The OpenVX™ Kernel Import Extension

Editor: Radhakrishna Giduthuri, The Khronos OpenVX Working Group

Version 1.0 (Provisional), 8th March 2018

Table of Contents

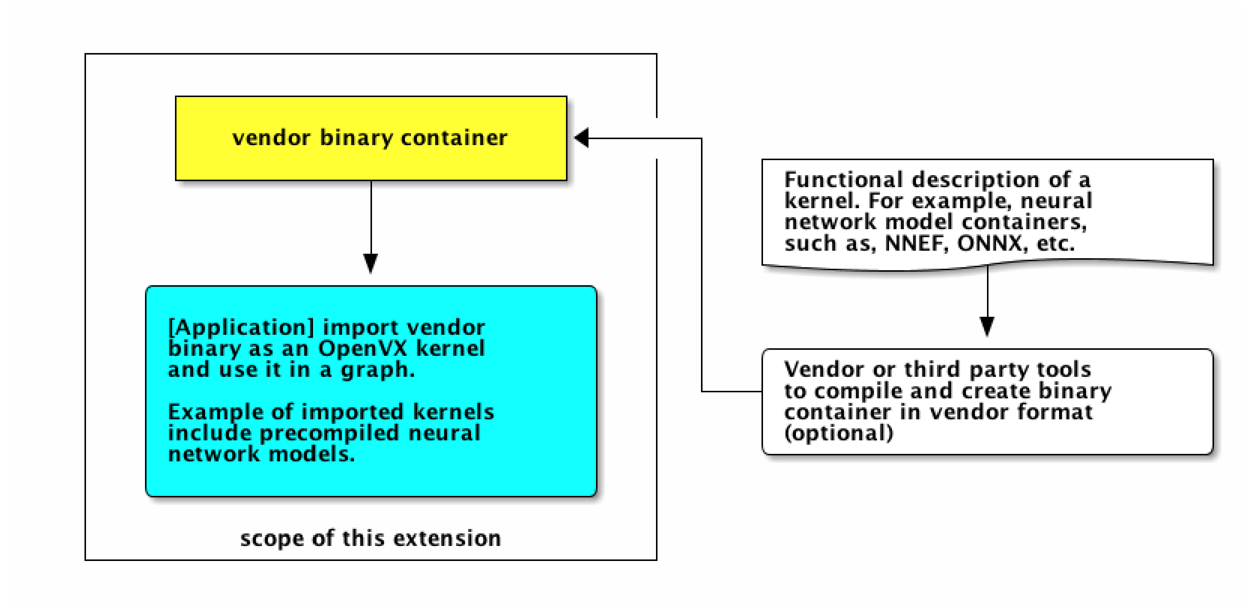
Technical Contributors

- Radhakrishna Giduthuri, AMD
- Niclas Danielsson, Axis Communications AB
- Frank Brill, Cadence
- Thierry Lepley, Cadence
- Adam Herr, Intel
- Jesse Villarreal, Texas Instruments

1. Purpose

This document details an extension to OpenVX 1.2, and references some APIs and symbols that may be found in that API, at https://www.khronos.org/registry/OpenVX/specs/1.2/html/index.html.

Provide a way of importing an OpenVX kernel from a vendor binary specified by URL.


vendor binary container
[Application] import vendor binary as an OpenVX kernel and use it in a graph.
Example of imported kernels include precompiled neural network models.
scope of this extension
Functional description of a kernel. For example, neural network model containers, such as, NNEF, ONNX, etc.
Vendor or third party tools to compile and create binary container in vendor format (optional)

The name of this extension is **vx_khr_import_kernel**.

1.1. Example: AlexNet graph

In order to use a neural network in OpenVX graph, one may to use the process outlined below:

- Import a pre-trained neural network kernel into the context from a vendor binary specified by URL. Use the `vxImportKernelFromURL` API to import the neural network kernel.
- Create an OpenVX graph that will use the imported neural network kernel.
- Create tensor objects for all neural network parameters (i.e., both input and output)
- Instantiate a neural network node into the graph using the `vxCreateGenericNode` and `vxSetParameterByIndex` APIs.
- Use the `vxVerifyGraph` API to verify and optimize the graph.
- Run the OpenVX graph in a loop

```
#include <VX/vx_khr_import_kernel.h>

void AlexNet( vx_size batchSize )
{
    // create OpenVX context
    vx_context context = vxCreateContext();

    // import neural network kernel
    const char * type = "vx_xyz_folder"; // XYZ's kernel binary container
    const char * url = "/assets/alexnet/"; // folder with AlexNet binary
    vx_kernel nn_kernel = vxImportKernelFromURL(context, type, url);

    // create input and output tensor objects
    vx_size input_sizes[] = { 224, 224, 3, batchSize };
    vx_size output_sizes[] = { 1, 1, 1000, batchSize };
    vx_tensor input = vxCreateTensor(context, 4, input_sizes, VX_TYPE_UINT8, 0);
    vx_tensor output = vxCreateTensor(context, 4, output_sizes, VX_TYPE_INT16, 8);

    // create OpenVX graph
    vx_graph graph = vxCreateGraph(context);

    // add neural network instance as a node in the OpenVX graph
    vx_node node = vxCreateGenericNode(graph, nn_kernel);
    vxSetParameterByIndex(node, 0, input);
    vxSetParameterByIndex(node, 1, output);
    vxReleaseNode(&node);

    // verify graph
    vxVerifyGraph(graph);

    // process graph with one batch at a time
    while( userGetNextJobInput(input) == VX_SUCCESS )
    {
        // execute the graph to run AlexNet
        vxProcessGraph(graph);

        // consume the output from AlexNet
        userConsumeOutput(output);
    }

    vxReleaseGraph(&graph);
    vxReleaseTensor(&input);
    vxReleaseTensor(&output);
    vxReleaseContext(&context);
}
```

2. The Kernel Import API functions

2.1. vxImportKernelFromURL

Import a kernel from binary specified by URL.

Declaration in VX/vx_khr_import_kernel.h

```
vx_kernel vxImportKernelFromURL(
    vx_context context,
    const vx_char * type,
    const vx_char * url
  );
```

Parameters

context

[in] OpenVX context

type

[in] Vendor-specific identifier that indicates to the implementation how to interpret the **url**. For example, if an implementation can interpret the **url** as a *file*, a *folder* a *symbolic label*, or a *pointer*, then a vendor may choose to use `"vx_<vendor>_file"`, `"vx_<vendor>_folder"`, `"vx_<vendor>_label"`, and `"vx_<vendor>_pointer"`, respectively for this field. Container types starting with `"vx_khr_"` are reserved. Refer to vendor documentation for list of container types supported.

url

[in] URL to binary container.

Return Value

- On success, a valid `vx_kernel` object. Calling `vxGetStatus` with the return value as a parameter will return VX_SUCCESS if the function was successful.

An implementation may provide several different error codes to give useful diagnostic information in the event of failure to create the context.

Description

This function imports a kernel object from a URL.

The name of kernel parameters can be queried using the `vxQueryReference` API with `vx_parameter` as **ref** and `VX_REFERENCE_NAME` as **attribute**.

Index