GPU-ACCELERATED VIDEO ENCODING/DECODING

CIS 565 Spring 2012

Philip Peng, 2012/03/19

Overview

1. Motivation
2. NVIDIA GPU
3. Compression Techniques
4. NPP & Performance
5. NVIDIA API Demo

MOTIVATION

Why use GPU?

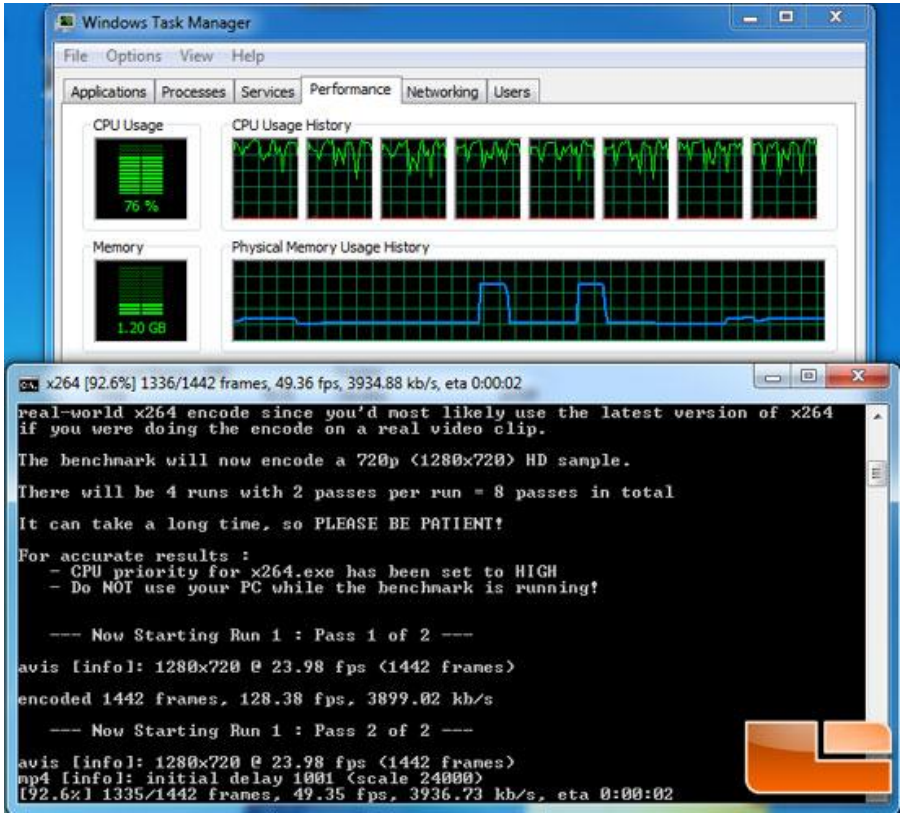
Motivation

- Why video encoding/decoding?
  - Large part of consumer activities
  - Watch YouTube videos everywhere
  - Encode HD videos to iPhone/etc.

Img src: http://cdn3.digitaltrends.com/wp-content/uploads/2010/07/youtube-and-iphone-20.jpg

Motivation

- Why use GPU?
  - Portable devices have limited processing power
  - Very computationally intensive
  - Faster + smaller = better



Img src: http://www.legitreviews.com/images/reviews/1501/x264_cpu.jpg

NVIDIA GPU

What’s currently out there?

NVIDIA GPU Video Encoding

- Facilities
  - SW H.264 codec designed for CUDA
    - Supports “baseline”, “main”, “high” profiles
- Interfaces:
  - C library (NVCUVENC)
  - Direct Show API
  - Win 7 MFT (multimedia framework)

NVIDIA GPU Video Decoding

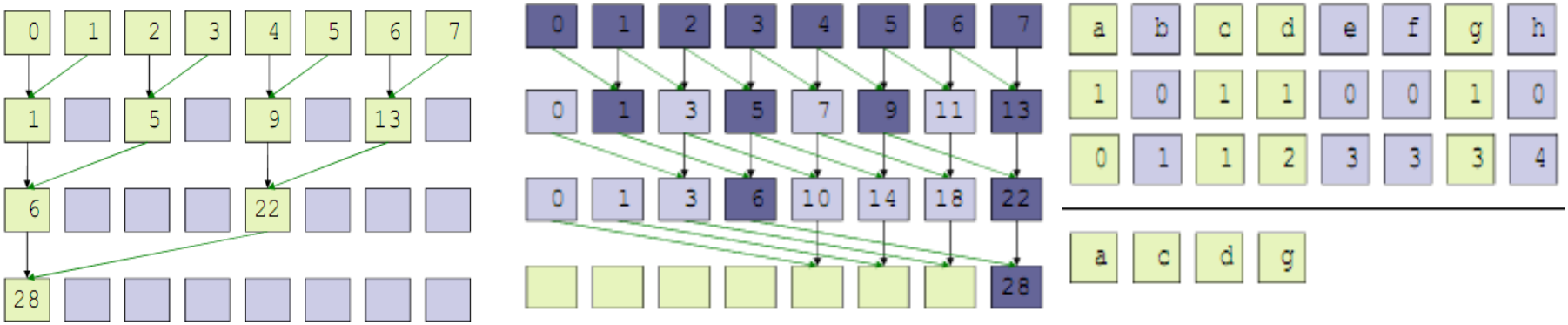
- Facilities
  - HW GPU acceleration
    - H.264, VC1, MPEG2
  - SW MPEG2 decoder designed for CUDA
- Interfaces:
  - C library (NVCUVIV) – HW & SW
  - DXVA & Win7 MFT – HW only
  - VDPAU library – HW only (Linux)

COMPRESSION TECHNIQUES

How is it done?

