Inference with OpenVX™

1:00 – 1:30 PM

Mike Schmit
Director of Software Engineering, Computer Vision

AMD EPYC

OpenVX™

KHORONOS GROUP
AMD & Khronos History at EVS

- 2016: First open source OpenVX™ 1.0.1 implementation
- 2017: Radeon Loom 360 video stitching library for OpenVX (open source)
- 2018: Neural Net extensions to OpenVX (open source)
- 2019: AMD ROCm MIVisionX toolkit for computer vision (open source)

Tutorial links:
- https://github.com/kiritigowda/MIVisionX-Inference-Tutorial
- https://github.com/rrawther/MIVisionX-OpenVX-Tutorial
MIVisionX = OpenVXTM with tools/libraries

**MIVisionX**

- Conformant OpenVXTM 1.0.1, Open source (MIT license)
- Neural net extensions w/ Optimized MIOpen libraries
- Model compiler / model optimizer
- OpenCV™ interop
- Radeon Loom 360 stitching library
- WinML for Windows
- Utilities
  - ADAT (AMD Dataset Analysis Tool)
  - RunVX (command line OpenVX interpreter)
  - GDF (OpenVX scripting language & debugger)
  - LoomShell (360 image scripting language & debugger)
MIVisionX toolkit is a comprehensive computer vision and machine intelligence libraries, utilities and applications bundled into a single toolkit.

AMD OpenVX is delivered as Open Source with MIVisionX

Primarily targeted at applications requiring a combination of machine learning inference and computer vision or image/video processing.

Includes a model compiler for converting and optimizing a pretrained model from existing formats such as Caffe, NNEF and ONNX to an OpenVX backend.

After compilation, MIVisionX generates an optimized library specific for a backend to run inferencing and vision pre- and post-processing modules.

It is beneficial to have lightweight and dedicated APIs optimized for AMD hardware for inference deployment as opposed to heavyweight frameworks.
NEURAL NETWORK DEPLOYMENT OPTIONS

Frameworks

- TensorFlow
- Caffe2
- mxnet
- CNTK

ONNX

- NNEF

MIVisionX Model Compiler / optimizer

Application

- OpenVX run-time & libraries
  Deployment Option #1

- OpenVX Binary run-time & libraries
  Deployment Option #2

- WinML run-time & libraries
  Deployment Option #3

Application

- Future target system(s)
  Deployment Option #4

Training

Network
ROCM = Radeon Open Compute platform
HIP = Heterogeneous-compute Interface for Portability
Tutorial Examples

- Tutorial #1: Image Classification with ONNX
- Tutorial #2: Object Detection with Caffe
- Tutorial #3: Image Classification with NNEF
- Tutorial #4: Object Detection with multi-stream HW video decode

Not all Tutorials may be presented based on time available

Links:

https://github.com/kiritigowda/MIVisionX-Inference-Tutorial#mivisionx-inference-tutorial
https://github.com/rrawther/MIVisionX-OpenVX-Tutorial
See printed instructions to get connected now
Tutorial Example 1: Image Classification

- Using pretrained ONNX model
Tutorial Example 2: Object Detection

- Using Pre-Trained Caffe model
Tutorial Example 3: Image Classification

- Using Pre-Trained NNEF model
Example shows decoding 4 video streams simultaneously using amd_media_decoder OpenVX node and running the inference on 4 streams and visualizing the results using OpenCV.
## Bytes Processed (Per 1000 Images)

<table>
<thead>
<tr>
<th>CLIENT: READ HDD</th>
<th>CLIENT: XMIT</th>
<th>SERVER: JPEG DECODE</th>
<th>COPY: PCIe TO GPU</th>
<th>GPU: INFERENCEn</th>
<th>SERVER: SEND RESULTS</th>
<th>CLIENT: DISPLAY RESULTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 MB/sec</td>
<td>15 MB/sec</td>
<td>150 MB/sec &amp; 600 MB/sec (best case w/ no resize)</td>
<td>600 MB/sec</td>
<td>1000 images</td>
<td>1000 * 64</td>
<td>partial results shown; Full results reported</td>
</tr>
</tbody>
</table>

### Example Capacities:

**Examples**

- HDD = 100-200 MB/sec
- SATA III SSD = 550 MB/sec
- NVMe = ~2GB/sec

<table>
<thead>
<tr>
<th>Examples</th>
<th>Client: Read HDD</th>
<th>Client: XMIT</th>
<th>Server: JPEG Decode</th>
<th>Copy: PCIe To GPU</th>
<th>GPU: Inference</th>
<th>Server: Send Results</th>
<th>Client: Display Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 Gbps (125 MB/sec)</td>
<td>32 cores</td>
<td>PCIe 3.0 16 GB/sec for x16</td>
<td>600 – 900 images/sec per GPU for Resnet-50 FP32</td>
<td>1 Gbps (125 MB/sec)</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100 Gbps</td>
<td>64 threads</td>
<td>16 GB/sec for x16</td>
<td>600 – 900 images/sec per GPU for Resnet-50 FP32</td>
<td>1 Gbps (125 MB/sec)</td>
<td>NA</td>
<td></td>
</tr>
</tbody>
</table>

**Example Capacities:**

- 1 Gbps (125 MB/sec)
- 32 cores
- 64 threads
- PCIe 3.0
- 16 GB/sec for x16
- 600 – 900 images/sec per GPU for Resnet-50 FP32
- 1 Gbps (125 MB/sec)
- 100 Gbps

**Notes:**

- Assume 10:1 compression.
## ResNet50 Overall Summary

<table>
<thead>
<tr>
<th></th>
<th>Images With Ground Truth</th>
<th>Images Without Ground Truth</th>
<th>Total Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top 5 Match</td>
<td>6307</td>
<td>1305</td>
<td>7680</td>
</tr>
<tr>
<td>Total Mismatch</td>
<td>1668</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accuracy on Top 5</td>
<td>79.08 %</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mismatch Percentage</td>
<td>20.92 %</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average Pass Confidence for Top 5</td>
<td>56.18 %</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average mismatch Confidence for Top 1</td>
<td>37.19 %</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Match Rank</th>
<th>1st Match</th>
<th>2nd Match</th>
<th>3rd Match</th>
<th>4th Match</th>
<th>5th Match</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4307</td>
<td>999</td>
<td>528</td>
<td>293</td>
<td>190</td>
</tr>
<tr>
<td></td>
<td>54.01 %</td>
<td>12.53 %</td>
<td>6.62 %</td>
<td>3.56 %</td>
<td>2.38 %</td>
</tr>
</tbody>
</table>