GPU TECHNOLOGY CONFERENCE

NVIDIA Performance Primitives & Video Codecs on GPU

Gold Room | Thursday 1st October 2009 | Anton Obukhov & Frank Jargstorff

NVIDIA.

Overview

- Two presentations:
  - NPP (Frank Jargstorff)
  - Video Codes on NVIDIA GPUs (Anton Obukhov)
- NPP Overview
  - NPP Goals
  - How to use NPP?
  - What is in NPP?
  - Performance

What is NPP?

- C Library of functions (**primitives**) running on CUDA architecture
- API identical to IPP (Intel Integrated Performance Primitives)
- Speedups up to 32x over IPP
- Free distribution
  - binary packages for Windows and Linux (32- and 64 bit), Mac OS X
- Release Candidate 1.0: Available to Registered Developers now.
  - Final release in two weeks at http://www.nvidia.com/npp

GPU TECHNOLOGY CONFERENCE

NPP's Goals

- Ease of use
  - no knowledge of GPU architecture required
  - integrates well with existing projects
    - work well if added into existing projects
    - work well in conjunction with other libraries
- Runs on CUDA Architecture GPUs
- High Performance
  - relieve developers from optimization burden
- Algorithmic Building Blocks (Primitives)
  - recombine to solve wide range of problems

NVIDIA.

Ease of Use

- Implements Intel's IPP API verbatim
  - IPP widely used in high-performance software development
  - well designed API
- Uses CUDA "runtime API"
  - device memory is handled via simple C-style pointers
  - pointers in the NPP API are device pointers
  - but: host and device memory management left to user (for performance reasons)
- Pointer based API
  - pointers facilitate interoperability with existing code (C for CUDA) and libraries (cuFFT, cuBLAS, etc.)
  - imposes no "framework" on developers

GPU TECHNOLOGY CONFERENCE

Example

```
        // 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);
```


NVIDIA

What is in NPP?

- Only Image-Processing Functions
  - subset of “IPPI” library
  - ~300 functions
- Limited set of data-types supported
  - 8-bit per channel: 8u_C1, 8u_C4, 8u_AC4
  - high bit depth: 32s_C1, 32f_C1
- Conversion functions to and from most other IPPI formats

GPU TECHNOLOGY CONFERENCE

What is in NPP?

- Data exchange & initialization
  - Set, Convert, CopyConstBorder, Copy, Transpose, SwapChannels
- Arithmetic & Logical Ops
  - Add, Sub, Mul, Div, AbsDiff
- Threshold & Compare Ops
  - Threshold, Compare
- Color Conversion
  - RGB To YCbCr (& vice versa), ColorTwist, LUT_Linear
- JPEG
  - DCTQuantInv/Fwd, QuantizationTable
- Filter Functions
  - FilterBox, Row, Column, Max, Min, Dilate, Erode, SumWindowColumn/Row
- Geometry Transforms
  - Resize , Mirror, WarpAffine/Back/Quad, WarpPerspective/Back/Quad
- Statistics
  - Mean, StdDev, NormDiff, MinMax, Histogram, SqrIntegral, RectStdDev
- Computer Vision
  - Canny Edge Detector

NVIDIA.

