GPU TECHNOLOGY CONFERENCE

NVIDIA

ストリームを用いた コンカレントカーネルプログラミングと最適化

エヌビディアジャパン　CUDAエンジニア 森野慎也

GTC Japan 2014

GPU TECHNOLOGY CONFERENCE

コンカレントな処理の実行

- システム内部の複数の処理を、平行に実行する。
  - CPU・GPU
  - メモリ転送・カーネル実行
  - 複数のカーネル間
- ストリーム
  - GPU上の処理キュー
  - カーネル実行・メモリ転送 の並列性・実行順序。

GPU TECHNOLOGY CONFERENCE

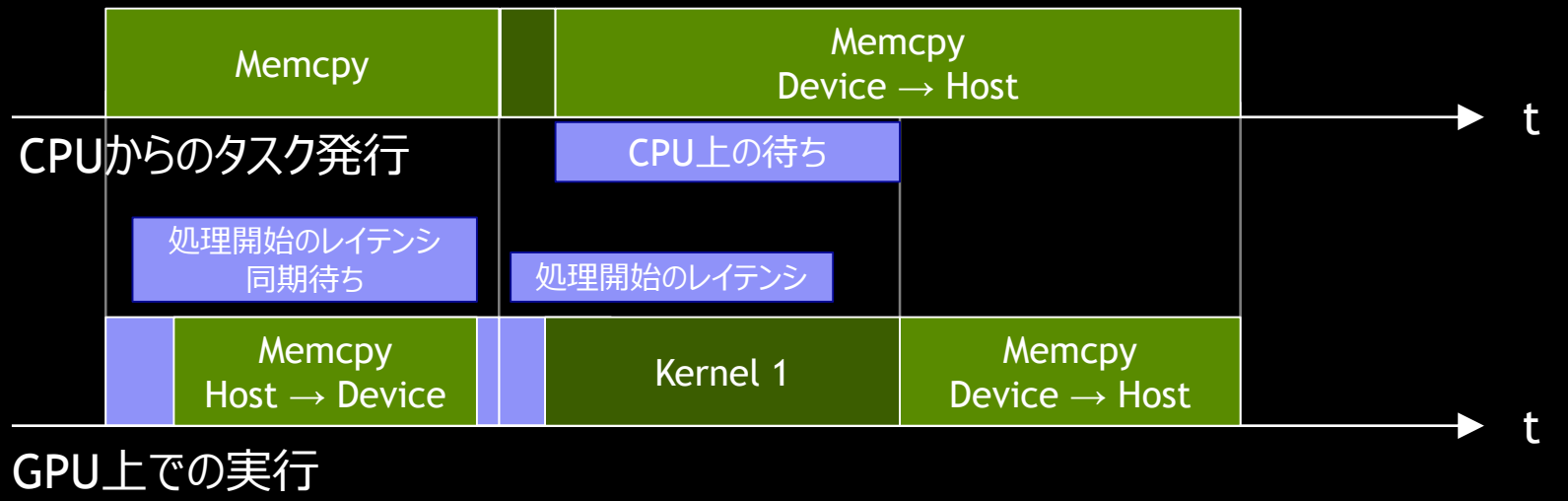
DEFAULT STREAM

- Stream：GPU上の処理を管理するキュー
- 無指定の場合、Default (NULL) Streamが使用される。

GPU TECHNOLOGY CONFERENCE

CPU/GPUのコンカレンシを考慮していない例

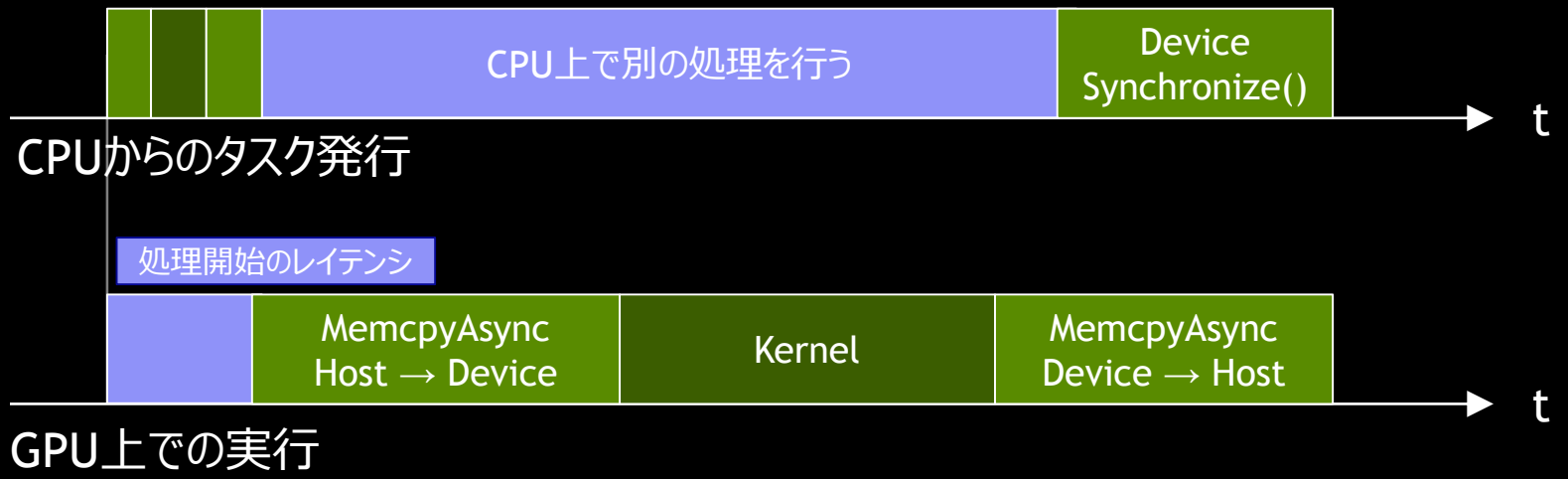
- cudaMemcpy()は、同期的に動作する。



GPU TECHNOLOGY CONFERENCE

CPU・GPU間のコンカレンシ

- 非同期コピー(cudaMemcpyAsync()) を使用。



