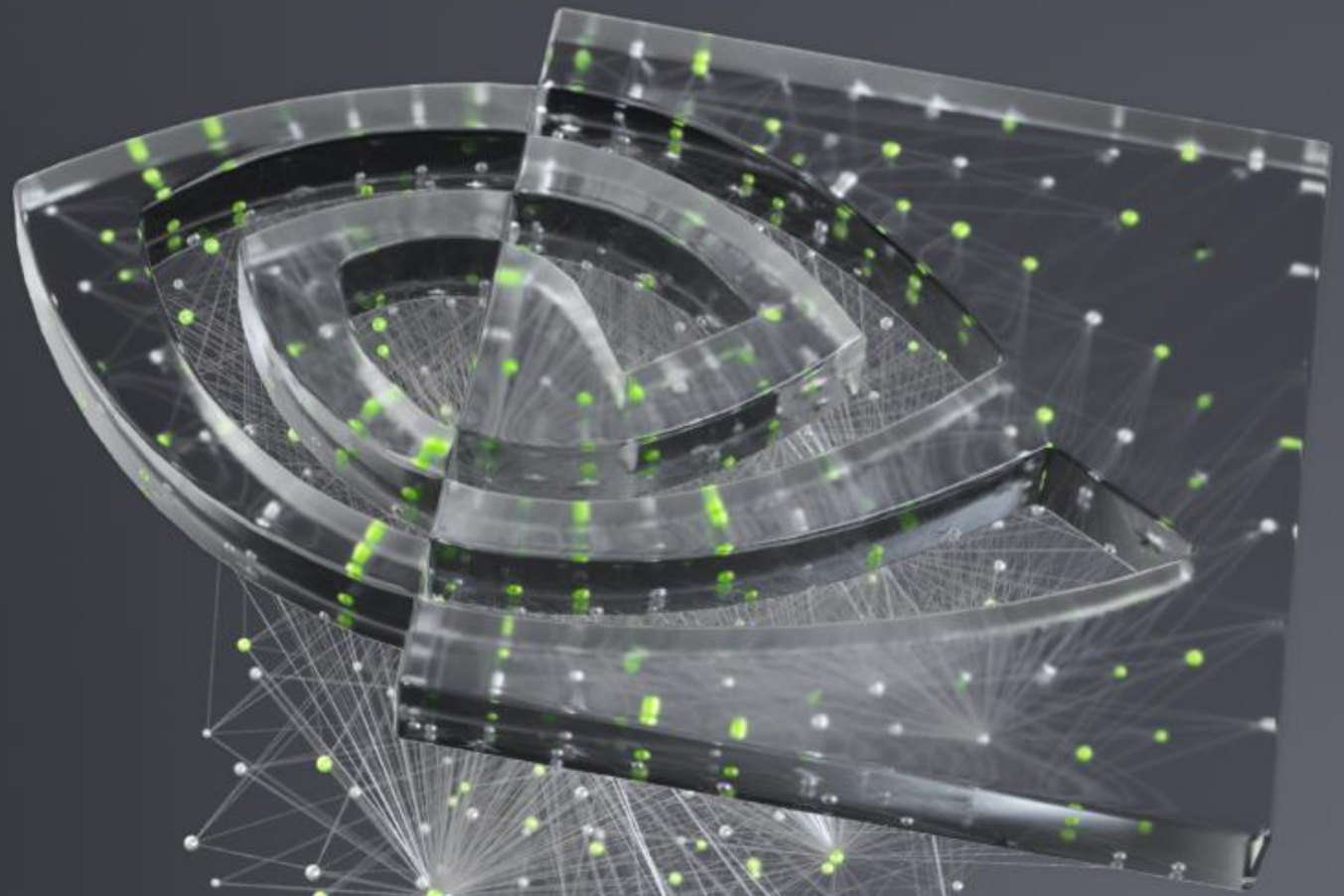


NVIDIA VIDEO TECHNOLOGIES

Abhijit Patait, GTC 2020

AGENDA

Video/Image Processing on NVIDIA GPUs

NVENC, NVDEC, NvOFA, NvJPEG

Software Updates and New Features

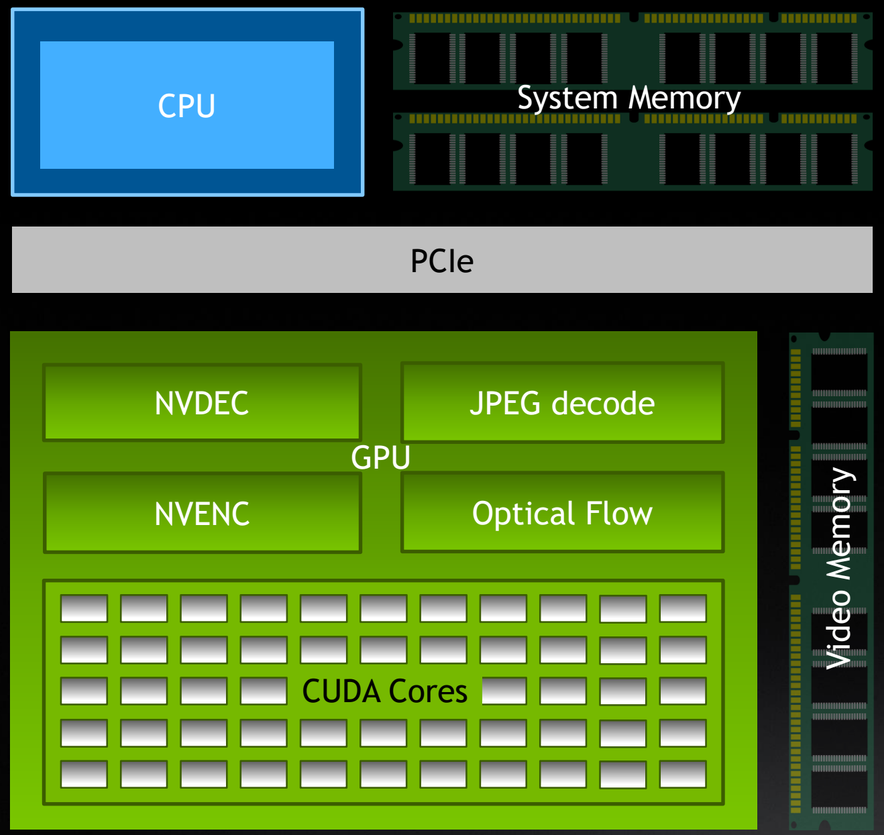
Updates, benchmarks and end-to-end use-cases

Roadmap

Upcoming features

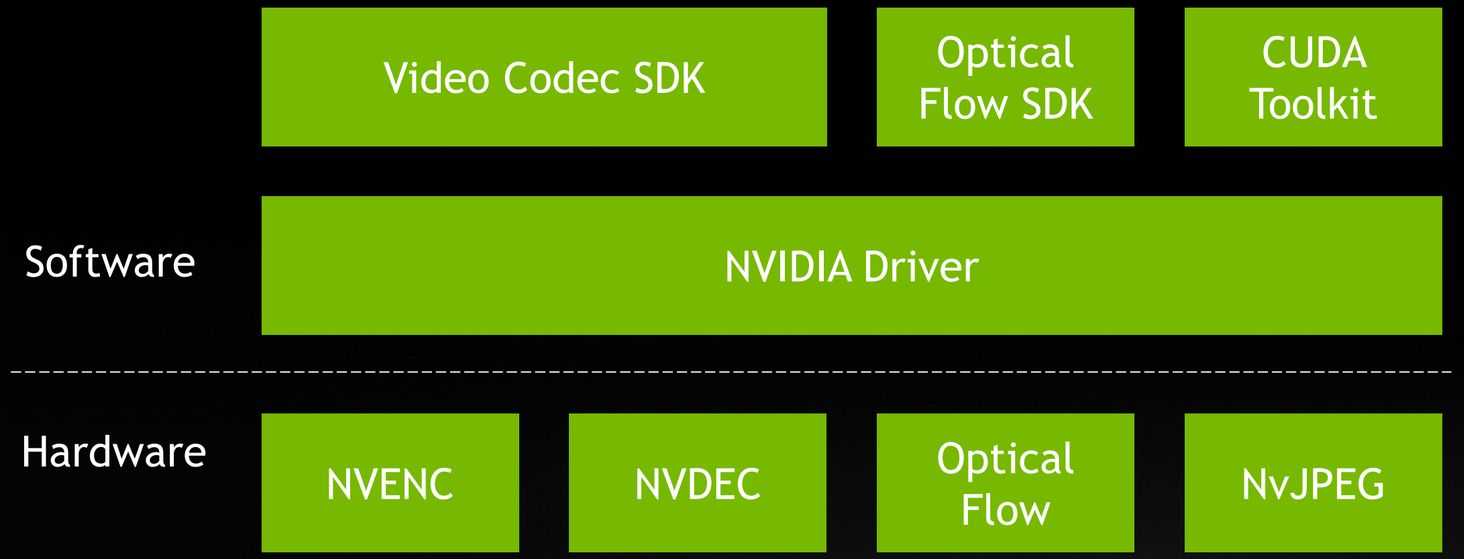
Video/Image Processing on NVIDIA GPUs

NVIDIA VIDEO/IMAGE HARDWARE



- CUDA Cores
- NVDEC – Video Decoder (H.264, H.265, VP9, MPEG-2…)
- NVENC – Video Encoder (H.264, H.265)
- Optical Flow (Turing+) – Track pixels
- JPEG Decoder (Ampere+ architecture)

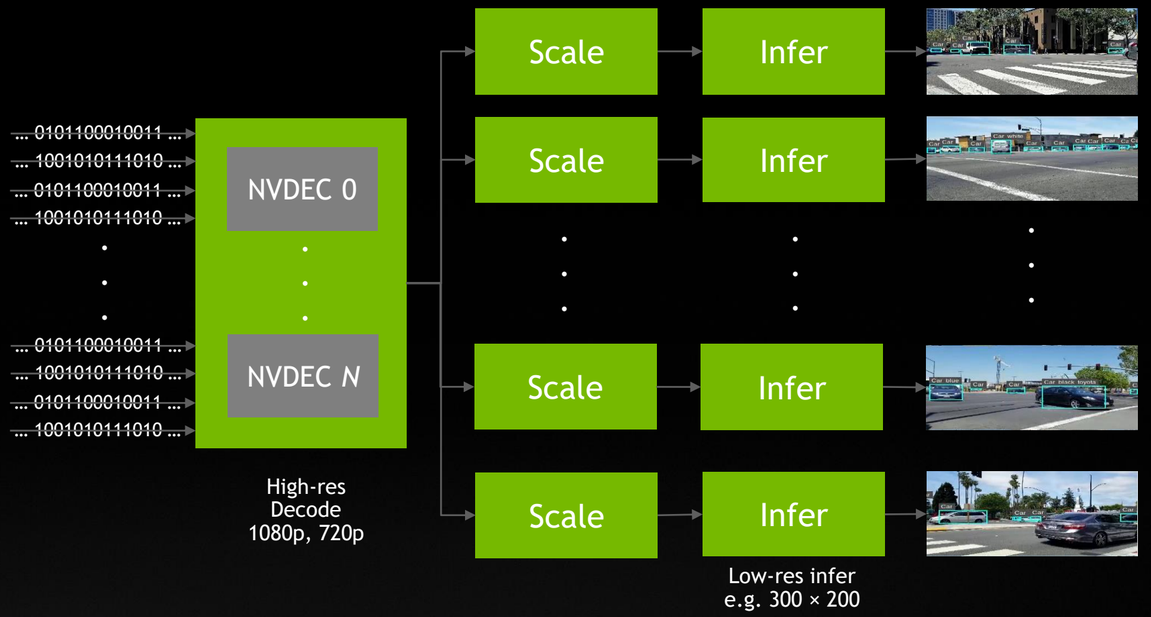
SOFTWARE



- All binaries in NVIDIA driver
- SDKs
  - APIs
  - Reusable samples
  - Documentation
- Linux & Windows
- CUDA, DirectX, OpenGL, Vulkan APIs
- Binary backward compatibility

WHAT'S NEW IN GA100 GPUs?