Performance

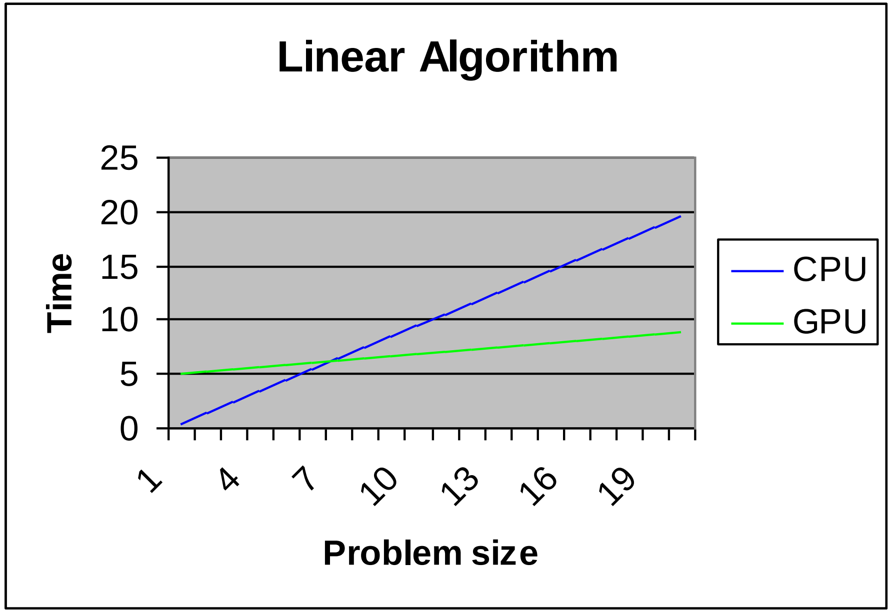
- Relative performance compared to IPP
  - Measuring methodology?
- Scalability
  - Problem size
  - Number of processor cores
- Some aggregated numbers
  - Performance suite averages

Performance Measuring Methodology

- Each primitive under test:
  - Is executed 25 times
  - Each iteration uses same data and same parameters
  - Data for GPU primitives is already on GPU (i.e. transfer times are not included in timings)
- All performance data gathered with single test application
  - test~2800 performance tests
  - most performance tests are simply repurposed functional tests
    - testing offset and oddly sized ROIs
    - testing various parameters
    - performance tests usually run at 720p and 2k x 2k image sizes

Scalability with Problem Size (1)

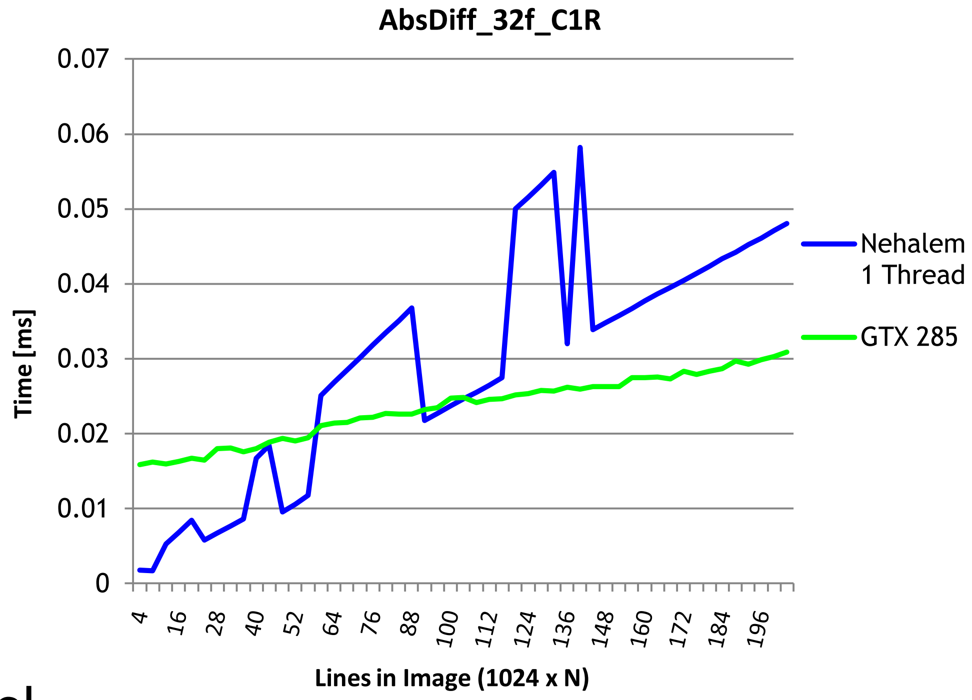
- Primtive: ippi/nppiAbsDiff_32f_C1R
  - computes per pixel absolute difference of two single-channel float image and stores result in third image
  - performance scales linearly with problem size
  - ***Time Plots: Lower is Better!***
- Expected Results
  - linear with offsets O_{CPU} & O_{GPU} and slopes S_{CPU} & S_{GPU}
  - $0 < O_{CPU} < O_{GPU}$
  - $S_{CPU} > S_{GPU} > 0$
- Where's the cross over point?