GPU TECHNOLOGY CONFERENCE

同期版・非同期版のAPI

- memcpy、memset系には、同期・非同期バージョンがある。
- 基本的には、非同期版を使用。

同期版	非同期版
cudaMemcpy()	cudaMemcpyAsync()
cudaMemcpy2D()	cudaMemcpy2DAsync()
cudaMemcpyToSymbol()	cudaMemcpyToSymbolAsync()
cudaMemcpyFromSymbol()	cudaMemcpyFromSymbolAsync()
cudaMemset()	cudaMemsetAsync()

CUDA Runtime APIのリファレンスから抜粋

GPU TECHNOLOGY CONFERENCE

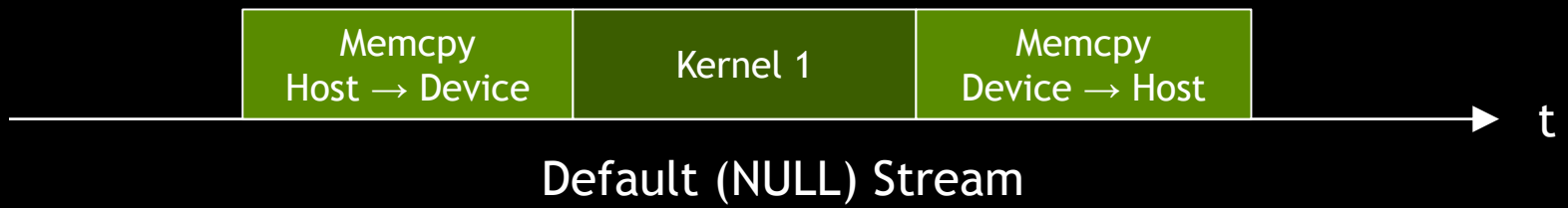
ブロックする処理

- cudaDeviceSynchronize()
  - 同期API

- 意図せずBlockする可能性があるのは…
  - メモリ確保・解放　cudaMalloc()、cudaFree()など。
    対応：あらかじめ確保しておく。
  - Pageable Memoryを使用したcudaMemcpy()系API
    対応：cudaHostAlloc()を使用して、Pinned Memoryをアロケート。

