3.0 Tag Scheduler 9

3.1 Use Cases 9
3.2 HAL Service Configuration File Configuration Settings 9
3.3 Inference Engine API Configuration 9
3.4 Pro 10
3.5 Cons 10
3.6 Note 10

4.0 Stream Scheduler 11

4.1 Use Case 11
4.2 HAL Service Configuration File Configuration Settings 11
4.3 Inference Engine API Configuration 11
4.4 Pros 11
4.5 Con 12
4.6 Note 12

5.0 SGAD Scheduler 13

5.1 Use Cases 13
5.2 HAL Service Configuration File Configuration Settings 13
5.3 Inference Engine API Configuration 13
5.4 Pros 13
5.5 Cons 13
5.6 Note 14

6.0 ByPass Scheduler 15

6.1 Use Case 15
6.2 HAL Service Configuration File Configuration Settings 15
6.3 Inference Engine API Configuration 15
6.4 Pros 15
6.5 Cons 15
6.6 Note 16

Tables

Table 1. Scheduler Overview 6

Revision History

Date	Revision	Description
June 2019	0.5	Initial release.

§

1.0 Introduction

The HDDL Service in the Intel® Distribution of OpenVINO™ toolkit includes five schedulers, each of which manages a specific number of Intel® Movidius™ Myriad™ X VPUs.

The following table provides a brief description of each scheduler. The remainder of this document goes into detail about each scheduler.

Table 1. Scheduler Overview

Scheduler	Description	Pros	Cons
Squeeze	Schedules workload among networks and devices smartly and periodically	Scheduling is free for users	Only two networks can run on an Intel® Movidius™ Myriad™ X VPU. Potential stability problem with more than one network per VPU
Tag	`hddl_service.config` controls deploying networks to devices	Easy to deploy networks among devices without programming code	Only one network allowed per Intel® Movidius™ Myriad™ X VPU. Deployment is fixed after service starts.
Stream	Inferences from one video stream are processed on one device, for context-waring networks.	Suitable for context awareness networks	Only one network allowed per Intel® Movidius™ Myriad™ X VPU.
SGAD	Load networks on all devices at once.	Simple	High memory consumption. Only one network allowed per Intel® Movidius™ Myriad™ X VPU, due to potential stability problems. No direct way to acquire the number of available devices
ByPass	Free for users to control loading and unloading networks directly to devices.	Full-size flexibility for users to control individual device.	You must handle your own scheduling. Only one network allowed per Intel® Movidius™ Myriad™ X VPU, due to potential stability problems. No direct way to get the number of available devices.

2.0 Squeeze Scheduler

Application developers often want to program the business logic of their applications, but not deal with deploying the neural networks among VPUs. In these cases, it is best if the HAL systems handle scheduling in a way that achieves high inference throughput.

The Squeeze Scheduler manages all VPUs that aren't assigned to other schedulers. This scheduler collects the workloads for all networks at a time interval, as defined in the HAL Service configuration file (`hddl_service.config`). The Squeeze Scheduler uses the information in this file to schedule the networks and VPUs, based on workload, to achieve a workload balance.

2.1 Use Case

You don't want to manage the scheduling tasks between networks and devices.

2.2 HAL Service Configuration File (`hddl_service.config`) Configuration Settings

Add these settings to the `hddl_service.config` file.

```
"device_schedule_interval":    5000, // in milliseconds
"max_cycle_switch_out":        3,    // See Note.
"max_task_number_switch_out":  20   // See Note.
```

2.3 Inference Engine API Configuration

None

2.4 Pros

- Handles scheduling.
- You can configure key scheduling parameters for basic fine tuning in `hddl_service.config`

2.5 Cons

- The scheduling policy can't meet all requirements for all use cases. For cases that the Squeeze Scheduler can't satisfy, fine tune the parameters, or use the ByPass Scheduler.
- A maximum of two networks can run on one Intel® Movidius™ Myriad™ X VPU.