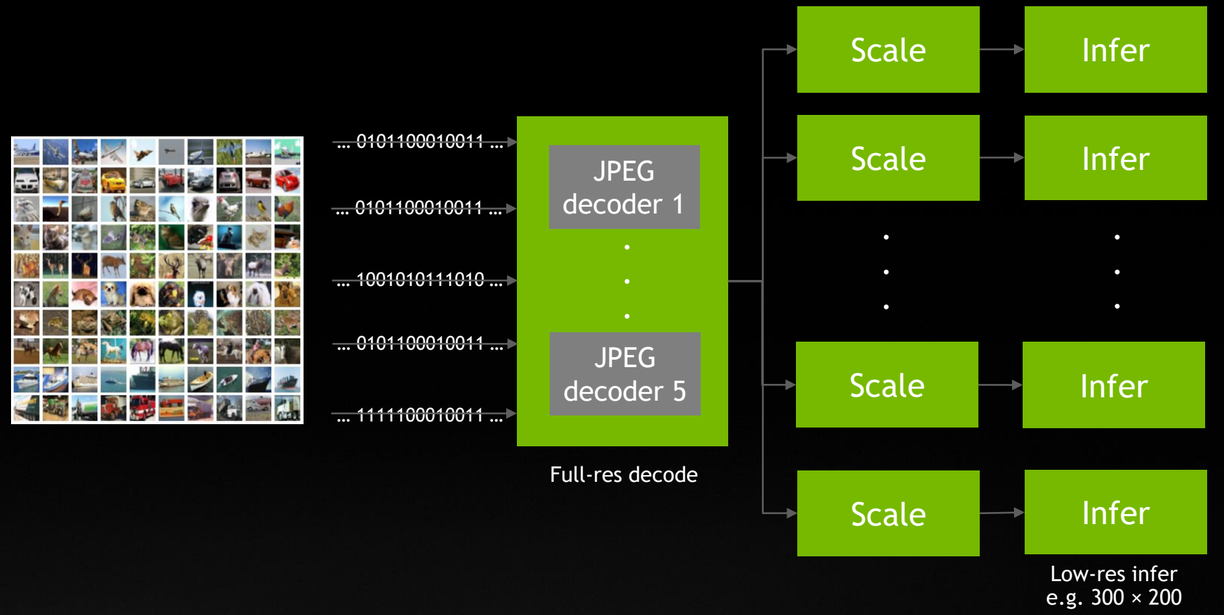
- NVDECs
  - Pascal: 1
  - Turing: 1-2
  - **GA100: 5**



Not all features will be in all GPUs. Please check NVIDIA developer zone web site for detailed information

WHAT'S NEW IN GA100 GPUs?

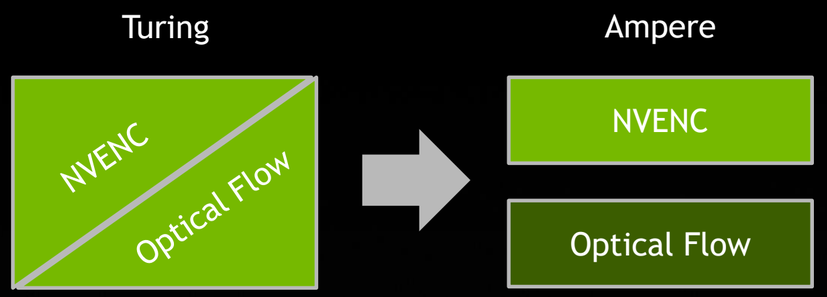
- NVDECs
  - Pascal: 1
  - Turing: 1-2
  - **GA100: 5**
- Dedicated 5-core JPEG decoder



Not all features will be in all GPUs. Please check NVIDIA developer zone web site for detailed information

WHAT'S NEW IN GA100 GPUs?

- NVDECs
  - Pascal: 1
  - Turing: 1-2
  - **GA100: 5**
- Dedicated 5-core JPEG decoder
- Improved optical flow engine, independent of NVENC



- Per-pixel flow vector
- Improved quality & perf
- 8192 × 8192
- Region of interest

Not all features will be in all GPUs. Please check NVIDIA developer zone web site for detailed information

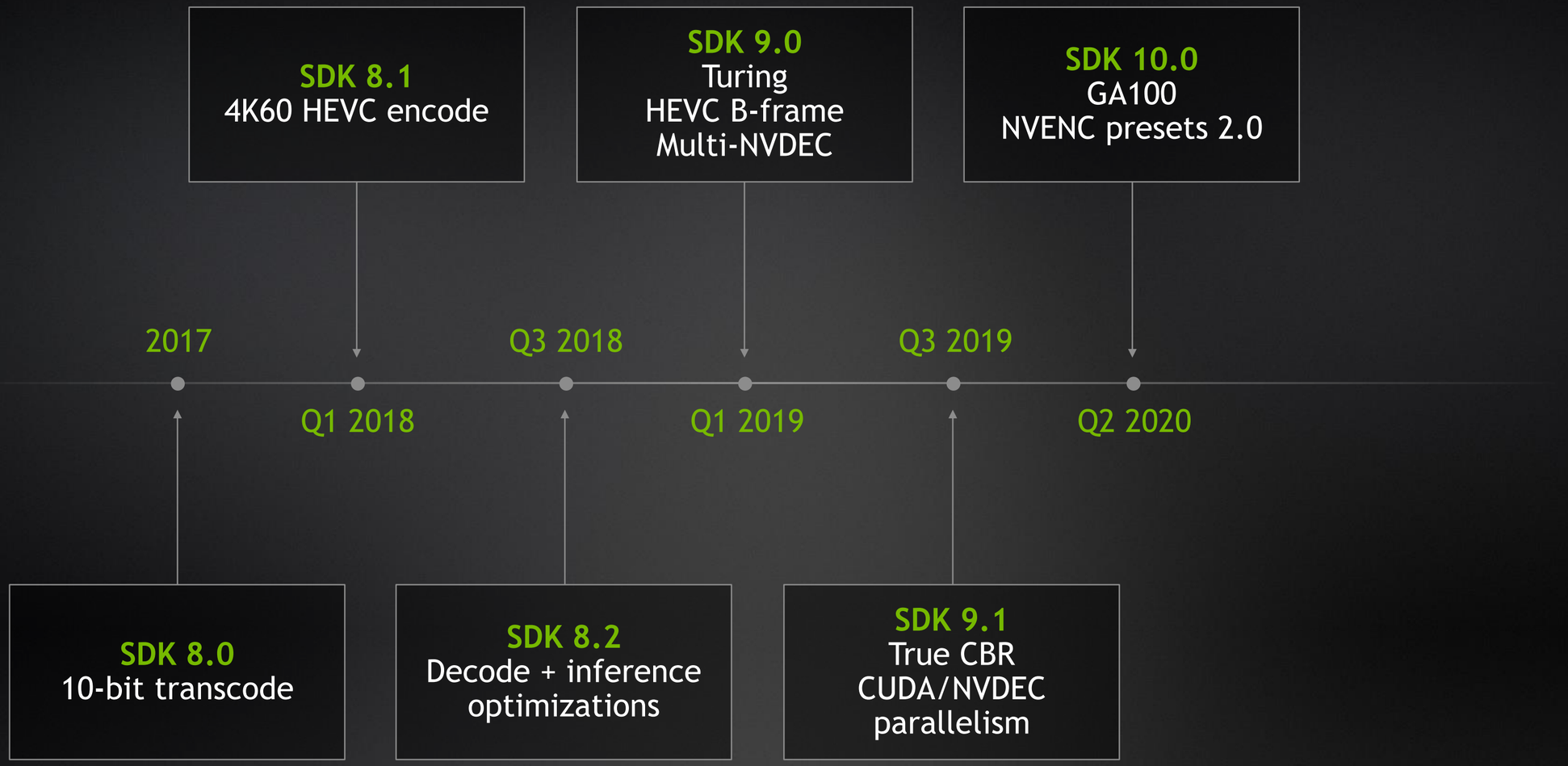
SOFTWARE UPDATES

Upcoming SDK Releases

- Video Codec SDK 10.0
- Optical Flow SDK 2.0
- NvJPEG decode (part of CUDA 11.0)

Video Codec SDK Update

VIDEO CODEC SDK



VIDEO SDK 10.0

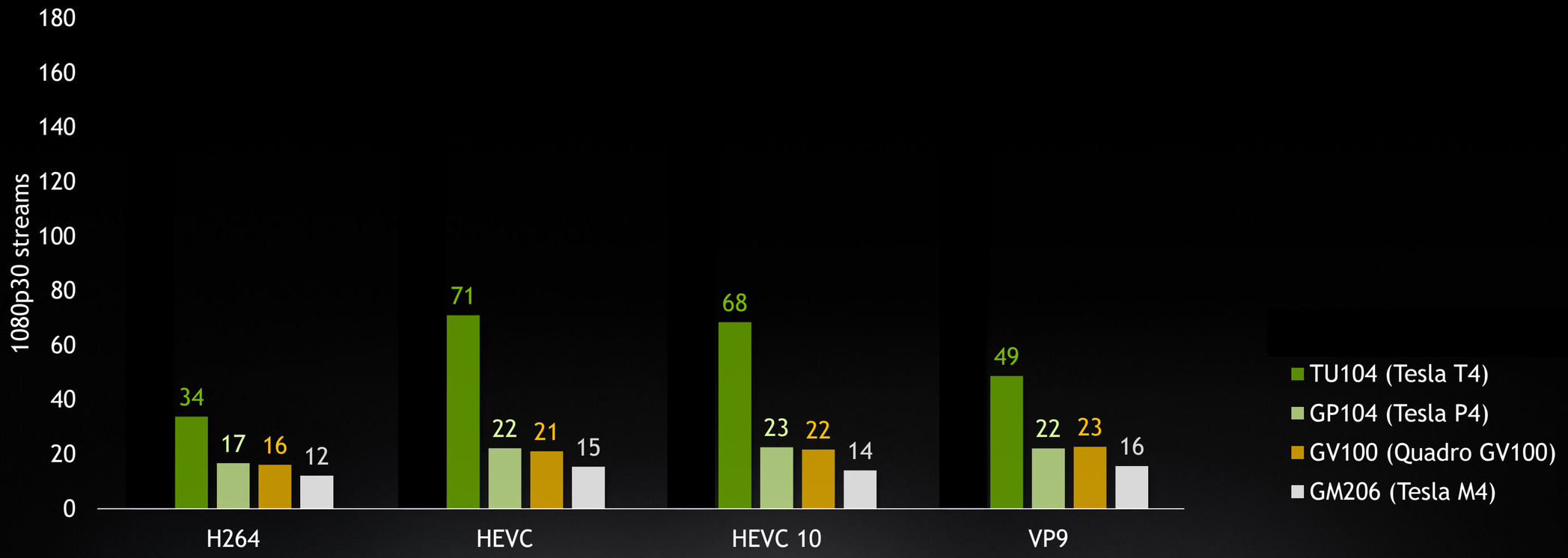
- GA100 Support (Ampere architecture)
  - Multi-NVDEC performance improvements
- Encoder (NVENC) presets 2.0
- June 2020