GPU Algorithm Primitives

- Fundamental building blocks for Parallel Programming
- Reduce: sum of absolute differences
- Scan: integral image calculations
- Compact: index creation
- CUDA C SDK, CUDPP, Thrust implementations



Img src: http://cis565-spring-2012.github.com/lectures/02-01-Parallel-Algorithms.pdf

Technique: Intra Prediction

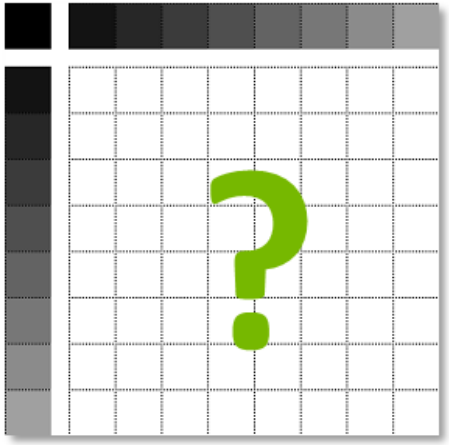
Img src: http://www.nvidia.com/content/GTC-2010/pdfs/2075_GTC2010.pdf

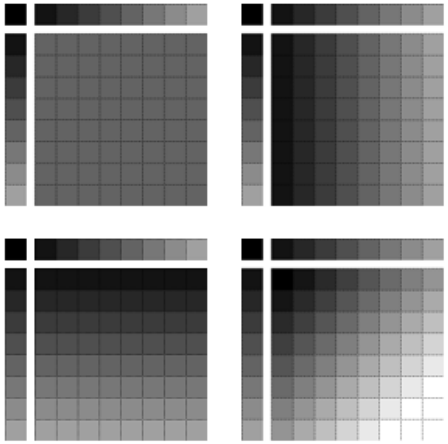
Technique: Intra Prediction

Img src: http://www.nvidia.com/content/GTC-2010/pdfs/2075_GTC2010.pdf

Technique: Intra Prediction

- Video processing technique via content prediction
  - Uses data within single video frame
  - Divides frame into blocks
  - Prediction rules
    - Constant: each pixel constant value
    - Vertical: each row uses top predictor
    - Horizontal: each column uses left predictor
    - Plane: gradient diagonal prediction





Img src: http://www.nvidia.com/content/GTC-2010/pdfs/2075_GTC2010.pdf

Technique: Intra Prediction

- For each block, calculate SAD between predicted pixels and actual

$$SAD = \sum_{x,y} \left| src[y][x] - predict[y][x] \right|$$

- Choose predictor with lowest prediction error
- Data compression = only store predictors and prediction rules