2.6 Note

If some graphs can't be loaded to the Intel® Movidius™ Myriad™ X VPU device, those graphs must be unloaded. When the unloaded graphs meet one of following conditions, they are loaded to run:

- The time unloaded is longer than `<max_cycle_switch_out>` times and an inference task is pending.
- The number of the inference task for this graph is larger than `<max_task_number_switch_out>`.

3.0 Tag Scheduler

Some deep learning tasks require a stable computing resource. For example, in a surveillance system, the number of IP cameras connected to an NVR (video gateway) system is usually fixed after deployment.

To perform face detection on each frame in each video stream, the face detection workload is calculable, given:

- The number of video streams
- The video stream frame rate
- The deep learning network used for “face detect”
- The processing capability of one VPU for this network (e.g. fps)

The Tag Scheduler manages VPUs that can process predictable tasks. This scheduler accepts a neural network graph that is attached with a valid tag, and allocates the graph to the specified number of VPUs, according to a configuration map. When the graph is no longer used, it is deallocated from the VPUs.

For example, three video streams request neural network graph-1 for face detection. After calculation, two VPUs can process all face detection of the three clients. A tag and two VPUs are assigned in configuration map.

The Tag Schedule offers a deep learning computing capability that guaranteed to be flexibly configurable.

3.1 Use Cases

- You want stable resources (Intel® Movidius™ Myriad™ X VPU) for some networks.
- You want to control the deployment of networks among devices, without programming.

3.2 HAL Service Configuration File Configuration Settings

```
"graph_tag_map":{},  // defines the map b/w graph tag and
device number, like: {"tagDetect": 6, "tagAttr": 1, "tagLPR":
1}
```

3.3 Inference Engine API Configuration

- Function to set the configuration in the Inference Engine:

```
InferenceEngine::Core::LoadNetwork(CNNNetwork network, const
std::string & deviceName, const std::map<std::string,
std::string> & config)
```

- Tag Scheduler configuration:

```
const std::map<std::string, std::string> config =
{{VPU_HDDL_CONFIG_KEY(GRAPH_TAG), "tagDetect"}};
InferenceEngine::Core::LoadNetwork(CNNNetwork network,
‘‘HDDL’’, config);
```

See `inference_engine/demos/security_barrier_camera_demo` for more information.

3.4 Pro

An easy way to specify network deployment among devices by editing a configuration file. You don't need to use programming code.

3.5 Cons

- Only one network can be allocated to an Intel® Movidius™ Myriad™ X VPU.
- Deployments aren't changeable after the service starts.

3.6 Note

You can implement this scheduler's functionality by including the Bypass Scheduler in the application programming.

4.0 Stream Scheduler

The Stream Scheduler is used with some context awareness networks, such as RNN or LSTM, in which the successive frames in one video stream should be processed in an uninterrupted context.

The Stream Scheduler manages multiple VPUs and only accepts networks that attach a "stream id." One "stream id" is allocated to one VPU and all inference with this "stream id" is directed to the specified VPU for deep learning inference.

If two networks with have the same "stream id", they are allocated to two different VPUs. This might be useful when one stream or frame has to go through two different deep learning process in the application.

If all the VPUs it manages have been allocated a "stream id", then new networks don't have a VPU for it and are rejected. When the network is no longer used, it is deallocated from the VPUs.

With the Stream Scheduler, context awareness networks can be adopted.

4.1 Use Case

The network is context-aware.

4.2 HAL Service Configuration File Configuration Settings

```
 ''stream_device_number'': 0 (default value) // Specify the
number of Intel® Movidius™ Myriad™ X VPU for Stream Scheduler.
```

4.3 Inference Engine API Configuration

```
const std::map<std::string, std::string> config =
{{ VPU_HDDL_CONFIG_KEY(STREAM_ID), "sid"}};
InferenceEngine::Core::LoadNetwork(CNNNetwork network,
''HDDL'', config);
```

See the Tag Scheduler for more information.

4.4 Pros

Suitable for context-aware networks.

4.5 Con

Only one network can be allocated to an Intel® Movidius™ Myriad™ X VPU.