GPU TECHNOLOGY CONFERENCE

デモ：NSIGHT で タイムラインを見る



GPU TECHNOLOGY CONFERENCE

DRIVER QUEUE LATENCY

- ドライバ内部のキューの処理レイテンシ
- Windowsにおけるデバイスドライバのモード
  - WDDM Mode：ディスプレイドライバ
    - ミリ秒を超えるレイテンシが発生しやすい。
    - cudaStreamQuery(NULL) で、デバイスに処理を流し込む。
  - TCC Mode：Tesla Compute Cluster
    - レイテンシは、数十 μ秒

GPU TECHNOLOGY CONFERENCE

ストリームを使用したコンカレントプログラミング

GPU TECHNOLOGY CONFERENCE

ユーザが作成できるSTREAM

- Blocking Stream

  例：cudaStreamCreate(&stm);
  cudaStreamCreateWithFlags(&stm, cudaStreamDefault);

- Non-blocking Stream

  例：cudaStreamCreateWithFlags(&stm, cudaStreamNonBlocking);

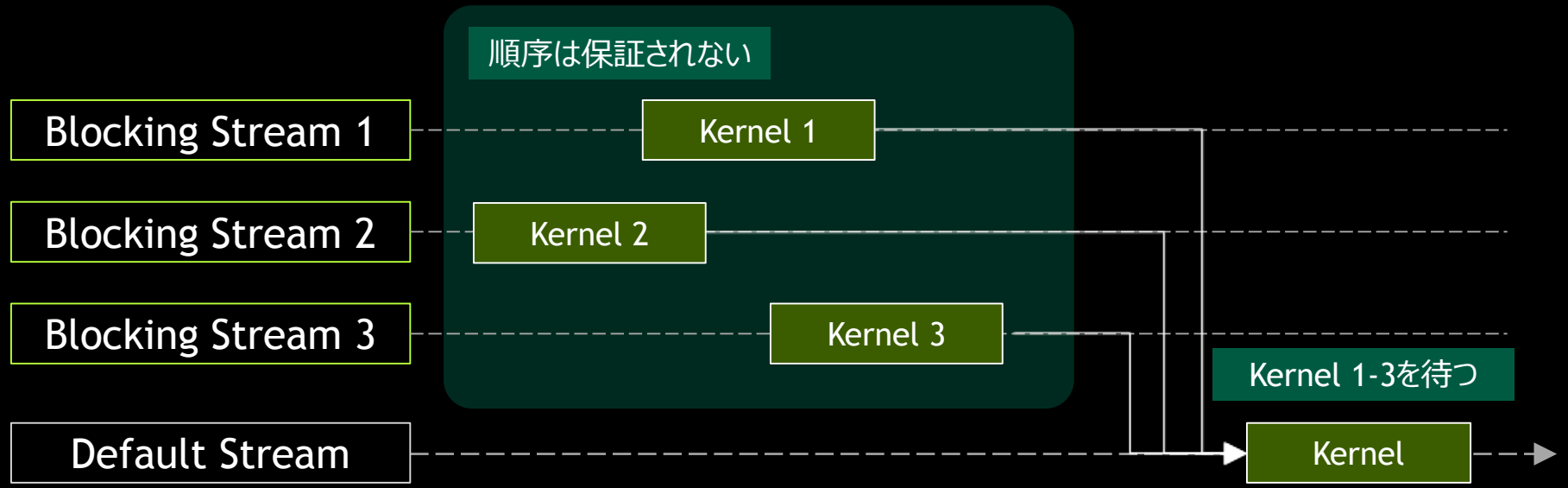
- Prioritized Stream (今日は説明しません)

  例：
  cudaStreamCreateWithPriority(&stm, cudaStreamDefault, priority);

