cudaNvSciNvMedia Workflow

The sample demonstrates CUDA-NvMedia interop with/without NvSciBuf/NvSciSync APIs. This app takes an RGBA image as input and converts it into YUV 420 with help of NvMedia2DBlitEx(), subsequently this YUV output is converted to grayscale image by CUDA kernel. This is done twice once with NvSci allocators/syncs and another without NvSci allocators/syncs.

*Workflow **with** NvSci allocators/syncs*
Start Timer
-- NvMedia/NvSci* Allocation & CUDA import:

- setupNvMediaSignalerNvSciSync() → Creates NvSciSyncObj sync object via NvSci APIs where NvMedia is the Signaler and CUDA is the waiter.
- setupCudaSignalerNvSciSync() → Creates NvSciSyncObj sync object via NvSci APIs where NvMedia is the Waiter and CUDA is the Signaler.
- setupNvMedia() → Allocates NvMedia image input/output buffers via NvSciBuf using NvMediaImageCreateUsingNvScibuf() API here we also pass GPU id of the GPU from where CUDA is going to access this buffer.
- setupCuda() → Imports the NvSciBuf output buffer allocated by the above NvMedia function through CUDA External Resource Interoperability API and imports both the NvSciSyncObj's for waiting & signaling. Extracts the cudaArray from the imported cudaMipMapArray, and creates a surface object over this cudaArray. And allocates cuda output buffer.

-- Operating on NvMedia input/output by NvMedia & CUDA iteratively for time measurement:
Start Timer
For N iterations

1. runNvMediaBlit2D()
    a. NvMedia thread takes a RGBA image as input and loads it into pre-allocated NvSciBuf srcImage.
    b. if it is the second call to this function then performs a wait via NvMedia2DInsertPreNvSciSyncFence() for cuda to signal NvSciSyncObj.
    c. does NvMedia2DSetNvSciSyncObjforEOF()
    d. Then executes 2D Blit operation NvMedia2DBlitEx() on the src NvMedia image converting the RGBA image into YUV 420.
    e. Finally, signals the b.) NvSciSync semaphore via NvMedia2DGetEOFNvSciSyncFence().
2. runCudaOperation()
    a. waits on imported NvSciSync semaphore to get signaled by NvMedia in the cuda stream via cudaWaitExternalSemaphoresAsync().
    b. CUDA kernel is enqueued in the cuda stream, this kernel converts the pre-imported YUV420 NvSciBuf buffer into grayscale by only writing the "Y" (luma) values into the separate cuda allocated buffer.

c. Finally, it signals NvMedia via signalExternalSemaphore() for completion of cuda operation until the second last iteration as after the last iteration no subsequent NvMedia2DBlitEx() operation shall be launched. Hence, in the last iteration we call cudaStreamSynchronize() to block the host until the enqueued cuda operation is completed.

End Timer

-- Cleanup all NvMedia, NvSci* & CUDA allocated buffers/synchronization objects.

End Timer

*Workflow **without** NvSci allocators/sync*

Start Timer
-- NvMedia & CUDA allocations:

- setupNvMedia() :
  Allocation:
  srcImage: NvMedia Image Buffer allocated with NvMediaImageCreateNew().
  dstImage: NvMedia Image Buffer allocated with NvMediaImageCreateNew().
  dstBuff: Host side buffer allocated with malloc() to copy output YUV image.
- setupCuda() :
  Allocation:
  d_yuvArray: CUDA array to store the input image from NvMedia and run CUDA kernel.
  d_outputImage: CUDA buffer to store the output grayscale image generated by cuda kernel.

-- Operating on NvMedia input/output by NvMedia & CUDA iteratively for time measurement:
Start Timer
For N iterations:

1. runNvMediaBlit2D()
   a. performs 2D Blit operation NvMedia2DBlitEx() on the “srcImage” converting the RGBA image into YUV 420.
   b. Copies the NvMedia output buffer via NvMediaImageGetBits() call to the host buffer “dstBuff”.
2. runCudaOperation()
   a. Copies as input the ““dstBuff” output buffer of NvMedia processed YUV 420 image to “d_yuvArray”.
   b. CUDA kernel is launched which converts the imported YUV420 buffer into grayscale by only writing the “Y” values into the output “d_outputImage” buffer.

End Timer

-- Cleanup all NvMedia & CUDA allocated buffers as well as host buffers used for copying NvMedia output to CUDA array.
End Timer