4.6 Note

The functionality of this scheduler can be implemented by Bypass Scheduler with programming in application.

5.0 SGAD Scheduler

Although the ByPass Scheduler provides flexibility, it requires extra effort to schedule networks on all devices. Repetitive operations are required to deploy the identical network on all devices. The SGAD Scheduler addresses this use case.

The SGAD Scheduler manages all VPUs assigned to this scheduler. It accepts all kinds of neural network graph and will allocate all graphs to every VPU it manages. When the graph is no more used, it will be deallocated from all VPUs.

With "SGAD scheduler", the need to load networks on multiple devices at once is met.

5.1 Use Cases

User want to load networks on all devices.

5.2 HAL Service Configuration File Configuration Settings

```
''sgad_device_number'': 0 (default value)     // Specify the
number of Intel® Movidius™ Myriad™ X VPU for SGAD Scheduler.
```

5.3 Inference Engine API Configuration

```
const std::map<std::string, std::string> config =
{{ VPU_HDDL_CONFIG_KEY(USE_SGAD), "yes"}};
InferenceEngine::Core::LoadNetwork(CNNNetwork network,
''HDDL'', config);
```

See the Tag Scheduler for more information.

5.4 Pros

- No need to repeatedly load same networks on all devices.
- Users can allocate any number of networks on a Myriad X as long as the memory of the device can hold these networks.

5.5 Cons

- Memory consumption is high because all networks are loaded onto all devices.
- There isn't a direct API in Inference Engine to acquire the number of Intel® Movidius™ Myriad™ X VPUs that the SGAD Scheduler is managing. Workarounds:
  - Use "`sgad_device_number`" in `hddl_service.config` to parse the configuration for the number.

- Hard-code the number in in application. This number must match `sgad_device_number”` in `hddl_service.config`.
- When all SGAD devices are used and an application tries to load a network with a new device tag, an error is returned. The application can use this to detect the number of devices in the SGAD Scheduler.

5.6 Note

You can implement the functionality of this scheduler by using the Bypass Scheduler and including code in your application.

6.0 ByPass Scheduler

The ByPass Scheduler creates a scheduling policy and is useful to manage the cases that Squeeze Scheduler cannot handle. The Bypass Scheduler exposes all Intel® Movidius™ Myriad™ X VPU devices that it manages.

6.1 Use Case

You want perform your own scheduling tasks.

6.2 HAL Service Configuration File Configuration Settings

```
 ‘‘bypass_device_number’’: 0 (default value) // Specify the
number of Intel® Movidius™ Myriad™ X VPU for Bypass Scheduler.
```

6.3 Inference Engine API Configuration

```
const std::map<std::string, std::string> config =
{{VPU_HDDL_CONFIG_KEY(DEVICE_TAG), "deviceTag"},
{VPU_HDDL_CONFIG_KEY(BIND_DEVICE), CONFIG_VALUE(YES)},
{VPU_HDDL_CONFIG_KEY(RUNTIME_PRIORITY), ‘‘1’’}};
InferenceEngine::Core::LoadNetwork(CNNNetwork network,
‘‘HDDL’’, config);
```

See the Tag Scheduler for more information.

6.4 Pros

- You can create any scheduling policy to satisfy your use case.
- You can allocate any number of networks on an Intel® Movidius™ Myriad™ X VPU as long as the memory of the device can hold these networks.

6.5 Cons

- You must develop the scheduling policy.
- There isn't a direct API in Inference Engine to acquire the number of Intel® Movidius™ Myriad™ X VPUs that the "Bypass Scheduler" is managing. When all Bypass devices are used and an application tries to load a network with a new device tag, an error returns. Applications can use this feature to detect the number of device in the Bypass Scheduler. Workarounds are:
  - Use "`bypass_device_number`" in `hddl_service.config` to configure the number of Intel® Movidius™ Myriad™ X VPU allocated to the Bypass Scheduler.

- You can hard code the number in application, but it must match the number defined by the "`bypass_device_number`".

6.6 Note

A more detailed description of how Bypass Scheduler works with its various configurations is in the attached slides.

§