Img src: http://www.nvidia.com/content/GTC-2010/pdfs/2075_GTC2010.pdf

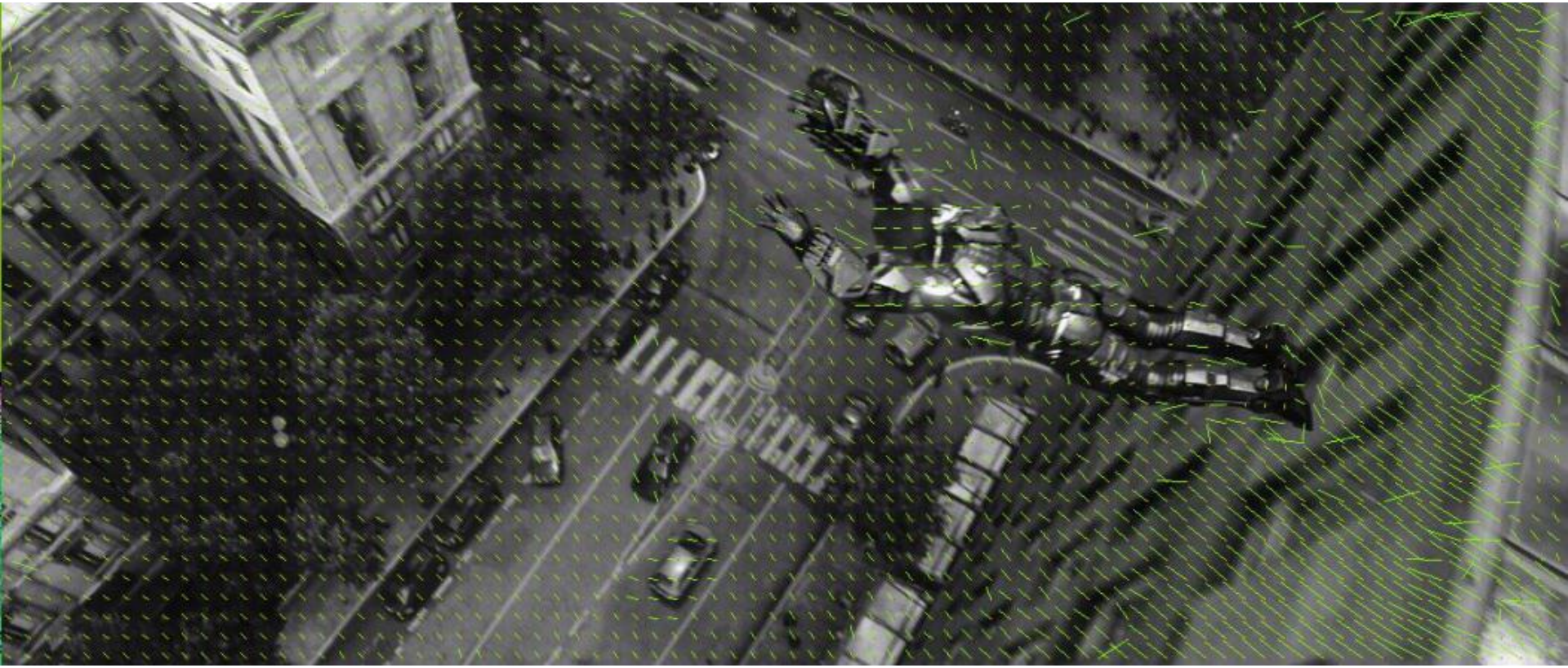
Technique: Motion Estimation

Img src: http://www.nvidia.com/content/GTC-2010/pdfs/2075_GTC2010.pdf

Technique: Motion Estimation

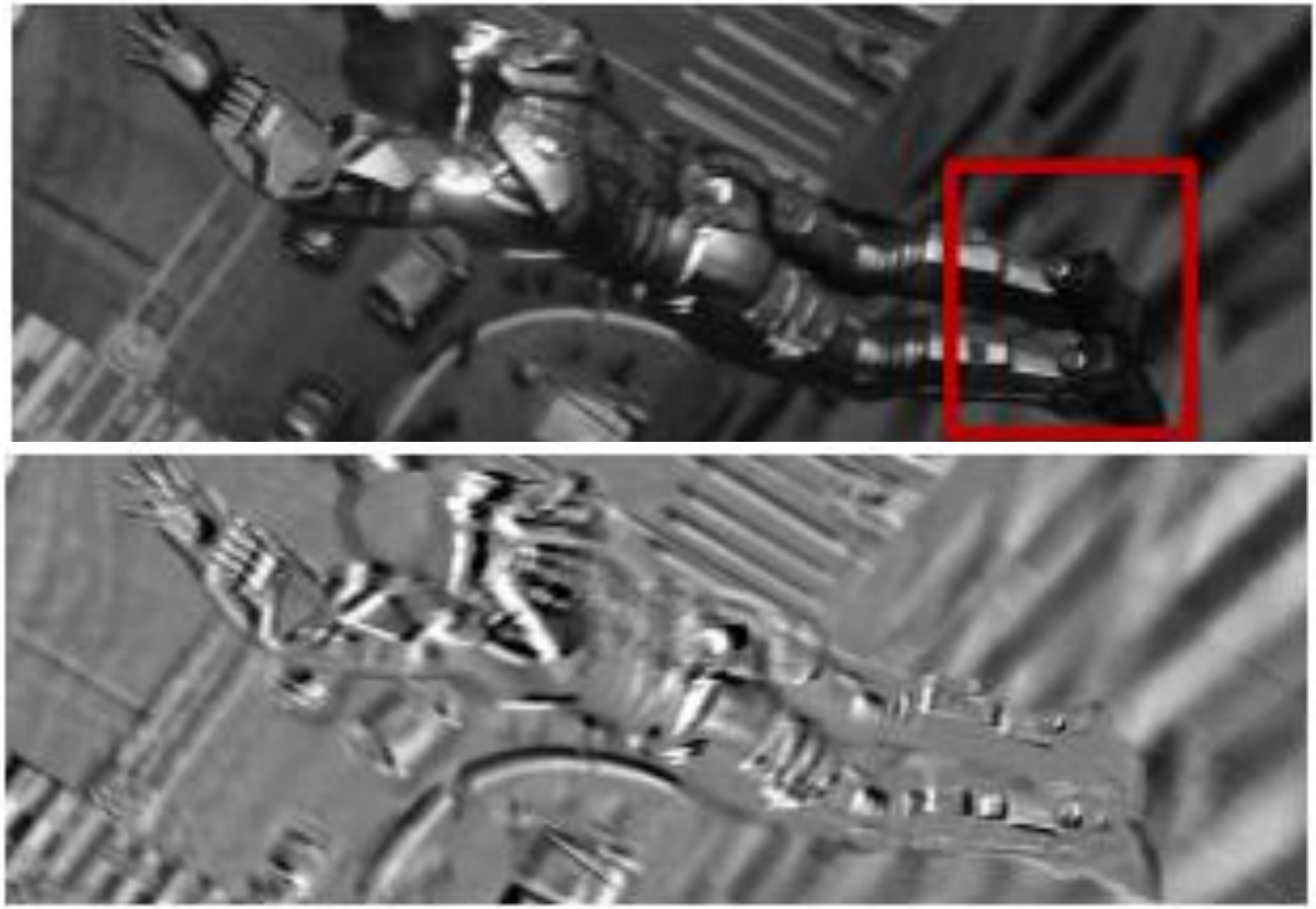
Img src: http://www.nvidia.com/content/GTC-2010/pdfs/2075_GTC2010.pdf

Technique: Motion Estimation



Img src: http://www.nvidia.com/content/GTC-2010/pdfs/2075_GTC2010.pdf

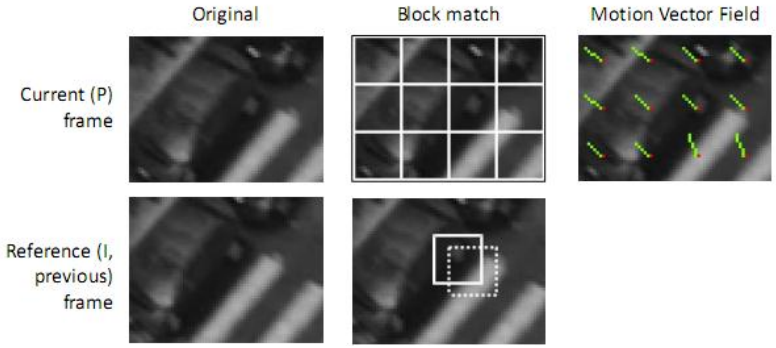
Technique: Motion Estimation

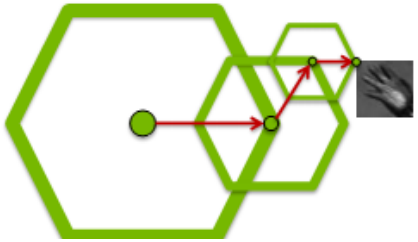


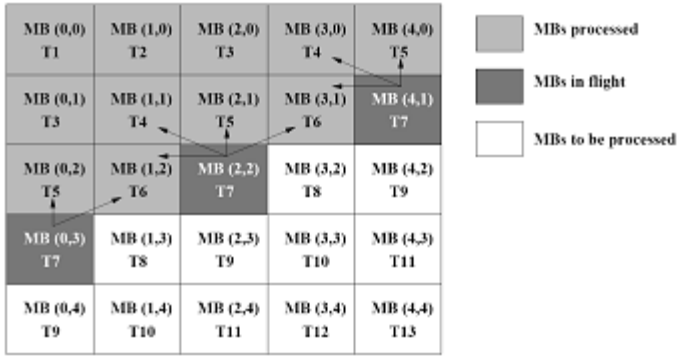
Img src: http://www.nvidia.com/content/GTC-2010/pdfs/2075_GTC2010.pdf

Technique: Motion Estimation

- Many different search algorithms for calculating motion vectors:
  - Full search: brute force feature comparison
  - Diamond search: template-based iterative matching
  - 2D/3D-wave: diagonal block-based checking
- Combination done in parallel on GPU