Scalability with Problem Size (2)

- Size from 1024x4 (16kB) to 1024x204 (~800kB)
- Offset & Slope:
  - CPU: O ~ 0 µs, S ~25 µs/100 lines
  - GPU: O ~ 15 µs, S ~10 µs/100 lines
- Crossover:
  - CPU slow:
    - 48 lines = 48kPixel (4Byte) = 192kB
  - CPU fast:
    - 108 line = 108kPixel (4Byte) = 432kB
  - Compare: 720p: 1280 x 720 = 900kPixel

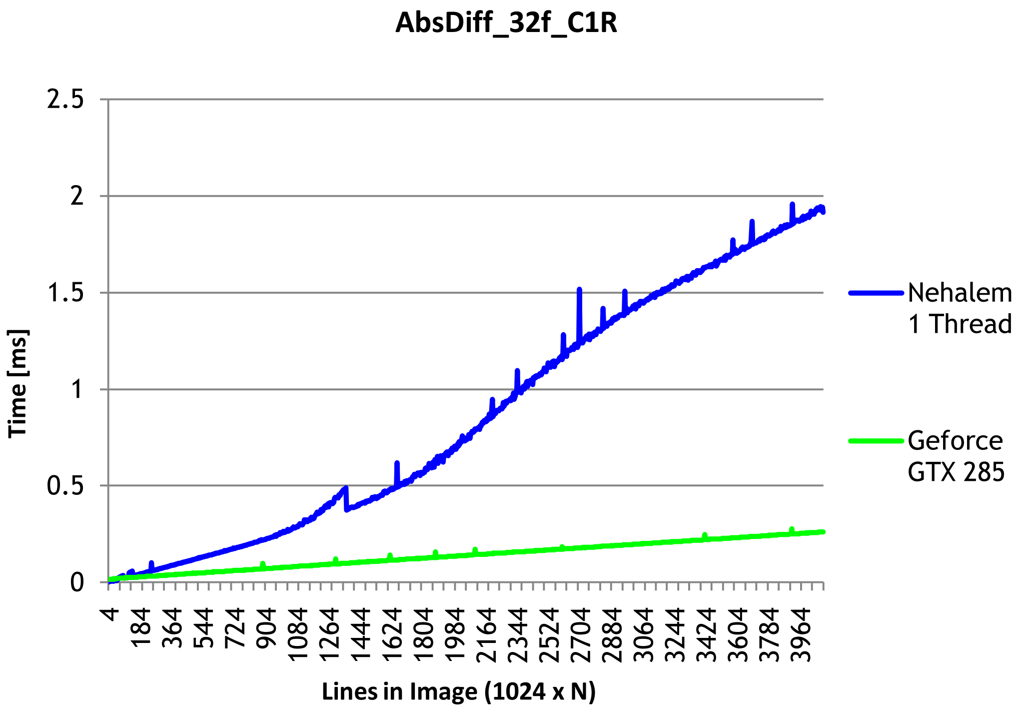
Intel Core i7 Extreme Edition i7-965
3.2 GHz, 4 (8) Core, 8MB Level 3 Cache