GPU TECHNOLOGY CONFERENCE

BLOCKING STREAM

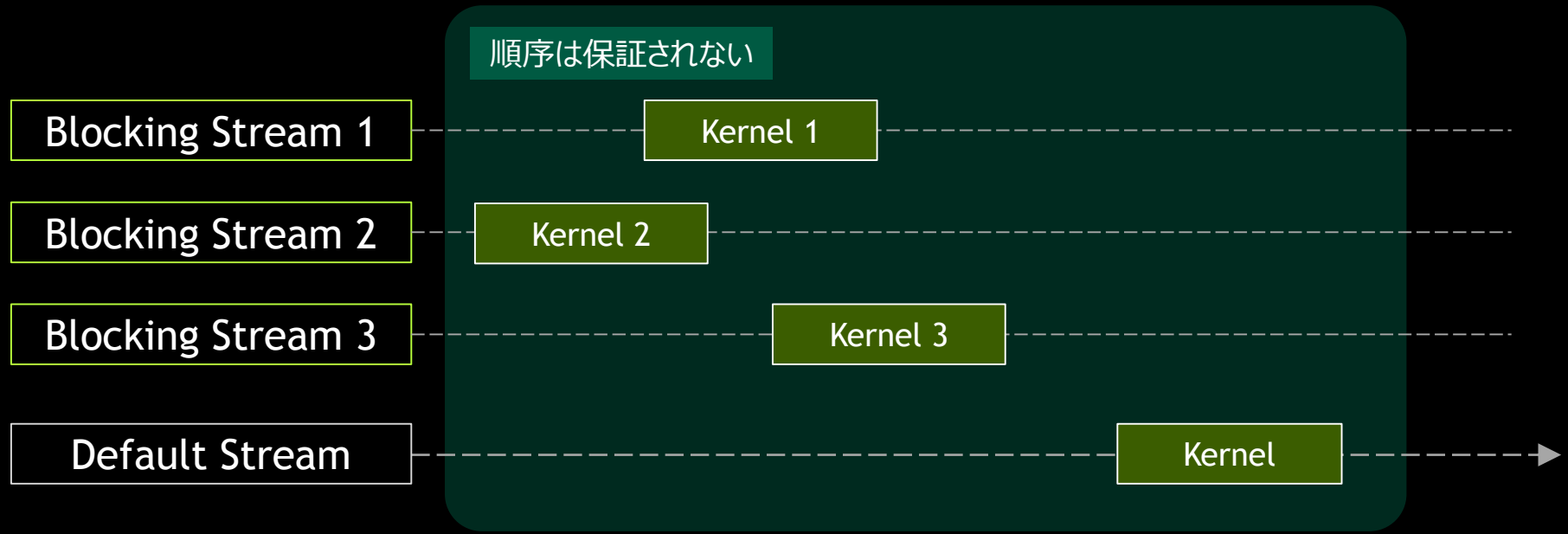
- Default Streamと同期する。
  - カーネル間でデータの依存性がある場合に有効。



GPU TECHNOLOGY CONFERENCE

NON-BLOCKING STREAM

- Default Streamに対しても非同期。



GPU TECHNOLOGY CONFERENCE

STREAMに対する同期プリミティブ

- 状態確認・同期・イベント同期

API	説明
cudaStreamQuery()	Stream上の処理が完了しているか確認
cudaStreamSynchronize()	Stream上の処理完了を確認。同期。
cudaStreamWaitEvent()	Stream上でEventを待つ。