VIDEO SDK 10.0

- GA100 Support (Ampere architecture)
  - **Multi-NVDEC performance improvements**
- Encoder (NVENC) presets 2.0

TURING ARCHITECTURE

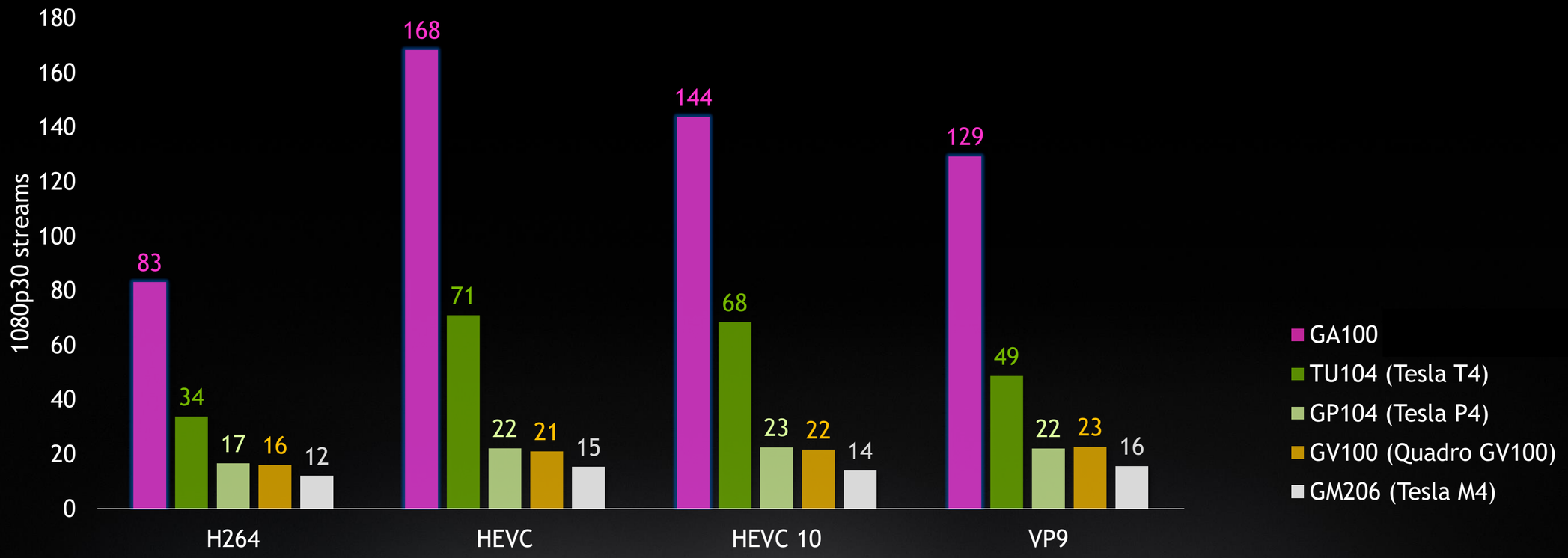
Up to 2 NVDECs per GPU for Decode + Inference



NVIDIA

GA100 WITH AMPERE ARCHITECTURE

5 NVDECs per GPU for Decode + Inference



VIDEO SDK 10.0

- GA100 (Ampere microarchitecture) Support
  - Multi-NVDEC
- **Encoder (NVENC) presets 2.0**

NVENC CONFIGURATION

Video Codec SDK 9.1 and earlier

- Choose from
  - **6 Presets:** HQ, DEFAULT, HP, LLHQ, LL_DEFAULT, LLHP
  - **6 Rate control modes:** Constant QP, VBR, CBR, CBR_LOWDELAY_HQ, CBR_HQ, VBR_HQ
  - **Advanced features:** B-frames, B-as-reference, Look-ahead, AQ, weighted prediction, VBV…
- Use-case specific configuration
  - Streaming: LL* + CBR_LOWDELAY_HQ + 1-frame VBV
  - Transcoding: HQ + VBR/CBR + B-frames + B-as-reference + AQ + high VBV
- Resolution-dependent quality/performance
- Extreme ends of quality/performance difficult to achieve

INTRODUCING NVENC PRESETS 2.0

Video Codec SDK 10.0

- Choose
  - **Use-case/Tuning Info:** High-quality, Low Latency, Ultra Low Latency, Lossless
  - **Preset:** P1 (highest performance) to P7 (highest quality)
  - **Rate Control Mode:** Constant QP, CBR, VBR
  - **Advanced Features:** Built-in; tune only if required

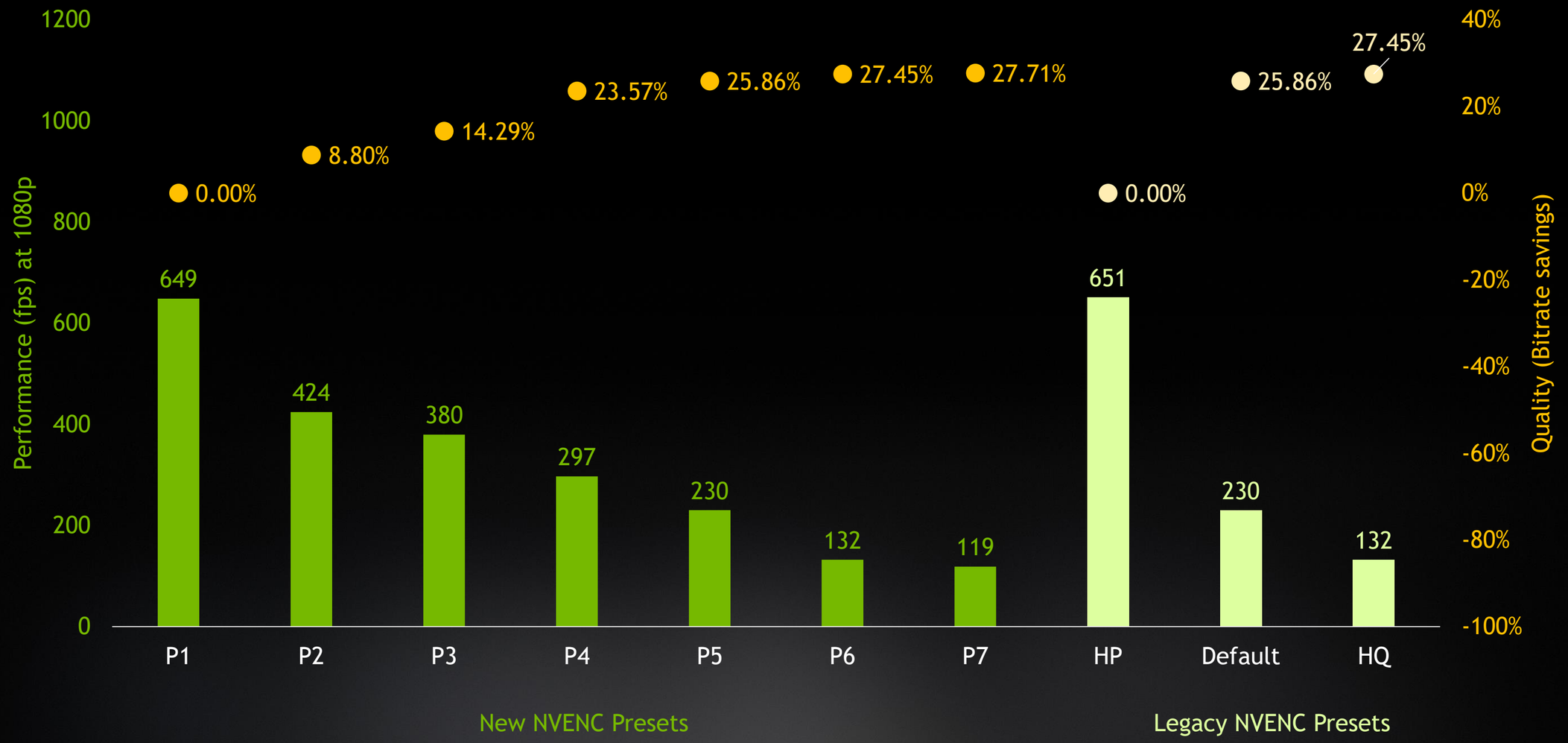
LEGACY VS NEW NVENC PRESETS

Comparison

	Legacy Presets	Presets 2.0
One-shot configuration	No	Yes
Advanced features config	Needed	Rarely needed
Use-case based	No	Yes
Performance scales with resolution	Sometimes	Always
Multi-pass rate control	Indirect	Direct
Fine granularity of perf vs quality	Low	High
Rate control modes	Complex	Easy to understand
Extreme ends of quality/perf trade-off	No	Yes

NVENC PRESETS: NEW VS LEGACY

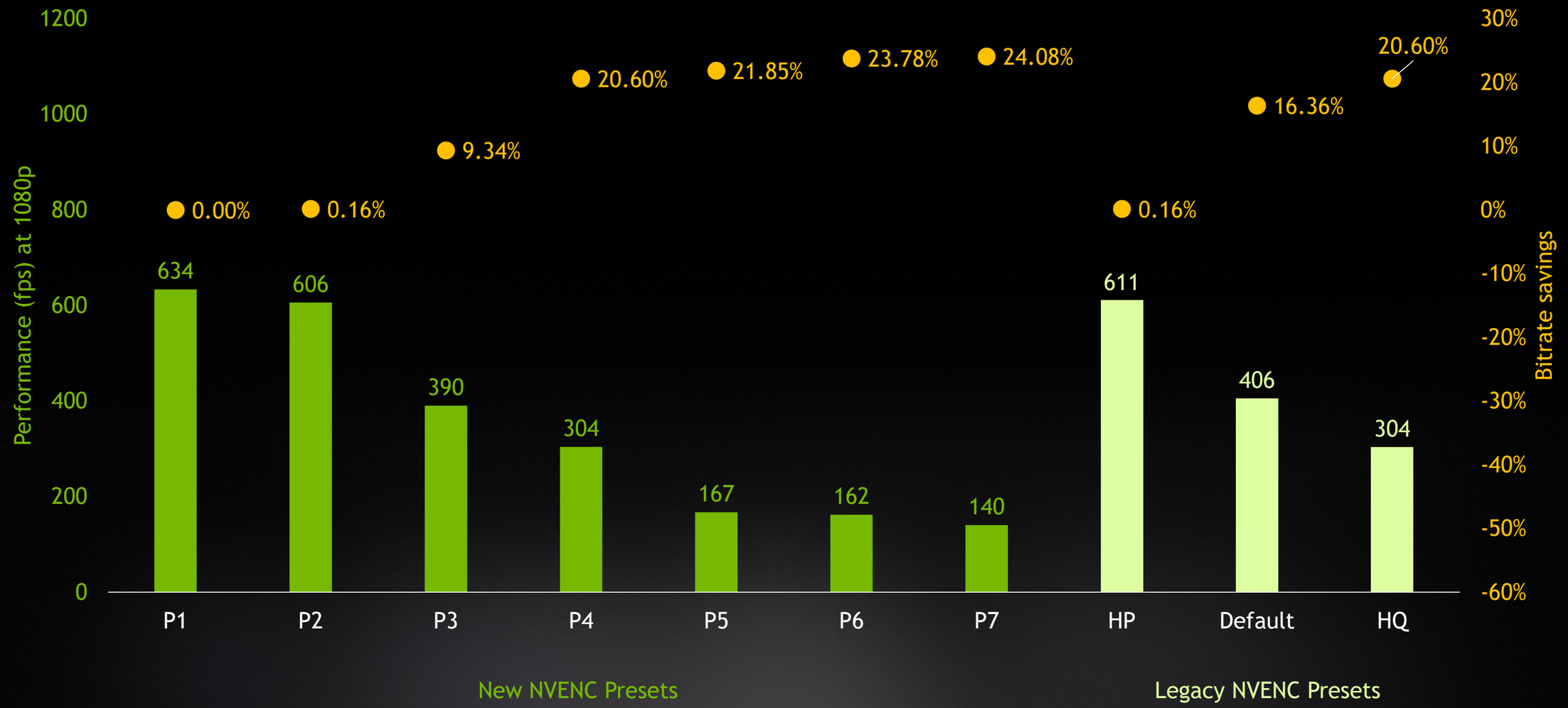
HEVC, High quality (latency tolerant), VBR