Img src: http://www.nvidia.com/content/GTC-2010/pdfs/2075_GTC2010.pdf

NPP & PERFORMANCE

GPU vs CPU

NVIDIA Performance Primitives

- What is it?
  - NPP = C library of GPU-accelerated functions/primitives designed for CUDA
  - API same as IPP (Intel Integrated Performance Primitives)
  - No GPU architecture knowledge required!
  - 5-10x faster performance vs CPU-only implementations
  - http://developer.nvidia.com/npp

NVIDIA Performance Primitives Code

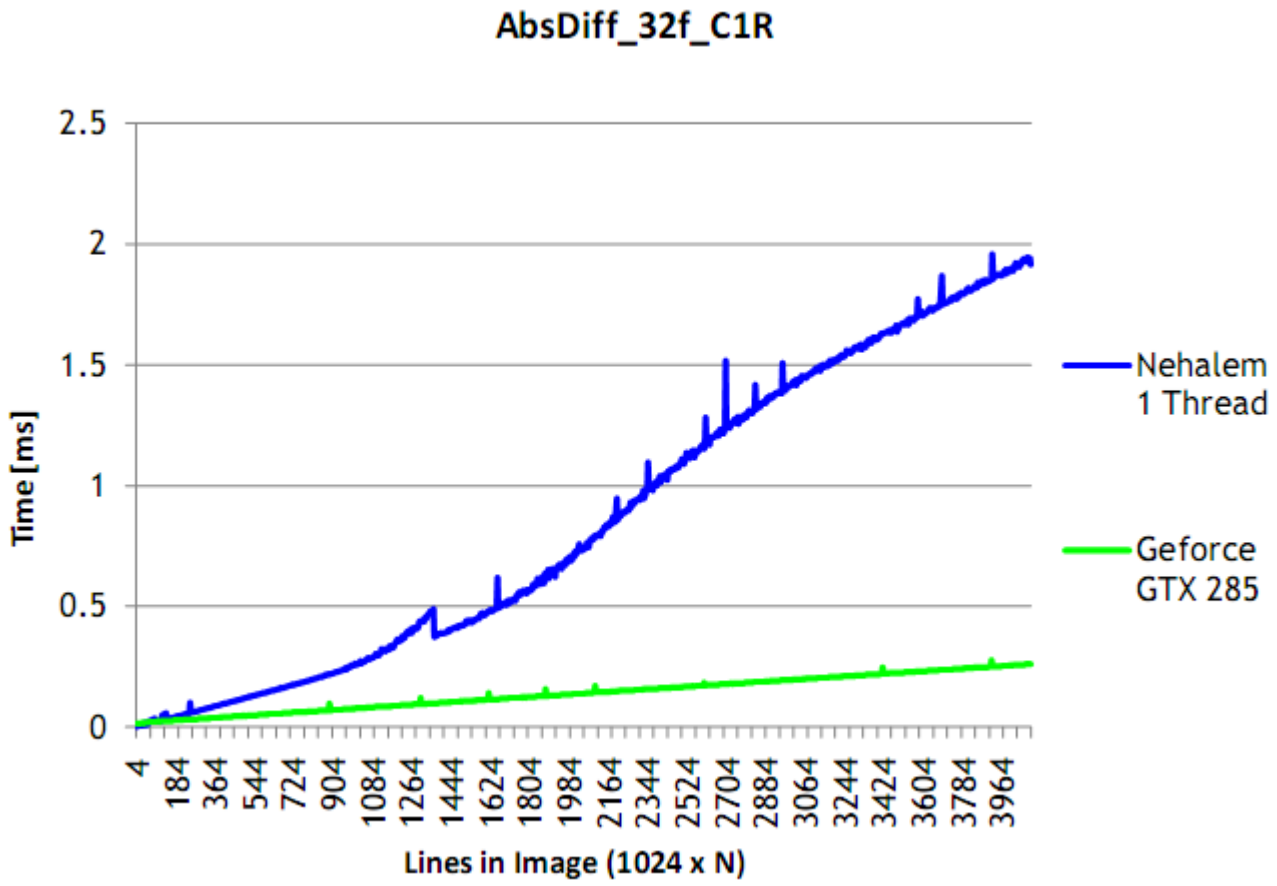
```
        // allocate source image
int sp;
Ipp8u * pSI = ippiMalloc_8u_C1(w, h, &sp);
        // fill with some image content
testPattern_8u_C1(pSI, sp, w, h);





        // allocated destination image
int dp;
Ipp8u * pDI = ippiMalloc_8u_C1(w, h, &dp);
        // Filter mask and achor
IppiSize  mask   = {5, 5};
IppiPoint anchor = {0, 0};
IppiSize  ROI    = {w - mask.width  + 1,
                    h - mask.height + 1};
        // run box filter
ippiFilterBox_8u_C1R(pSI, sp, pDI, dp,
                     ROI, mask, anchor);
```

```
        // allocate host source image
int hp;
Ipp8u * pHI = ippiMalloc_8u_C1(w, h, &hp);
        // fill with some image content
testPattern_8u_C1(pHI, hp, w, h);
        // allocated device source image
int sp;
Npp8u * pSI = nppiMalloc_8u_C1(w, h, &sp);
        // copy test image up to device
cudaMemcpy2D(pSI, sp, pHI, hp, w, h,
             cudaMemcpyHostToDevice);
        // allocate device result image
int dp;
Npp8u * pDI = nppiMalloc_8u_C1(w, h, &dp);
        // Filter mask and achor
NppiSize  mask   = {5, 5};
NppiPoint anchor = {0, 0};
NppiSize  ROI    = {w - mask.width  + 1,
                    h - mask.height + 1};
        // run box filter
nppiFilterBox_8u_C1R(pSI, sp, pDI, dp,
                     ROI, mask, anchor);
```