GPU TECHNOLOGY CONFERENCE

今日のお題

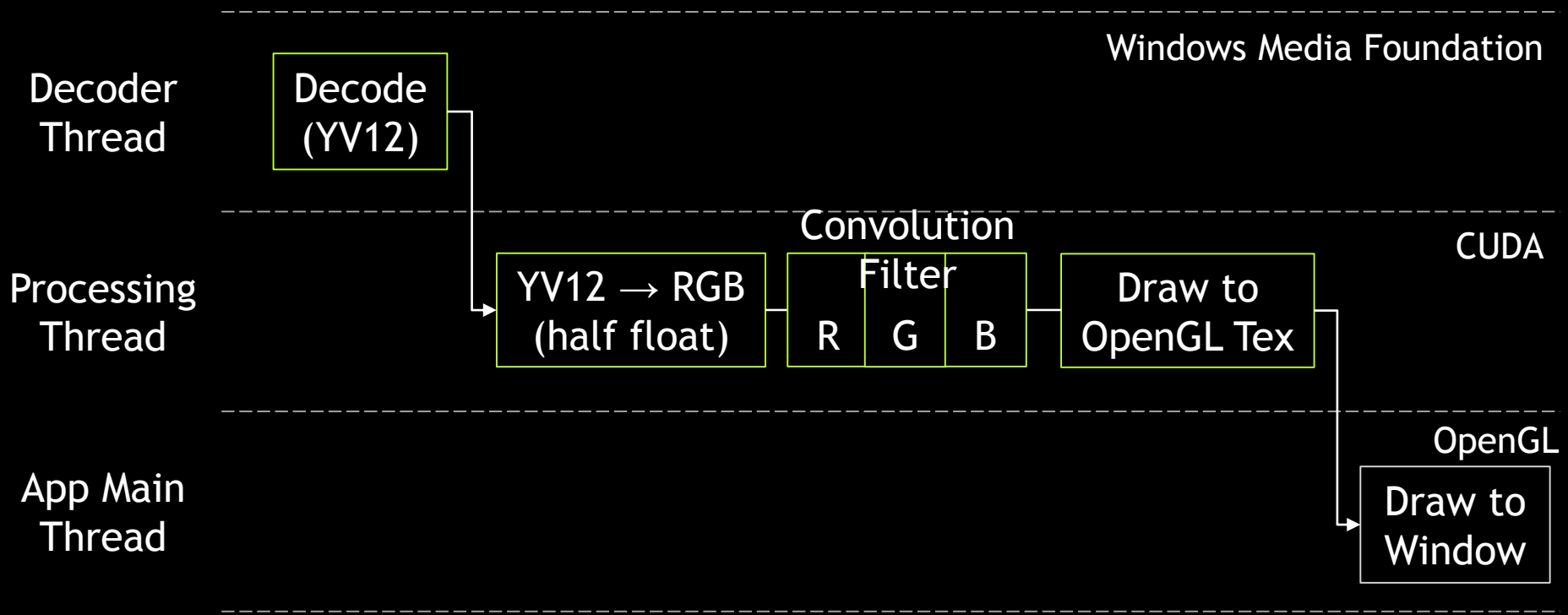
フィルタ付き動画プレーヤー

動かしてみる。

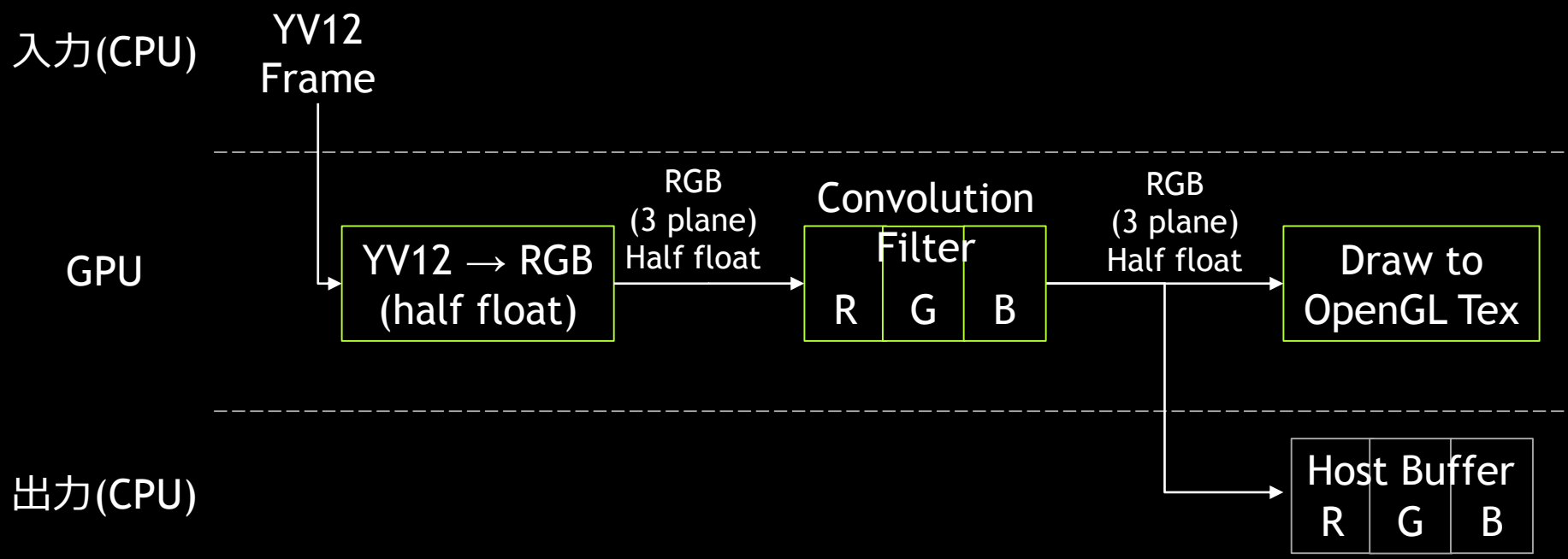
GPU TECHNOLOGY CONFERENCE

フィルタ付き動画プレーヤー

- 3 CPUスレッドでパイプラインを構成



データフロー (PROCESSING THREAD)