GPU TECHNOLOGY CONFERENCE

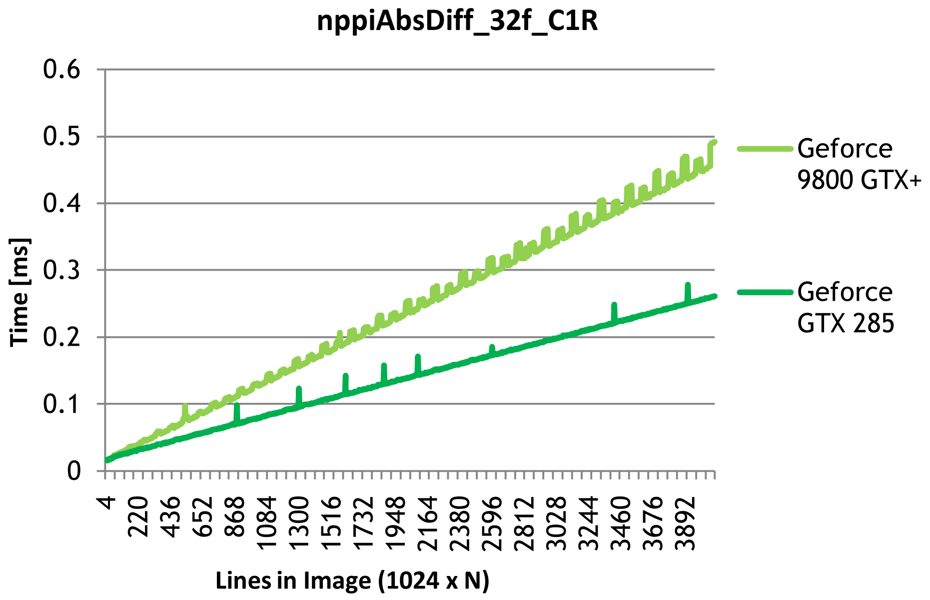
Scalability with Problem Size (3)

- Going in size up to 4096 lines
- GPU scales linearly
- Asymptotically CPU ~7.5x GPU
- CPU: Slope transition
  - Between ~1000 and ~3000 lines
  - 1000 lines = 4MByte image
  - 8MB level 3 cache size



Scalability With Number of Cores (1)

- For GPU not easy to control number of cores used.
- Compare two different GPUs/Graphics Cards:
  - Geforce 9800 GTX+: 16 SMs, 738MHz => 11808
  - Geforce GTX 285: 30 SMs, 648MHz => 19440
  - 19440/11808 = 1.64
- Chart at max size:
  - 9800 GTX: 480µs
  - GTX 285: 260µs
  - 4.8/2.6 = 1.84
- **GPU scales linearly with number of SMs (cores) across full range of problem sizes.**