NVIDIA

NVENC PRESETS: NEW VS LEGACY

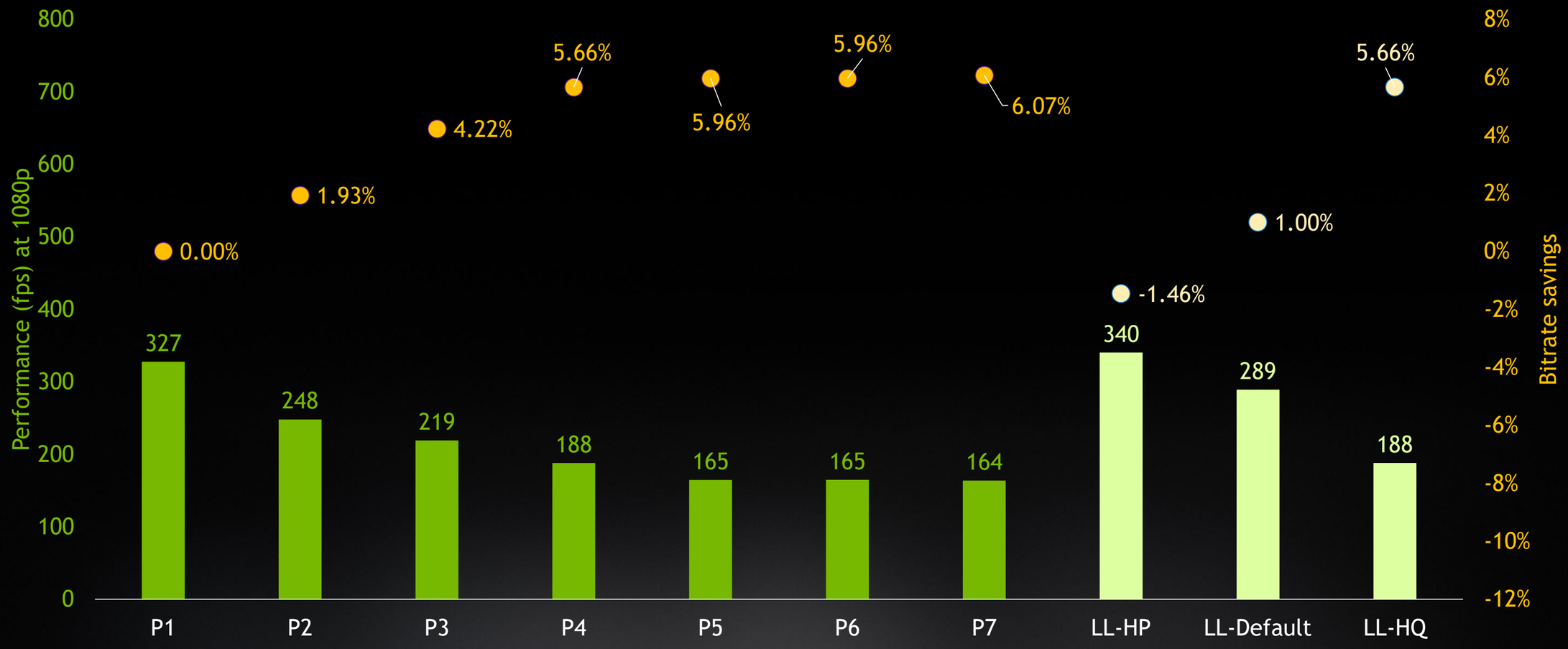
H.264, High quality (latency tolerant), VBR



NVIDIA

NVENC PRESETS: NEW VS LEGACY

HEVC, Ultra Low Latency (latency sensitive), CBR



NVIDIA

MIGRATION TO NEW NVENC PRESETS

Video Codec SDK 10.0

- Strongly recommended to move to new presets
- Mapping table in Video Codec SDK 10.0
- Backward compatibility – Binary ✓ ; Source ✗
- Old presets will be removed from API in future SDK
- SDK release – June 2020


Encode (NVENC) Benchmarks

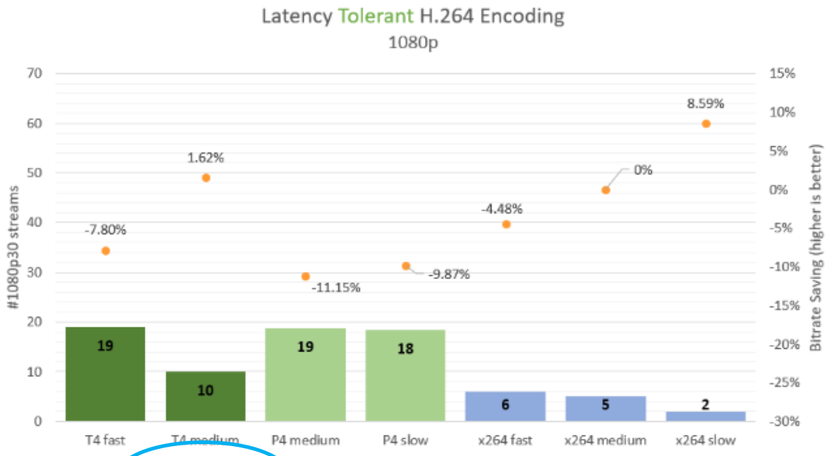
ENCODE BENCHMARKS

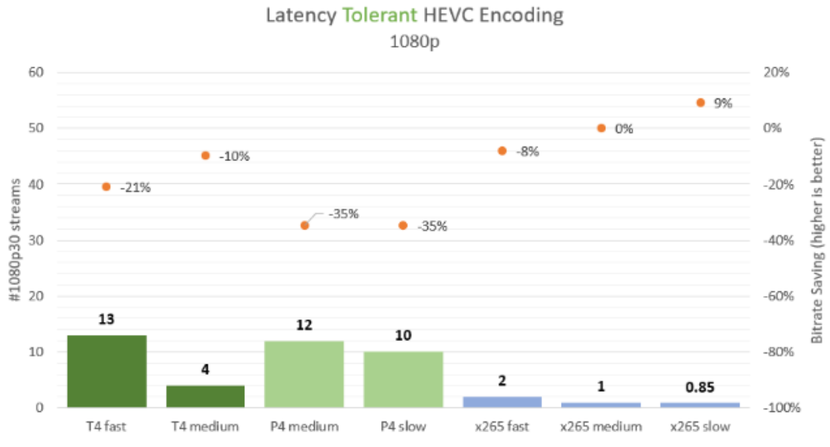
https://developer.nvidia.com/nvidia-video-codec-sdk

NVENC – Hardware-Accelerated Video Encoding

NVIDIA GPUs - beginning with the Kepler generation - contain a **hardware-based encoder (referred to as NVENC)** which provides fully accelerated hardware-based video encoding and is independent of graphics performance. With complete encoding (which is computationally complex) offloaded to NVENC, the graphics engine and the CPU are free for other operations. For example, in a game recording and streaming scenario like streaming to Twitch.tv using Open Broadcaster Software (OBS), encoding being completely offloaded to NVENC makes the graphics engine bandwidth fully available for game rendering. NVENC makes it possible to:

- Encode and stream games and applications at high quality and ultra-low latency without utilizing CPU
- Encode at very high quality for archiving, OTT streaming, web videos
- Encode with ultra-low power consumption per stream (Watts/stream)



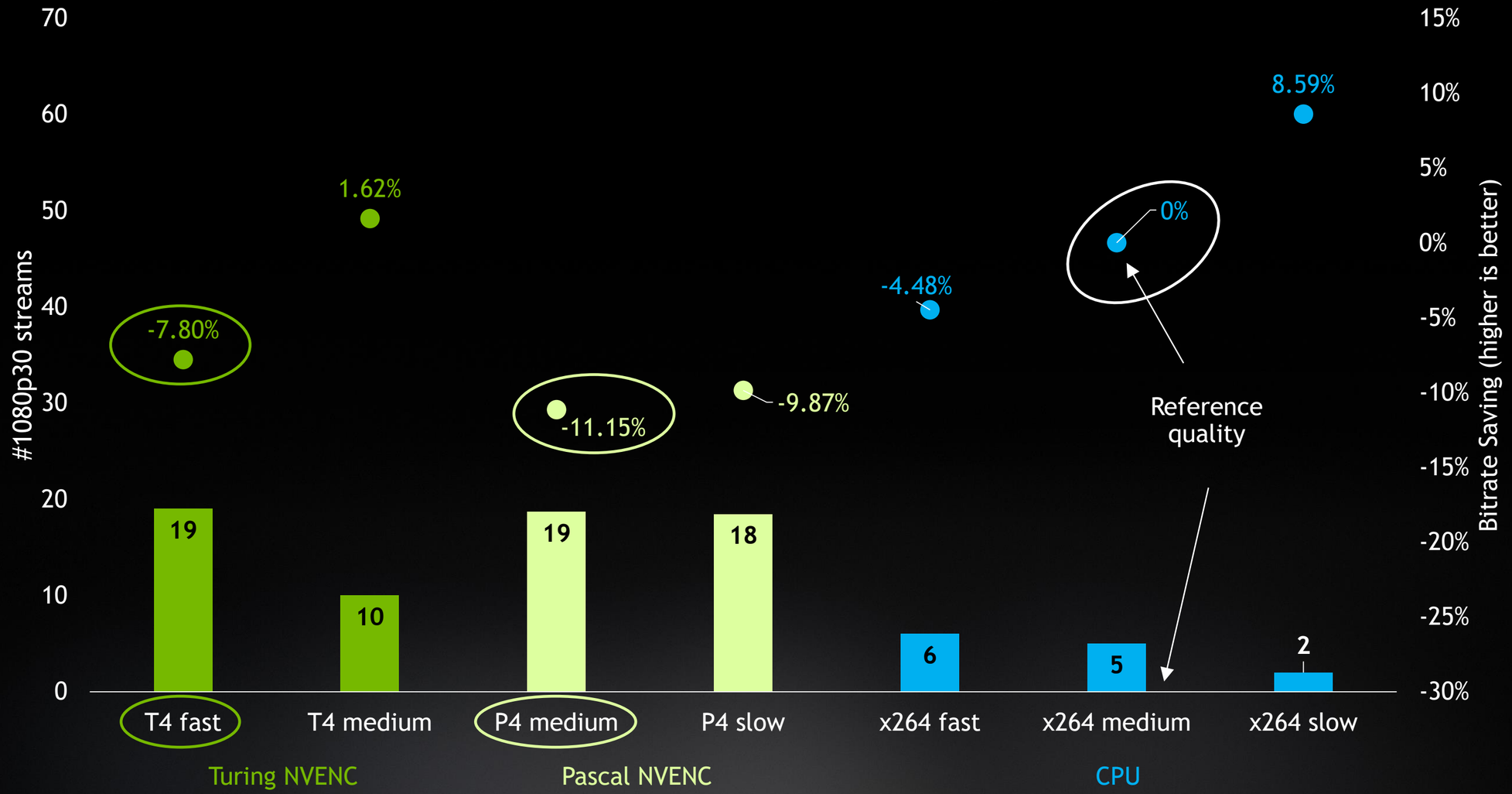


Note: The graphs assume:

- Bitrate savings are BD-BR based on PSNR, average across a large variety of content, using FFmpeg
- In all tests, NVENC is fully loaded and is running at the highest clock
- x264 medium and x265 medium setting are considered "reference" for comparing bitrate savings

ENCODE BENCHMARK – H.264

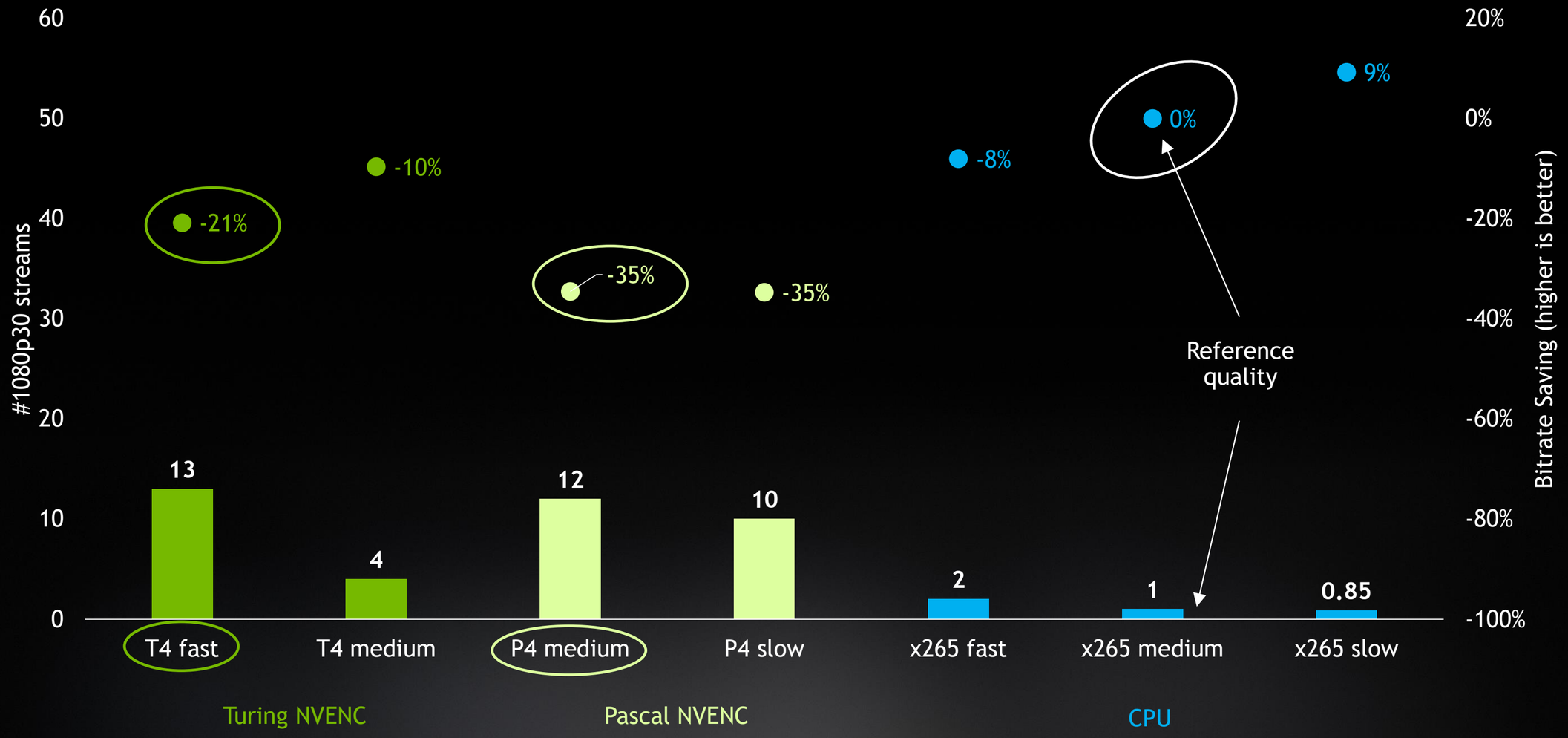
Latency Tolerant H.264 Encoding vs x264



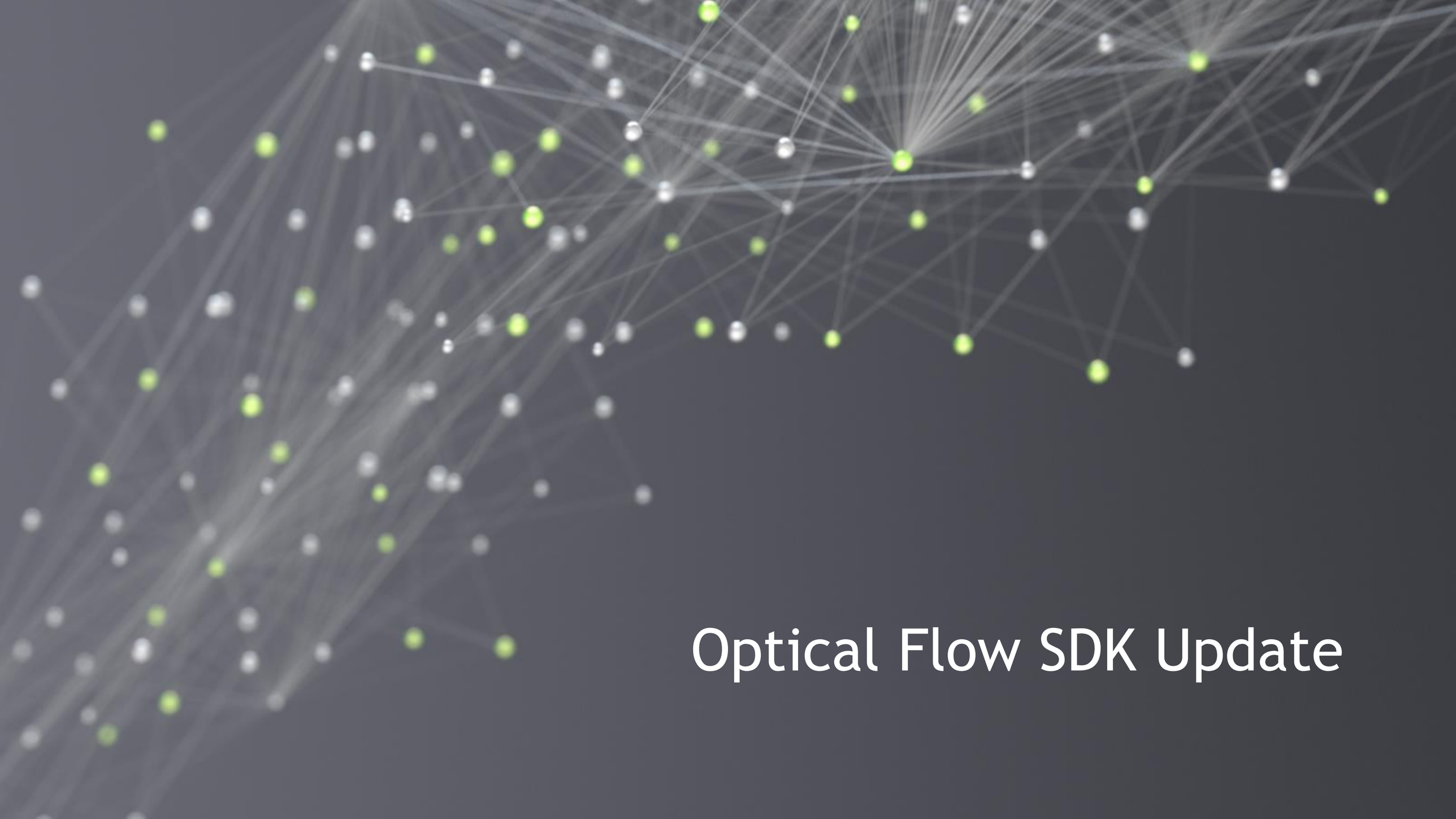
NVIDIA

ENCODE BENCHMARK - HEVC

Latency tolerant HEVC Encoding vs x265

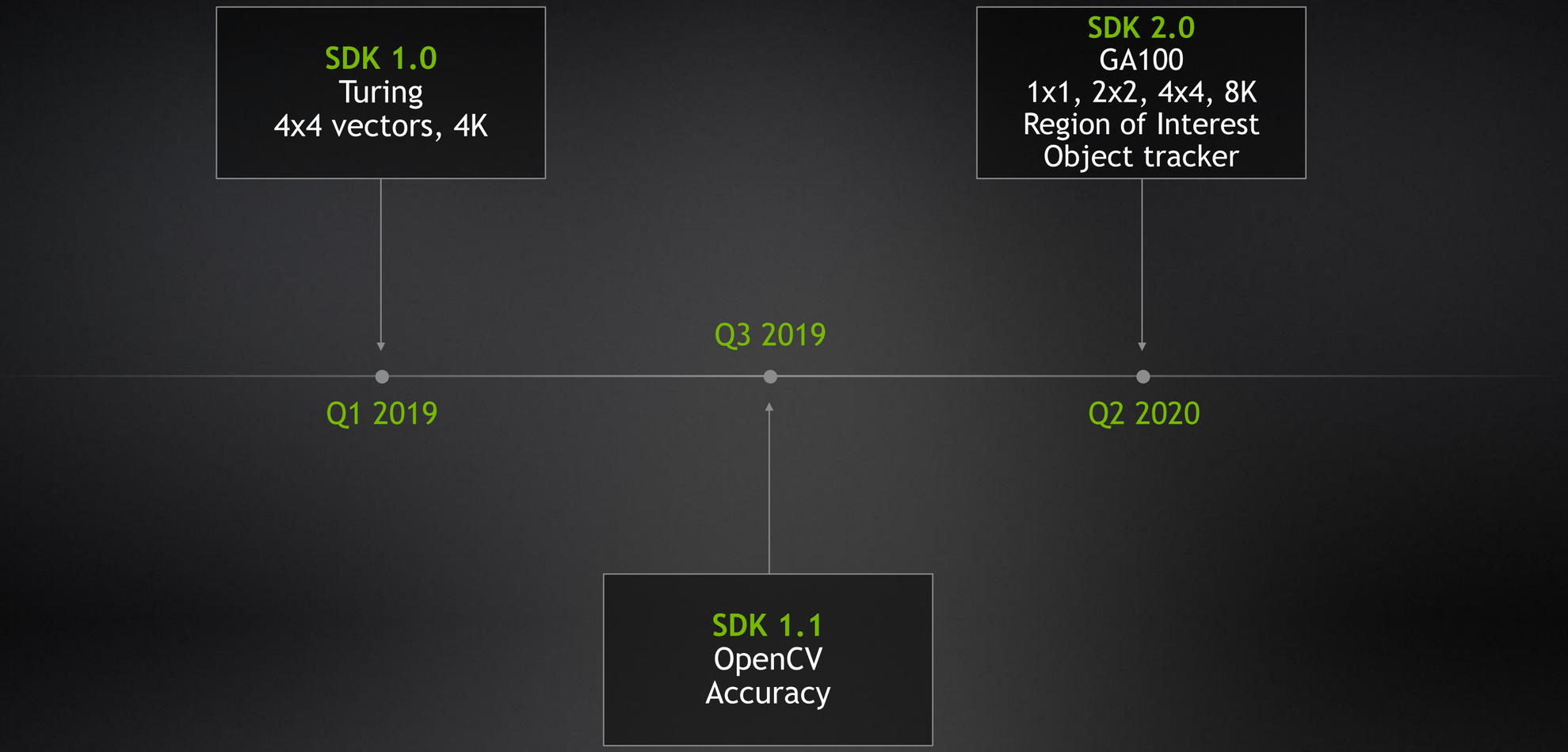


NVIDIA



Optical Flow SDK Update

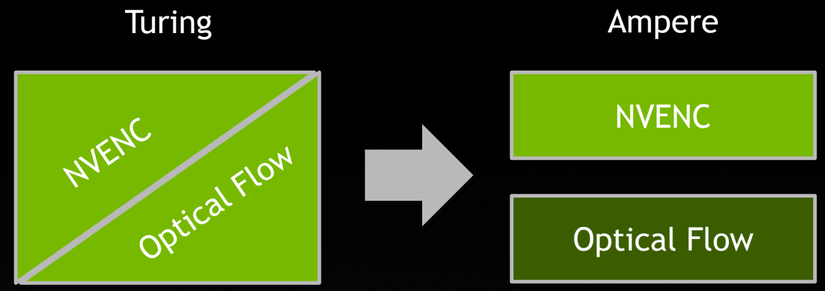
OPTICAL FLOW SDK



OPTICAL FLOW SDK 2.0

What's New?

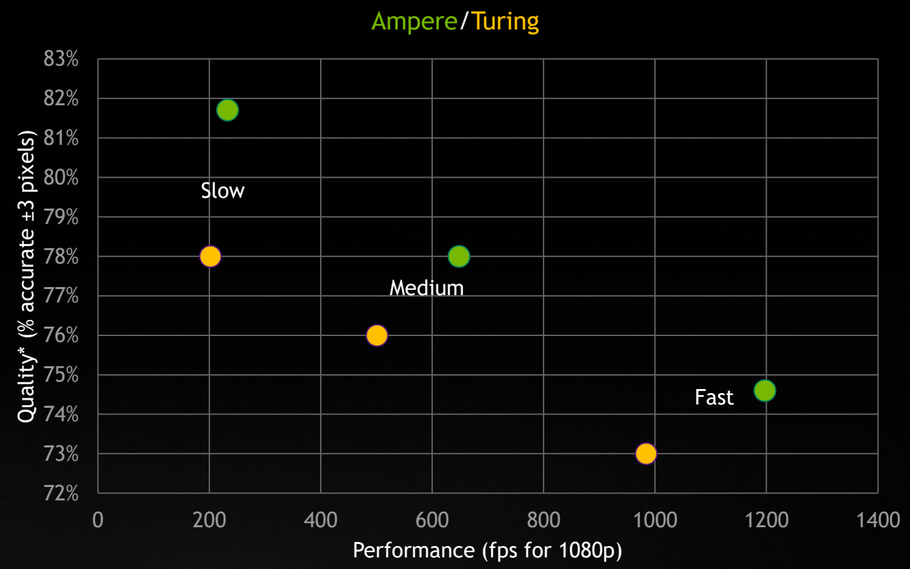
- Ampere architecture GPUs
  - Improved hardware, independent of NVENC



OPTICAL FLOW SDK 2.0

What's New?

- Ampere architecture GPUs
  - Improved hardware, independent of NVENC
  - Better performance and higher accuracy than Turing
  - Up to 300 fps at 4K*

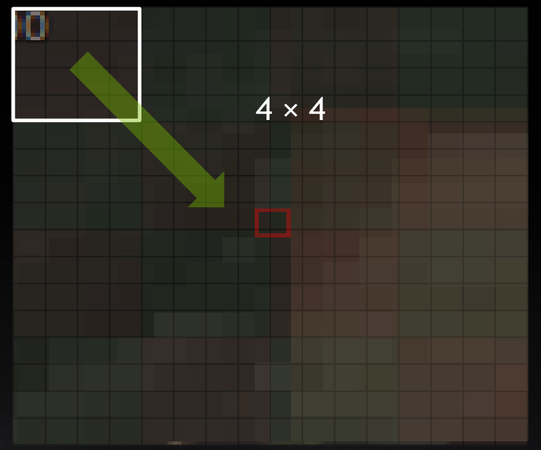