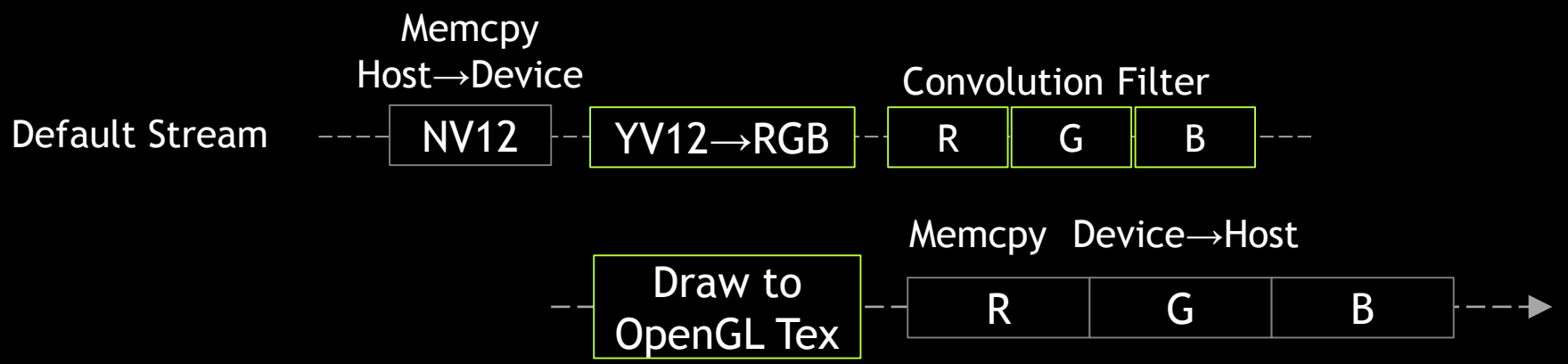


GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE

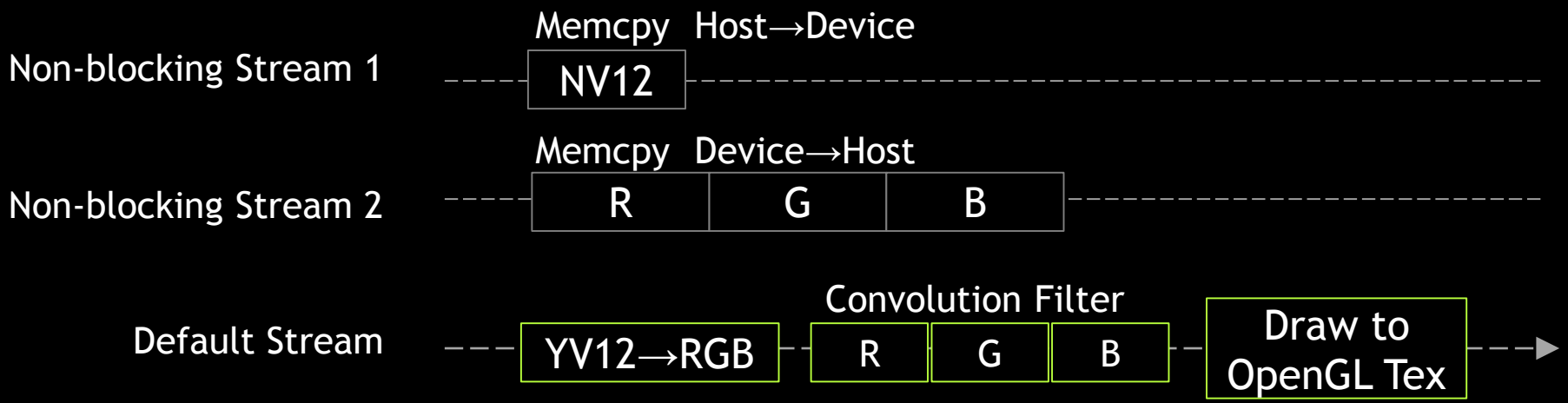
DEFAULT STREAMを使用した場合…