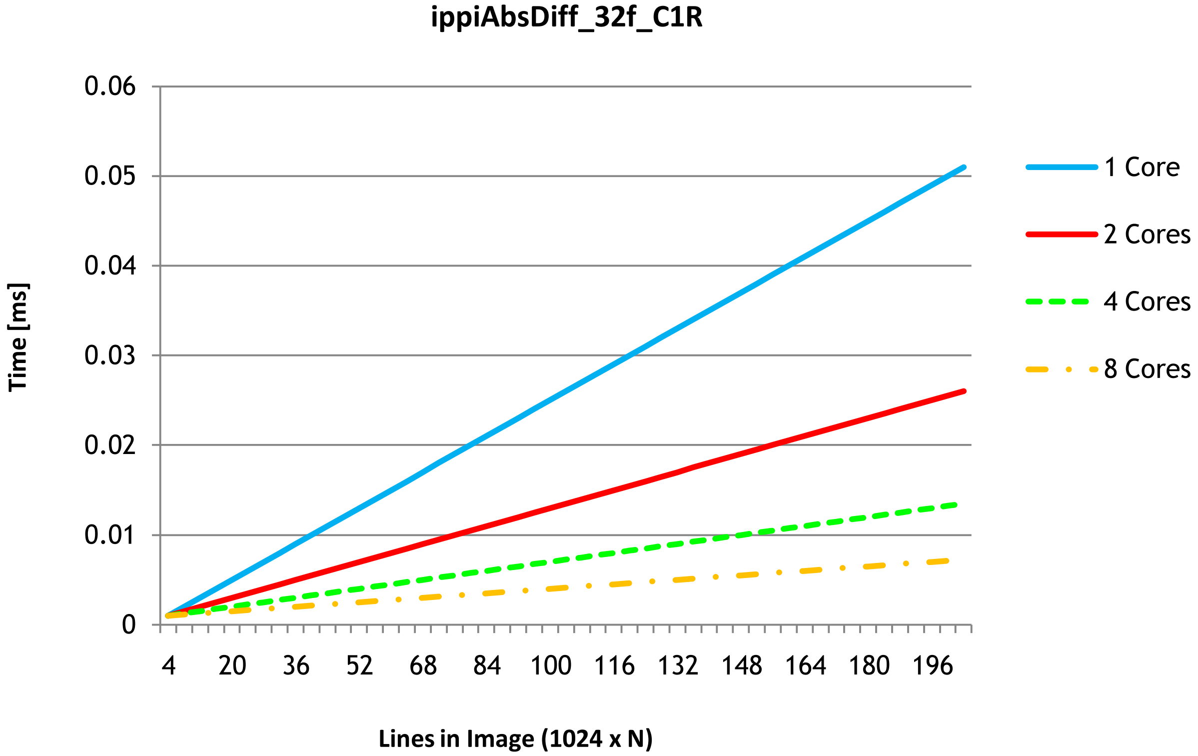


GPU TECHNOLOGY CONFERENCE

NVIDIA.

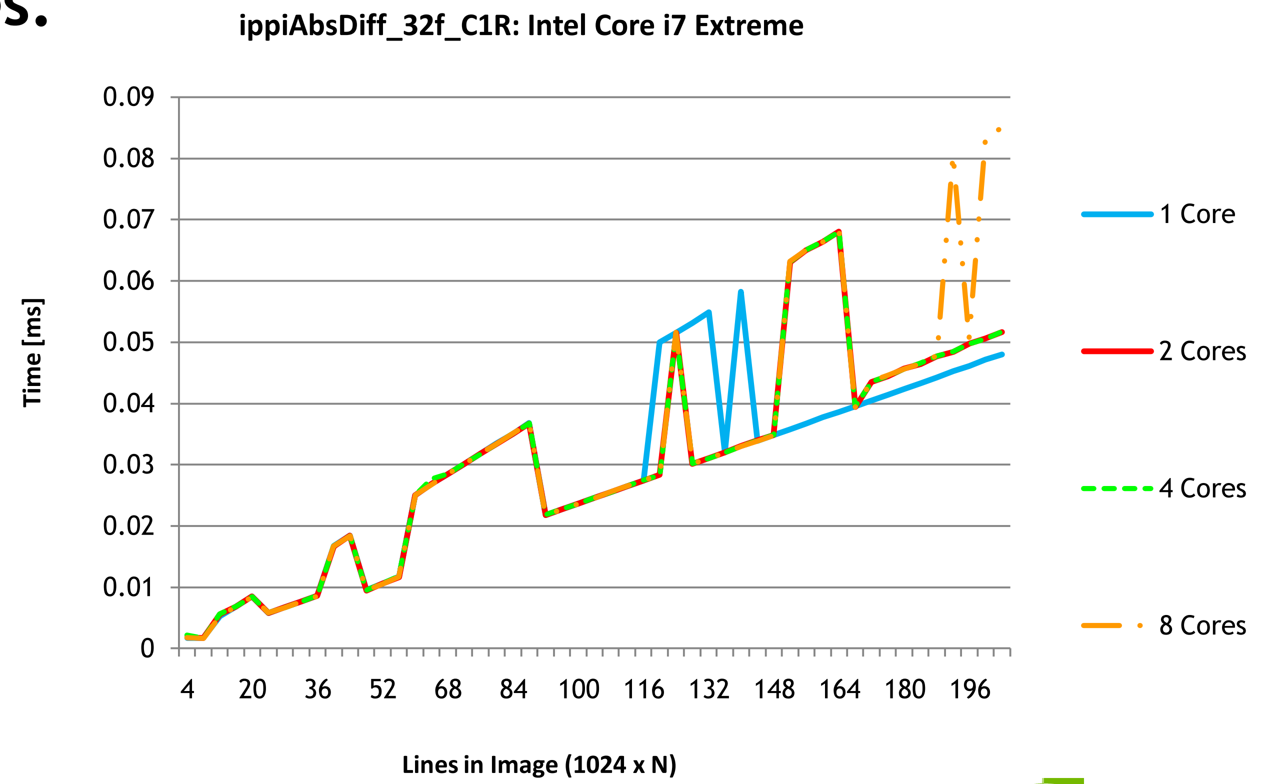
Scalability With Number of Cores (2)