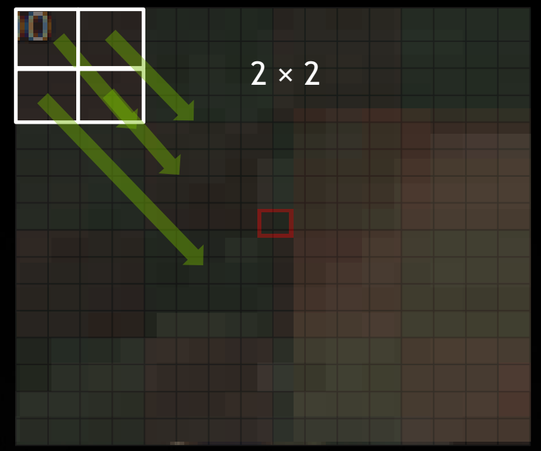
*Performance is dependent on clocks, available memory bandwidth and well-designed application pipelining

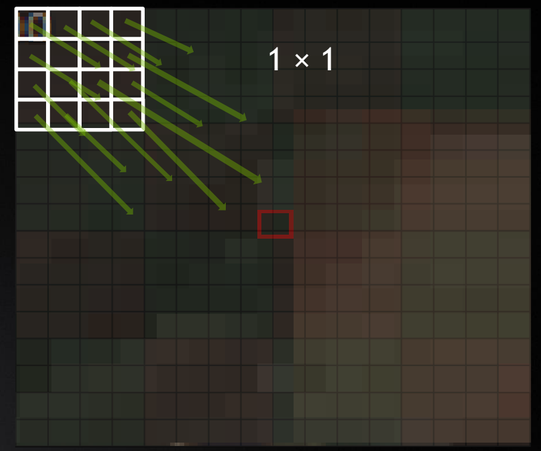
OPTICAL FLOW SDK 2.0

What's New?

- Ampere architecture GPUs
  - Improved hardware, independent of NVENC
  - Better performance and higher accuracy than Turing
  - Up to 300 fps at 4K*
  - Granularity: 1×1, 2×2, 4×4 pixels





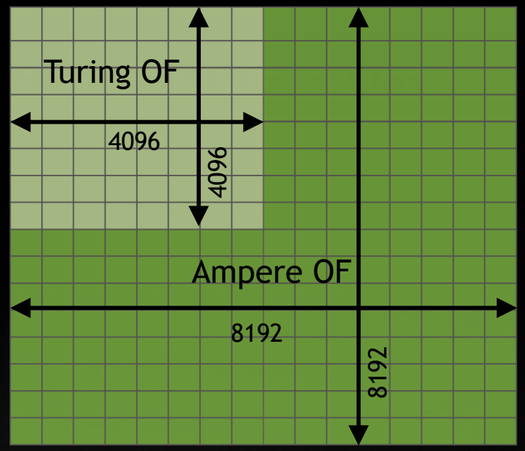


*Performance is dependent on clocks, available memory bandwidth and well-designed application pipelining

OPTICAL FLOW SDK 2.0

What's New?

- Ampere architecture GPUs
  - Improved hardware, independent of NVENC
  - Better performance and higher accuracy than Turing
  - Up to 300 fps at 4K*
  - Granularity: 1×1, 2×2, 4×4 pixels
  - Resolution up to 8192 × 8192

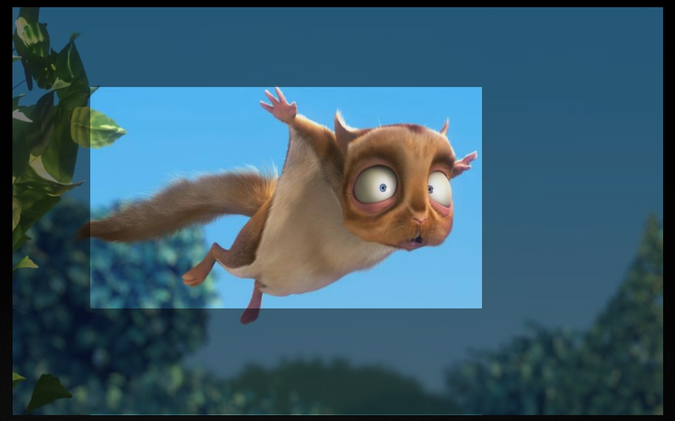


*Performance is dependent on clocks, available memory bandwidth and well-designed application pipelining

OPTICAL FLOW SDK 2.0

What's New?