- 実行時間：4.17 ms
- データ転送：1.42 ms
- カーネル：2.43 ms



メモリ転送をコンカレントに実行(オーバーラップ)

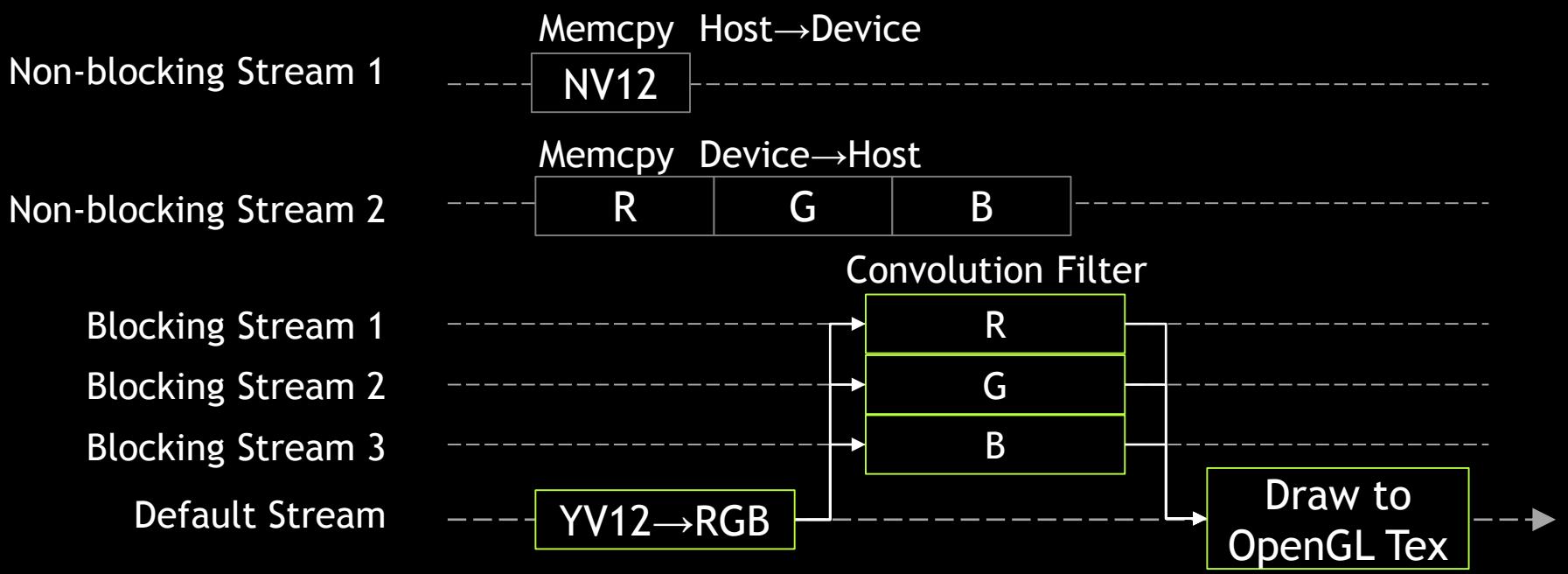
- 実行時間：2.85 ms
- データ転送：1.25 ms
- カーネル：2.42 ms