Img src: http://www.nvidia.com/content/GTC/documents/1028_GTC09.pdf

NVIDIA Performance Primitives Tests

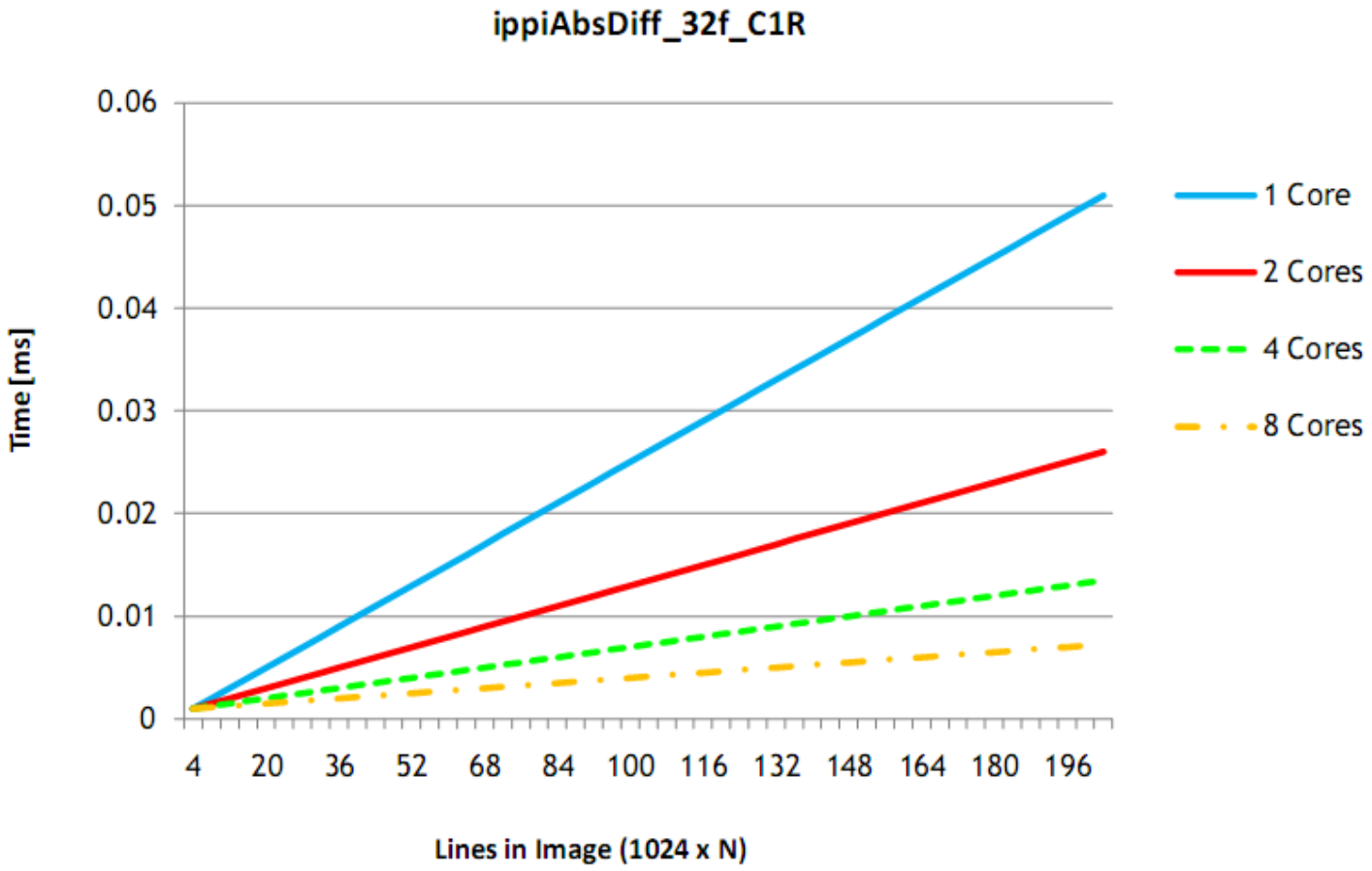
- Data scalability:



Img src: http://www.nvidia.com/content/GTC/documents/1028_GTC09.pdf

NVIDIA Performance Primitives Tests

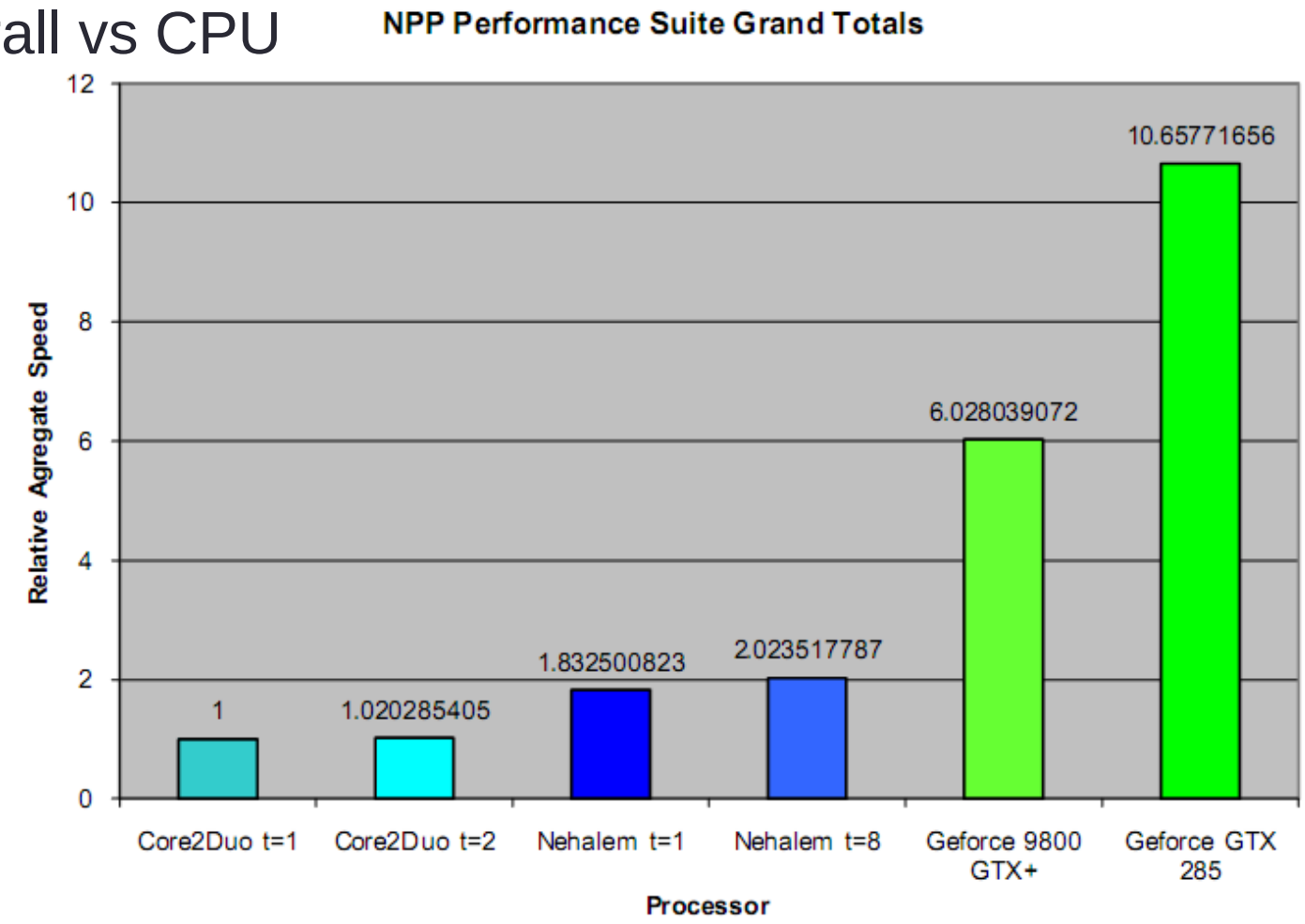
- Number of cores scalability:



Img src: http://www.nvidia.com/content/GTC/documents/1028_GTC09.pdf

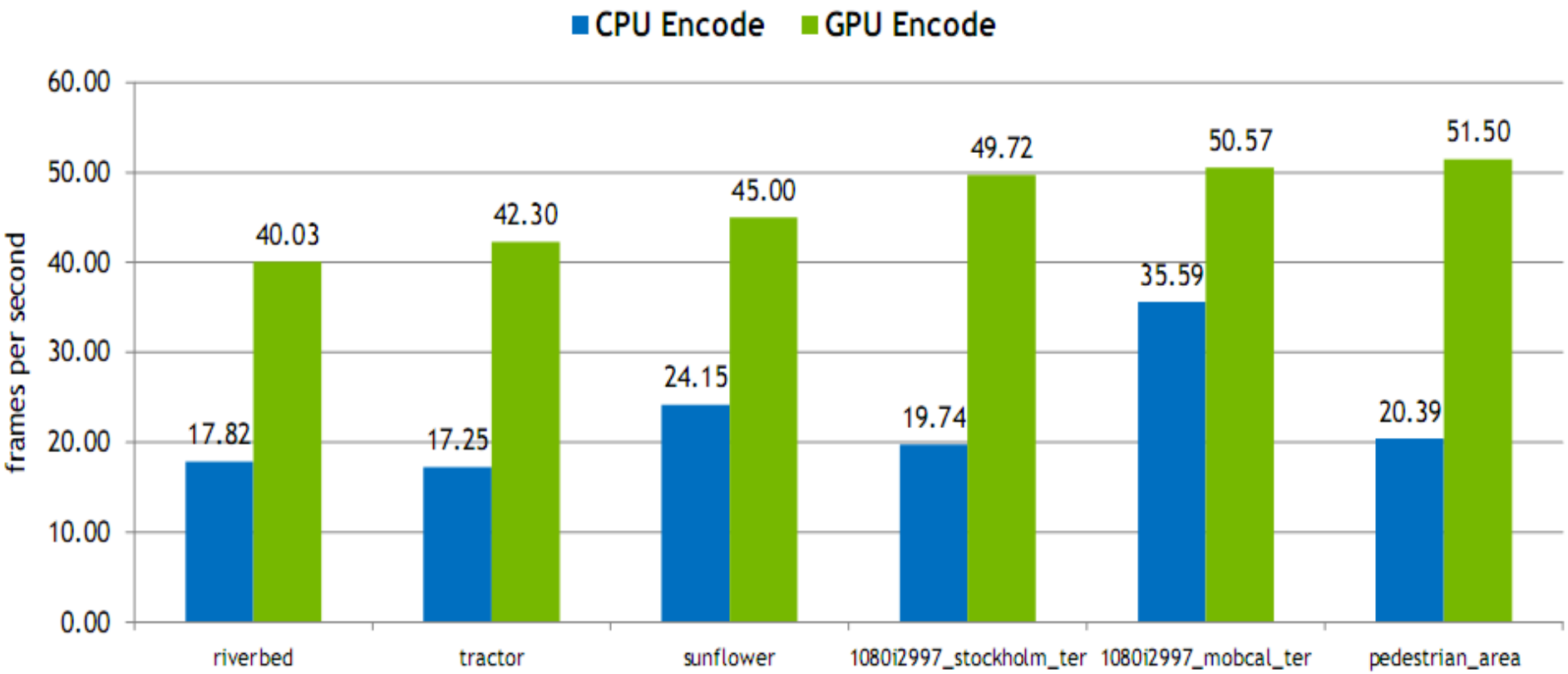
NVIDIA Performance Primitives Tests

- Overall vs CPU



Img src: http://www.nvidia.com/content/GTC/documents/1028_GTC09.pdf

Encoding Performance



Img src: http://www.nvidia.com/content/GTC/documents/1028_GTC09.pdf

Decoding Performance

Without NVIDIA GPU

H.264: Bitstream Processing CAVLC CABAC | Inverse Transform | Motion Compenstation | Deblocking

VC-1: Bitstream Processing CAVLC CABAC | Inverse Transform | Motion Compenstation | Deblocking

MPEG-2: Bitstream Processing CAVLC CABAC | Inverse Transform | Motion Compenstation

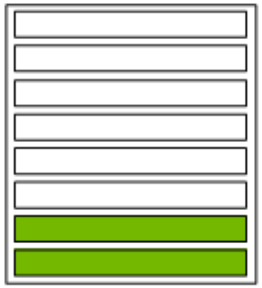
High CPU Utilization

H.264: Bitstream Processing CAVLC CABAC | Inverse Transform | Motion Compenstation | Deblocking

VC-1: Bitstream Processing CAVLC CABAC | Inverse Transform | Motion Compenstation | Deblocking

MPEG-2: Bitstream Processing CAVLC CABAC | Inverse Transform | Motion Compenstation

Minimal CPU Utilization



Img src: http://www.nvidia.com/content/GTC/documents/1028_GTC09.pdf

NVIDIA API DEMO

See the numbers for yourself!