- Ampere architecture GPUs
  - Improved hardware, independent of NVENC
  - Better performance and higher accuracy than Turing
  - Up to 300 fps at 4K*
  - Granularity: 1×1, 2×2, 4×4 pixels
  - Resolution up to 8192 × 8192
  - Flow vectors for region of Interest (ROI)



**Performance is dependent on clocks, available memory bandwidth and well-designed application pipelining*

OPTICAL FLOW SDK 2.0

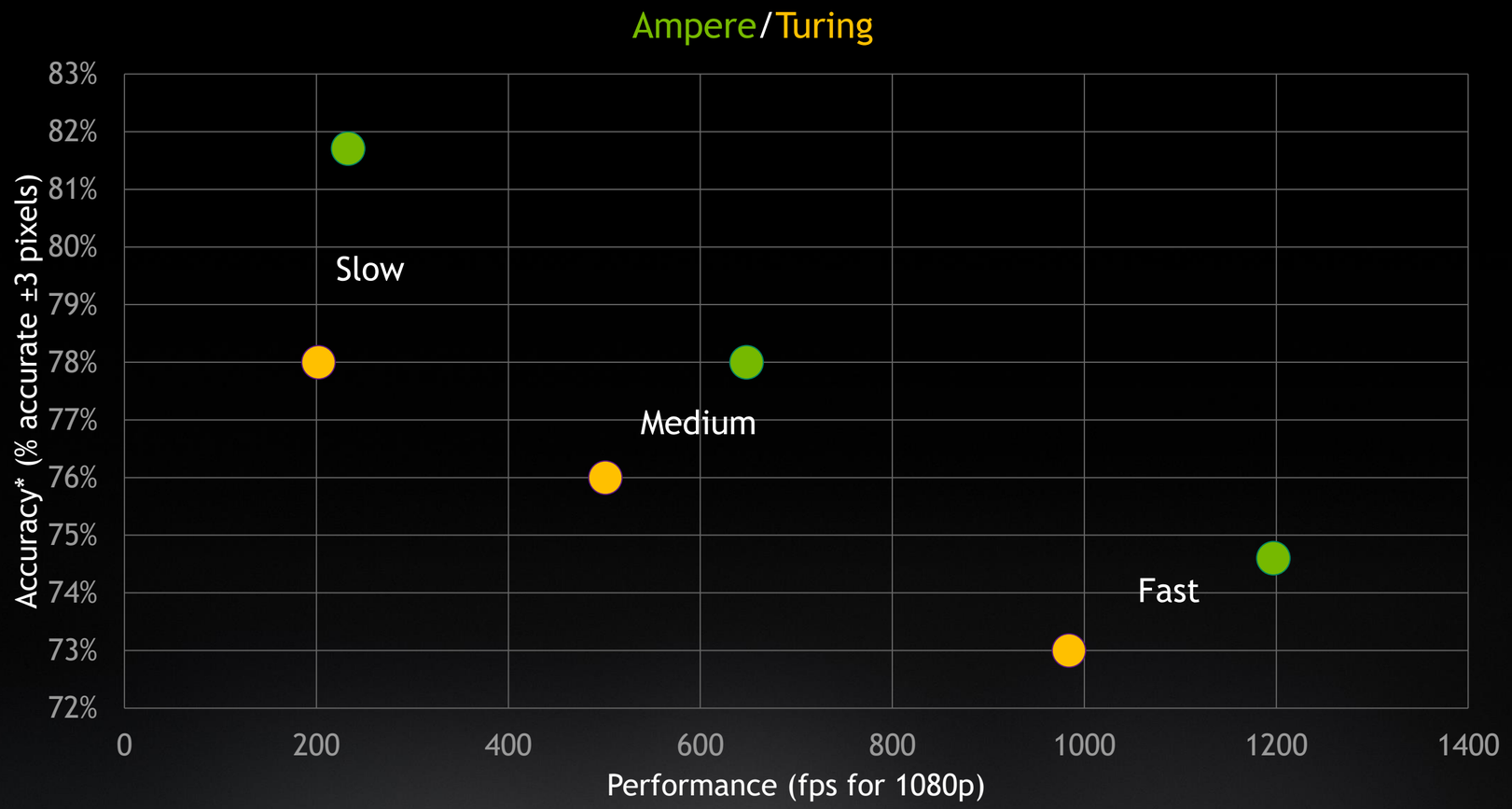
What's New?

- Ampere architecture GPUs
  - Improved hardware, independent of NVENC
  - Better performance and higher accuracy than Turing
  - Up to 300 fps at 4K*
  - Granularity: 1×1, 2×2, 4×4 pixels
  - Resolution up to 8192 × 8192
  - Flow vectors for region of Interest (ROI)
  - ¼ pixel accuracy
  - Improved confidence (cost)
- Optical-Flow based object tracker

**Performance is dependent on clocks, available memory bandwidth and well-designed application pipelining*

OPTICAL FLOW SDK 2.0

Accuracy vs Performance



*KITTI 2015 FL-ALL NOC

OPTICAL FLOW REGION OF INTEREST

Static background, moving foreground objects; e.g. video surveillance

Identify object of interest

Define ROI with extended bounds ±*N* pixels

Advantages: Improved performance, less noise in flow vectors

Main Functionality

nvOpticalFlowCommon.h

CUDA and DirectX Buffer Management

nvOpticalFlowCuda.h

nvOpticalFlowD3D11.h

Reusable Classes

NvOF.h

NvOFCuda.h

NvOFD3D11.h

```
NV_OF_STATUS(NVOFAPI* PFNNVOFINIT)
(NvOFHandle hOf, const NV_OF_INIT_PARAMS
*initParams);

NV_OF_STATUS(NVOFAPI* PFNNVOFEXECUTE)
(NvOFHandle hOf, const
NV_OF_EXECUTE_INPUT_PARAMS
*executeInParams,
NV_OF_EXECUTE_OUTPUT_PARAMS
*executeOutParams);

NV_OF_STATUS(NVOFAPI* PFNNVOFDESTROY)
(NvOFHandle hOf);
```

USE VIA OPENCV

```
Mat frameL = imread(pathL, IMREAD_GRAYSCALE);

Mat frameR = imread(pathR, IMREAD_GRAYSCALE);

GpuMat d_flowL(frameL), d_flowR(frameR), d_flow;

Mat flowx, flowy, flowxy;

int gpuId = 0;

int width = frameL.size().width, height = frameL.size().height;
```

```
Ptr<cuda::FarnebackOpticalFlow> OpticalFlow =
cuda::FarnebackOpticalFlow::create();
OpticalFlow->calc(d_flowL, d_flowR, d_flow);
d_flow.download(flowxy);
```

USE VIA OPENCV

```
Mat frameL = imread(pathL, IMREAD_GRAYSCALE);

Mat frameR = imread(pathR, IMREAD_GRAYSCALE);

GpuMat d_flowL(frameL), d_flowR(frameR), d_flow;

Mat flowx, flowy, flowxy;

int gpuId = 0;

int width = frameL.size().width, height = frameL.size().height;
```

```
Ptr<cuda::NvidiaOpticalFlow> OpticalFlow =
cuda::NvidiaOpticalFlow::create(perfPreset, width, height, gpuId);
OpticalFlow->calc(frameL, frameR, d_flow);
d_flow.download(flowxy);
```

OPTICAL FLOW APPLICATIONS

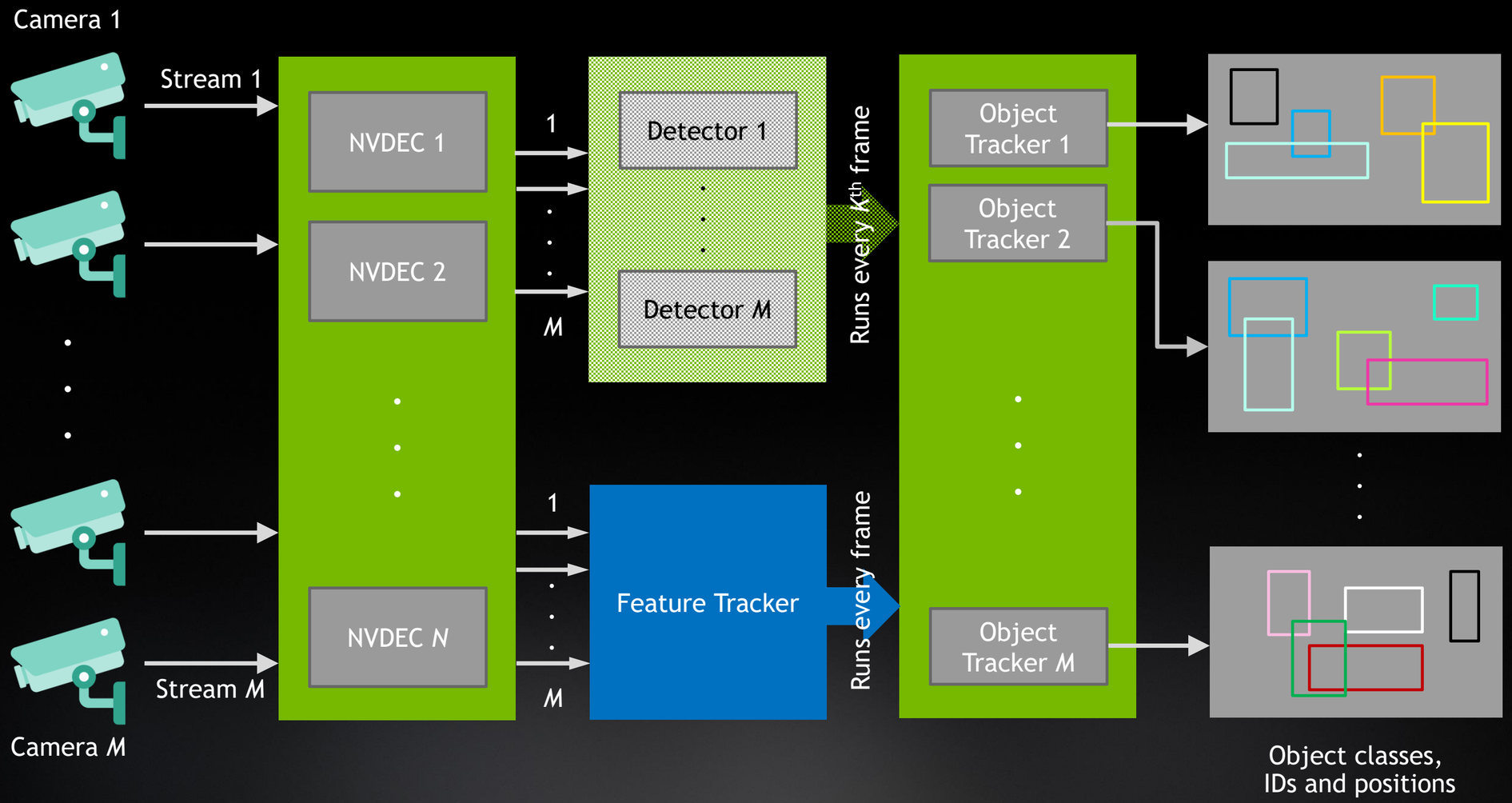
- Object tracking
- Video frame interpolation
- Video frame extrapolation
- Action recognition

OPTICAL FLOW APPLICATIONS

- Object tracking - **API and source code in Optical Flow SDK 2.0**
- Video frame interpolation
- Video frame extrapolation
- Action recognition