GPU TECHNOLOGY CONFERENCE

カーネルも並列実行

- 実行時間：2.78 ms, データ転送：1.15 ms, カーネル：2.37 ms



GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE

性能比較

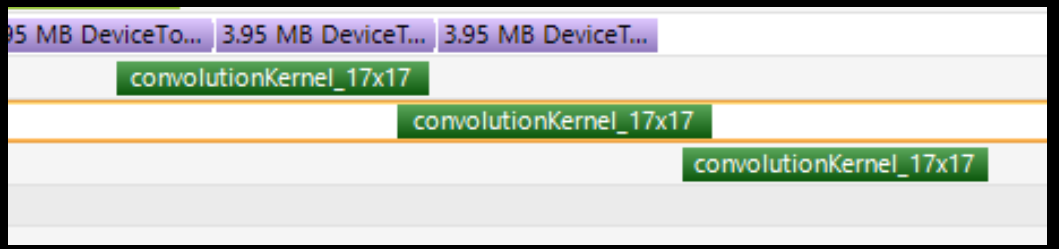
#	実装	処理時間	短縮分	メモリ転送時間	カーネル実行時間
1	Default Streamのみ	4.17 ms	—	1.42 ms	2.43 ms
2	メモリ転送をオーバーラップ	2.85 ms	1.32 ms	1.25 ms	2.42 ms
3	カーネル実行もオーバーラップ	2.78 ms	1.39 ms	1.15 ms	2.37 ms

- メモリ転送のオーバーラップ分、速くなった。
- カーネルのオーバーラップは、”今回は” ちょっとだけ効果あり。

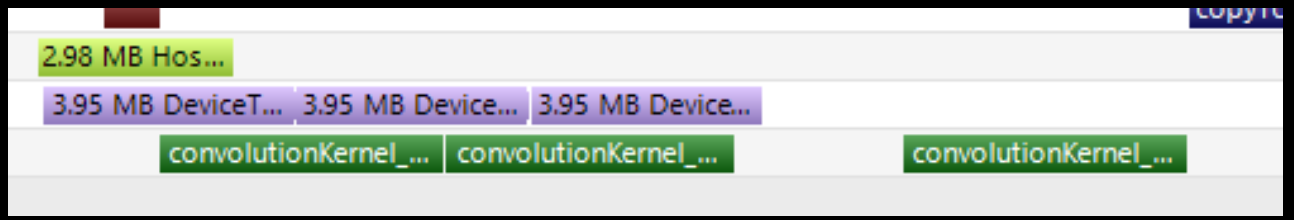
カーネル実行もオーバーラップする

- 別ストリームで実行
  - 「順序に依存しない」、「データの依存性がない」
  - 効果のある事例については、機会を改めて。(Hyper-Qも扱いたい)

オーバーラップしている



オーバーラップしていない



GPU TECHNOLOGY CONFERENCE

まとめ

1. CPU・GPU間のコンカレンシ。
   - 非同期APIの使用。
2. カーネルとメモリ転送のコンカレンシ
   - Dual Copy Engine
3. カーネル間のコンカレンシ
   - カーネル間のデータ依存性

- ストリーム
  - Default / Blocking / Non-blocking Stream