NVIDIA CUDA Video Encode API

http://developer.nvidia.com/cuda-cc-sdk-code-samples#cudaEncode

C:\Windows\system32\cmd.exe

```
=====================
NVIDIA CUDA (GPU) version
Running: cudaEncode.exe plush_480p_60fr.yuv 704x480-h264.c
=====================

Press any key to continue . . .
[cudaEncode.exe] starting...

[ CUDA H.264 Encoder ]
  argv[0] = cudaEncode.exe
  argv[1] = plush_480p_60fr.yuv
  argv[2] = 704x480-h264.cfg
  argv[3] = output_cuda.264

Configuration  file: <704x480-h264.cfg>
Source  input  file: <plush_480p_60fr.yuv>
Encoded output file: <output_cuda.264>
Measurement: (FPS) Frames Per Second

VideoEncoder() - NVCreateEncoder (SUCCESS!), hr = 00000000
  NVSetCodec (H.264 Video)
[cudaEncode.exe] - [Frame: 0016, 200.2 fps, frame time: 5.
[cudaEncode.exe] - [Frame: 0032, 144.8 fps, frame time: 6.
[cudaEncode.exe] - [Frame: 0048, 149.3 fps, frame time: 6.

[VideoEncoder() (Stopped)]
  [input]  = plush_480p_60fr.yuv
  [output] = output_cuda.264

[H.264 Encoding Statistics]
        Number of Coded Frames      : 60
        Elapsed time (hh:mm:ss:ms)  : [redacted]
        Average FPS (end to end)    : 138.568130
        CPU utilization (8 cores)   : [redacted]:43.23%,

> cudaEncode.exe encoded OK, return value = 0
[cudaEncode.exe] test results...
PASSED

> exiting in 3 seconds: 3...2...1...done!

Press any key to continue . . .
```

GPU CUDA = 138.57 FPS

C:\Windows\system32\cmd.exe

```
=====================
x264 (CPU) version - Processing FPS match
Running: x264_x64.exe --input-res 704x480 -B 1500 -o ou
80p_60fr.yuv
=====================

Press any key to continue . . .
yuv [info]: 704x480p 0:0 @ 25/1 fps (cfr)
x264 [info]: using cpu capabilities: MMX2 SSE2Fast SSSI
x264 [info]: profile High, level 3.0
x264 [info]: frame I:1     Avg QP:19.59  size: 50405
x264 [info]: frame P:15    Avg QP:18.23  size: 12999
x264 [info]: frame B:44    Avg QP:22.87  size:  1639
x264 [info]: consecutive B-frames:  1.7%  0.0%  5.0% 93
x264 [info]: mb I  I16..4:  4.4% 47.2% 48.4%
x264 [info]: mb P  I16..4:  0.3%  2.5%  1.9%  P16..4: 3
.0%    skip:28.9%
x264 [info]: mb B  I16..4:  0.0%  0.0%  0.0%  B16..8: 2
 1.5%  skip:68.3%  L0:30.4% L1:60.7% BI: 9.0%
x264 [info]: final ratefactor: 14.38
x264 [info]: 8x8 transform intra:49.7% inter:58.0%
x264 [info]: coded y,uvDC,uvAC intra: 90.7% 81.3% 61.8%
x264 [info]: i16 v,h,dc,p: 22% 46%  5% 27%
x264 [info]: i8 v,h,dc,ddl,ddr,vr,hd,vl,hu: 16% 37% 12%
x264 [info]: i4 v,h,dc,ddl,ddr,vr,hd,vl,hu: 24% 39%  7%
x264 [info]: i8c dc,h,v,p: 39% 34% 23%  5%
x264 [info]: Weighted P-Frames: Y:0.0% UV:0.0%
x264 [info]: ref P L0: 69.7% 14.9% 12.4%  3.0%
x264 [info]: ref B L0: 92.2%  6.8%  1.0%
x264 [info]: ref B L1: 95.3%  4.7%
x264 [info]: kb/s:1058.37

encoded 60 frames, 125.26 fps, 1058.37 kb/s

Press any key to continue . . .
```

CPU x264 = 125.26 FPS

C:\Windows\system32\cmd.exe

```
=====================
x264 (CPU) version - Filesize match
Running: x264_x64.exe --input-res 704x480 -B 4000 -o
480p_60fr.yuv
=====================

Press any key to continue . . .
yuv [info]: 704x480p 0:0 @ 25/1 fps (cfr)
x264 [info]: using cpu capabilities: MMX2 SSE2Fast SS
x264 [info]: profile High, level 3.0
x264 [info]: frame I:1     Avg QP:11.18  size:107684
x264 [info]: frame P:15    Avg QP:10.69  size: 41138
x264 [info]: frame B:44    Avg QP:14.94  size:  6145
x264 [info]: consecutive B-frames:  1.7%  0.0%  5.0%
x264 [info]: mb I  I16..4:  4.3% 36.1% 59.6%
x264 [info]: mb P  I16..4:  0.6%  5.1%  4.9%  P16..4:
.0%    skip: 7.6%
x264 [info]: mb B  I16..4:  0.1%  0.4%  0.3%  B16..8:
 4.9%  skip:52.3%  L0:35.0% L1:49.4% BI:15.6%
x264 [info]: final ratefactor: 8.48
x264 [info]: 8x8 transform intra:45.0% inter:41.4%
x264 [info]: coded y,uvDC,uvAC intra: 98.1% 96.9% 94.

x264 [info]: i16 v,h,dc,p: 11% 28% 22% 39%
x264 [info]: i8 v,h,dc,ddl,ddr,vr,hd,vl,hu: 13% 40% 1
x264 [info]: i4 v,h,dc,ddl,ddr,vr,hd,vl,hu: 19% 37% 1
x264 [info]: i8c dc,h,v,p: 44% 33% 17%  5%
x264 [info]: Weighted P-Frames: Y:0.0% UV:0.0%
x264 [info]: ref P L0: 71.3% 10.9% 13.9%  3.8%
x264 [info]: ref B L0: 90.1%  8.4%  1.5%
x264 [info]: ref B L1: 95.9%  4.1%
x264 [info]: kb/s:3317.17

encoded 60 frames, 96.15 fps, 3317.17 kb/s

Press any key to continue . . .
```