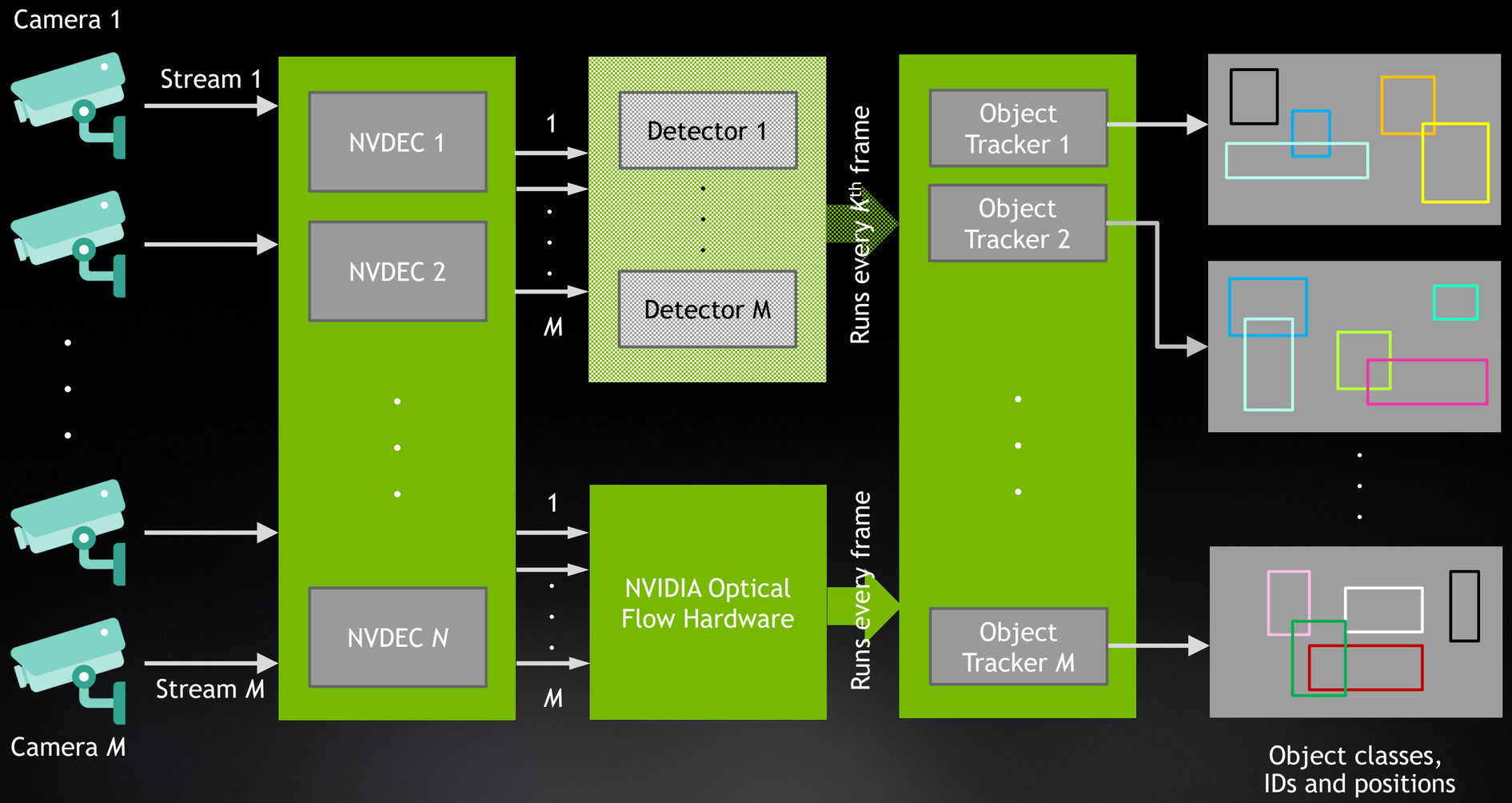
OBJECT TRACKING

Using Feature Tracker or Correlation Filter



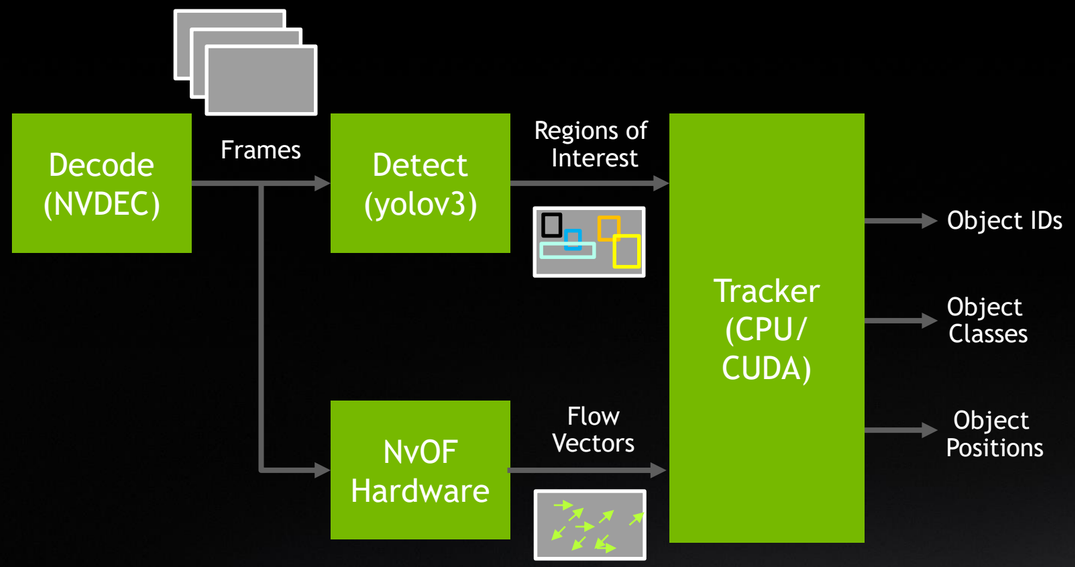
OBJECT TRACKING WITH ZERO GPU USAGE

Optical Flow is "Free"!!



OBJECT TRACKER ALGORITHM

NvOFTracker in Optical Flow SDK 2.0



- Decode video bitstream to frames, *P*-2, *P*-1, *P* (current)
- Detect objects of interest (person, bicycles, cars, …)
  - Define regions of interest to track
  - Runs every K^{th} frame
- Optical Flow (*P*, *P*-1)
  - Dense flow field ➔ Representative flow
  - Warp object position
- Tracker minimizes cost
  - Cost = *f*(Centroid distance, IOU, OF cost, …)

NV OBJECT TRACKER API

Source code in Optical Flow SDK 2.0

NvOFTracker.h

C-API

- NvOFTCreate
- NvOFTProcess
- NvOFTDestroy

COFTracker.h

C++ Reusable Class

- COFTracker::COFTracker
- COFTracker::InitTracker
- COFTracker:: TrackObjects

Sample Code

```
int main(int argc, char** argv) {
    // Instantiate a video decoder
    cv::VideoCapture videoIn(clFields.inputFile, cv::CAP_FFMPEG);

    // Initialize and instantiate object detector
    CDetector objectDetector(detectorEngineFile, gpuId);

    // Instantiate NvOFT Object tracker
    COFTracker objectTracker(w, dh, NvOFT_SURFACE_MEM_TYPE_SYSTEM,
            NvOFT_SURFACE_FORMAT_Y, gpuId);

    while (frame_available) {
        // Read the next video frame
        cv::Mat frame; videoIn >> frame;

        if (nFrame % N == 0)
            // Run detector every Nth frame and get bounding boxes
            boxes = objectDetector.Run(frame.data, frameProperties);
        else
            // Track detected objects in every frame
            trackedObjects = objectTracker.TrackObjects(frame.data,
              frameSize, frame.step[0], inputObjectsVector, FALSE);

        DrawTrackedObjects(…)
    }

    videoIn.release();
}
```

OBJECT TRACKER IN OPTICAL FLOW SDK

Contents

- Source code for end-to-end pipeline with
  - Video decoder
  - Object detector (non-NVIDIA, yoloV3)
  - Object tracker based on NVIDIA optical flow
- API for easy integration
- Sample application and documentation
- Integrable with NVIDIA DeepStream SDK

OBJECT TRACKER PROFILING

Roadmap

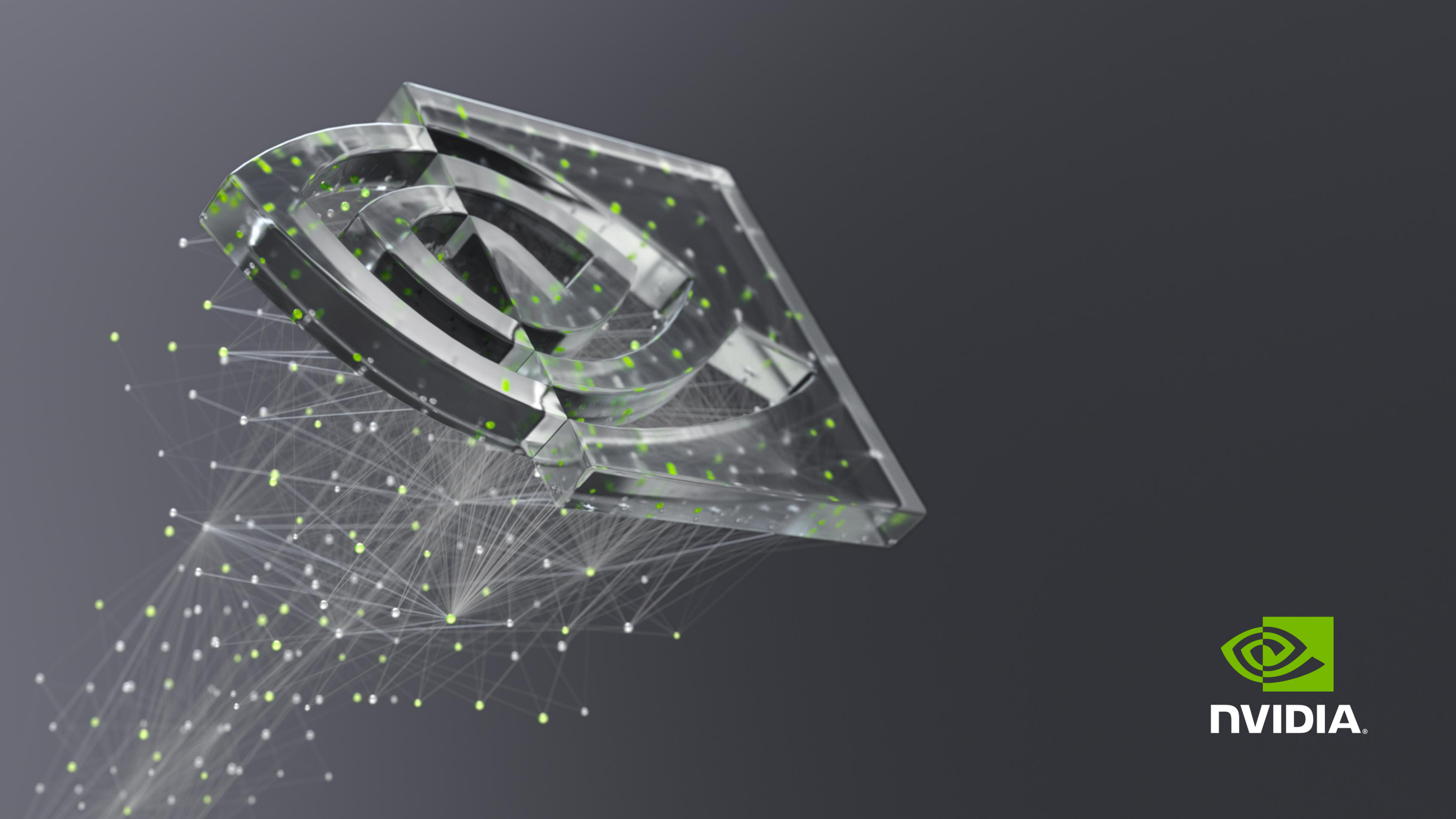
Roadmap

- Video Codec SDK 11.0 (Q3 2020)
  - Deprecation of presets
  - Samples moved to GitHub, online documentation
  - DirectX 12 and Vulkan support via sample apps

- Optical Flow SDK 3.0 (Q3 2020)
  - Ampere architecture GPUs
  - Object detector & tracker
  - Frame rate interpolation/extrapolation
  - Optical vector pre/post-processing APIs

RESOURCES

- Video Codec SDK: https://developer.nvidia.com/nvidia-video-codec-sdk
- Optical Flow SDK: https://developer.nvidia.com/nvidia-video-codec-sdk
- Video Technologies Developer Forum: https://devtalk.nvidia.com/default/board/175/video-technologies/
- Support: video-devtech-support@nvidia.com
- CWE21120: How to Use Video Codec and Optical Flow SDK on NVIDIA GPUs Effectively


NVIDIA