CPU x264 = 96.15 FPS

NVIDIA CUDA Video Decoder GL API

http://developer.nvidia.com/cuda-cc-sdk-code-samples#cudaDecodeGL

```
MPlayer: playing plush1_720p_10s.m2v

=====================
MPlayer benchmark decoding (novsync)
Running: mplayer.exe -benchmark plush1_720p_10s.m2v
=====================

Press any key to continue . . .
MPlayer Sherpya-SVN-r29355-4.5.0 (C) 2000-2009 MPlayer Team

Playing plush1_720p_10s.m2v.
MPEG-ES file format detected.
VIDEO:  MPEG2  1280x720  (aspect 3)  29.970 fps  14455.2 kbps (1806.9 kbyte/s)
===========================================================================
Opening video decoder: [mpegpes] MPEG 1/2 Video passthrough
VDec: vo config request - 1280 x 720 (preferred colorspace: Mpeg PES)
Could not find matching colorspace - retrying with -vf scale...
Opening video filter: [scale]
The selected video_out device is incompatible with this codec.
Try appending the scale filter to your filter list,
e.g. -vf spp,scale instead of -vf spp.
VDecoder init failed :(
Opening video decoder: [ffmpeg] FFmpeg's libavcodec codec family
Unsupported PixelFormat -1
Selected video codec: [ffmpeg2] vfm: ffmpeg (FFmpeg MPEG-2)
===========================================================================
Audio: no sound
Starting playback...
VDec: vo config request - 1280 x 720 (preferred colorspace: Planar YV12)
Could not find matching colorspace - retrying with -vf scale...
Opening video filter: [scale]
VDec: using Planar YV12 as output csp (no 0)
Movie-Aspect is 1.78:1 - prescaling to correct movie aspect.
SwScaler: reducing / aligning filtersize 1 -> 4
SwScaler: reducing / aligning filtersize 1 -> 4
SwScaler: reducing / aligning filtersize 1 -> 1
SwScaler: reducing / aligning filtersize 5 -> 4
[swscaler @ 00F0D894]BICUBIC scaler, from yuv420p to yuyv422 using MMX2
[swscaler @ 00F0D894]using 4-tap MMX scaler for horizontal luminance scaling
[swscaler @ 00F0D894]using 4-tap MMX scaler for horizontal chrominance scaling
[swscaler @ 00F0D894]using n-tap MMX scaler for vertical scaling (BGR)
[swscaler @ 00F0D894]1280x720 -> 1280x720
VO: [directx] 1280x720 => 1280x720 Packed YUY2
New_Face failed. Maybe the font path is wrong.
Please supply the text font file (~/.mplayer/subfont.ttf).
subtitle font: load_sub_face failed.
New_Face failed. Maybe the font path is wrong.
Please supply the text font file (~/.mplayer/subfont.ttf).
subtitle font: load_sub_face failed.
V:  11.0 329/329 16%  1%  0.0% 0 0

BENCHMARKs: VC:   1.809s VO:   0.165s A:   0.000s Sys:   0.169s =   2.143s
BENCHMARK%: VC: 84.4144% VO:  7.6995% A:  0.0000% Sys:  7.8861% = 100.0000%
```

CPU mplayer = 2.143s

```
C:\Windows\system32\cmd.exe

=====================
NVIDIA CUDA decoding (novsync)
Running: cd "cudaDecodeGL\NVIDIA GPU Computing SDK\C\bin\win64\Release"
Running: cudaDecodeGL.exe --displayvideo
=====================

Press any key to continue . . .
[cudaDecodeGL.exe] starting...

[CUDA/OpenGL Video Decode]
[cudaDecodeGL]: input file: <../../../src/cudaDecodeGL/data/plush1_720p_10s.m2v>

        VideoCodec       : MPEG-2
        Frame rate       : 30000/1001fps ~ 29.97fps
        Sequence format  : Progressive
        Coded frame size: [1280, 720]
        Display area     : [0, 0, 1280, 720]
        Chroma format    : 4:2:0
        Bitrate          : 14116kBit/s
        Aspect ratio     : 16:9

> Device 0: < GeForce GT 555M >, Compute SM 2.1 detected
  -> GPU 0: < GeForce GT 555M > driver mode is: WDDM

>> initGL() creating window [1280 x 720]

argv[0] = cudaDecodeGL.exe
argv[1] = --displayvideo

> Using GPU Device: GeForce GT 555M has SM 2.1 compute capability
  Total amount of global memory:     3072.0000 MB
>> modInitCTX<NV12ToARGB_drvapi_x64.ptx > initialized OK
>> modGetCudaFunction< CUDA file:            NV12ToARGB_drvapi_x64.ptx >
   CUDA Kernel Function (0x08df4b90) = <   NV12ToARGB_drvapi >
>> modGetCudaFunction< CUDA file:            NV12ToARGB_drvapi_x64.ptx >
[cudaDecodeGL] - [Frame: 0160, 267.6 fps, frame time: 3.74 (ms) ]
[cudaDecodeGL] - [Frame: 0176, 272.1 fps, frame time: 3.67 (ms) ]
[cudaDecodeGL] - [Frame: 0192, 255.6 fps, frame time: 3.91 (ms) ]
[cudaDecodeGL] - [Frame: 0208, 251.8 fps, frame time: 3.97 (ms) ]
[cudaDecodeGL] - [Frame: 0224, 253.2 fps, frame time: 3.95 (ms) ]
[cudaDecodeGL] - [Frame: 0240, 251.5 fps, frame time: 3.98 (ms) ]
[cudaDecodeGL] - [Frame: 0256, 254.2 fps, frame time: 3.93 (ms) ]
[cudaDecodeGL] - [Frame: 0272, 267.3 fps, frame time: 3.74 (ms) ]
[cudaDecodeGL] - [Frame: 0288, 268.0 fps, frame time: 3.73 (ms) ]
[cudaDecodeGL] - [Frame: 0304, 257.2 fps, frame time: 3.89 (ms) ]

[cudaDecodeGL] statistics
        Video Length (hh:mm:ss.msec)    = 00:00:01.309
        Frames Presented (inc repeats) = 311
        Average Present Rate     (fps) = 237.54
        Frames Decoded   (hardware)    = 312
        Average Rate of Decoding (fps) = 238.30
```

GPU CUDA = 1.309s