- Use `ippSetNumThreads(int n);` to control number of cores used.
- Expected Result:

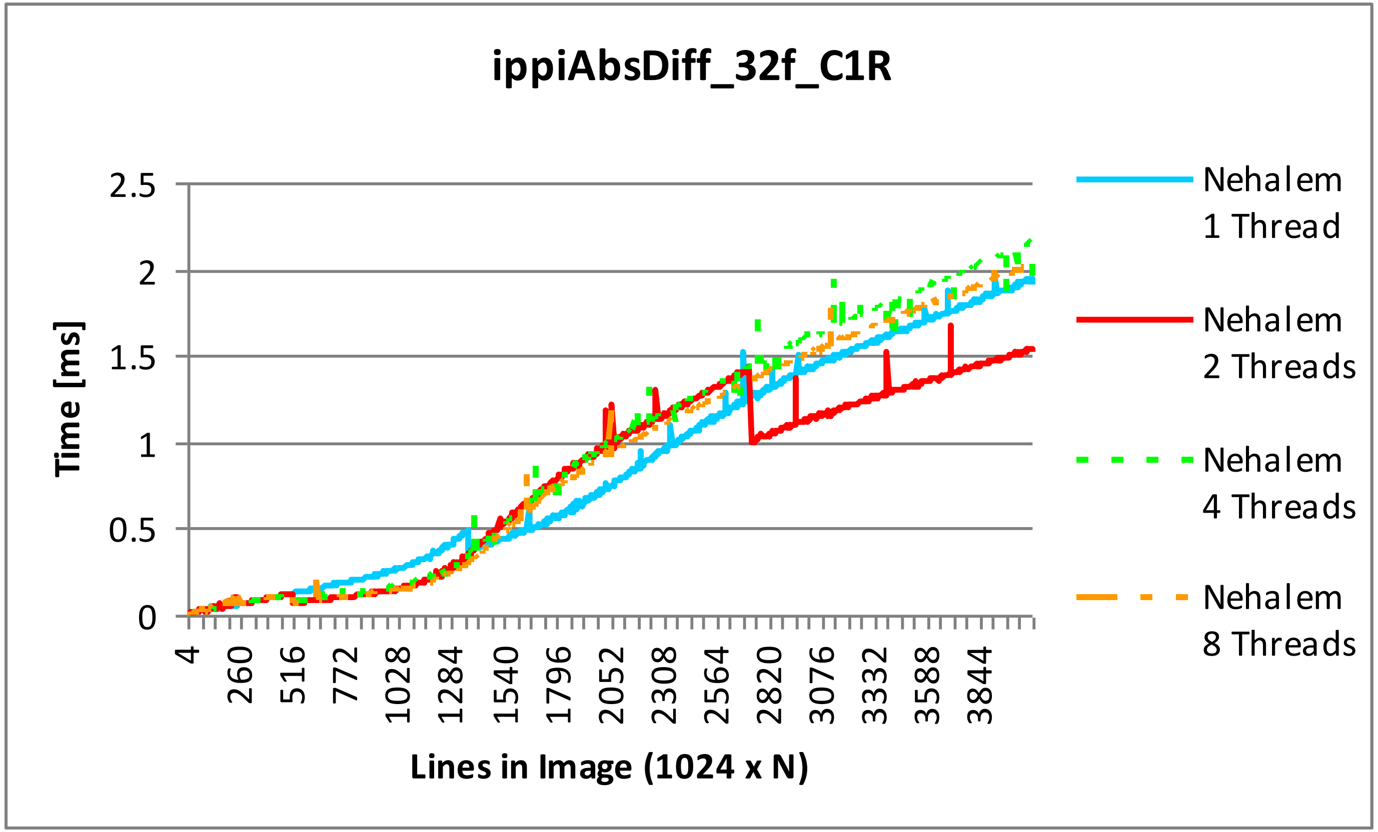


Scalability With Number of Cores (3)

- CPU performance does not scale with number of cores, even for small problem sizes.
- Actual Result:



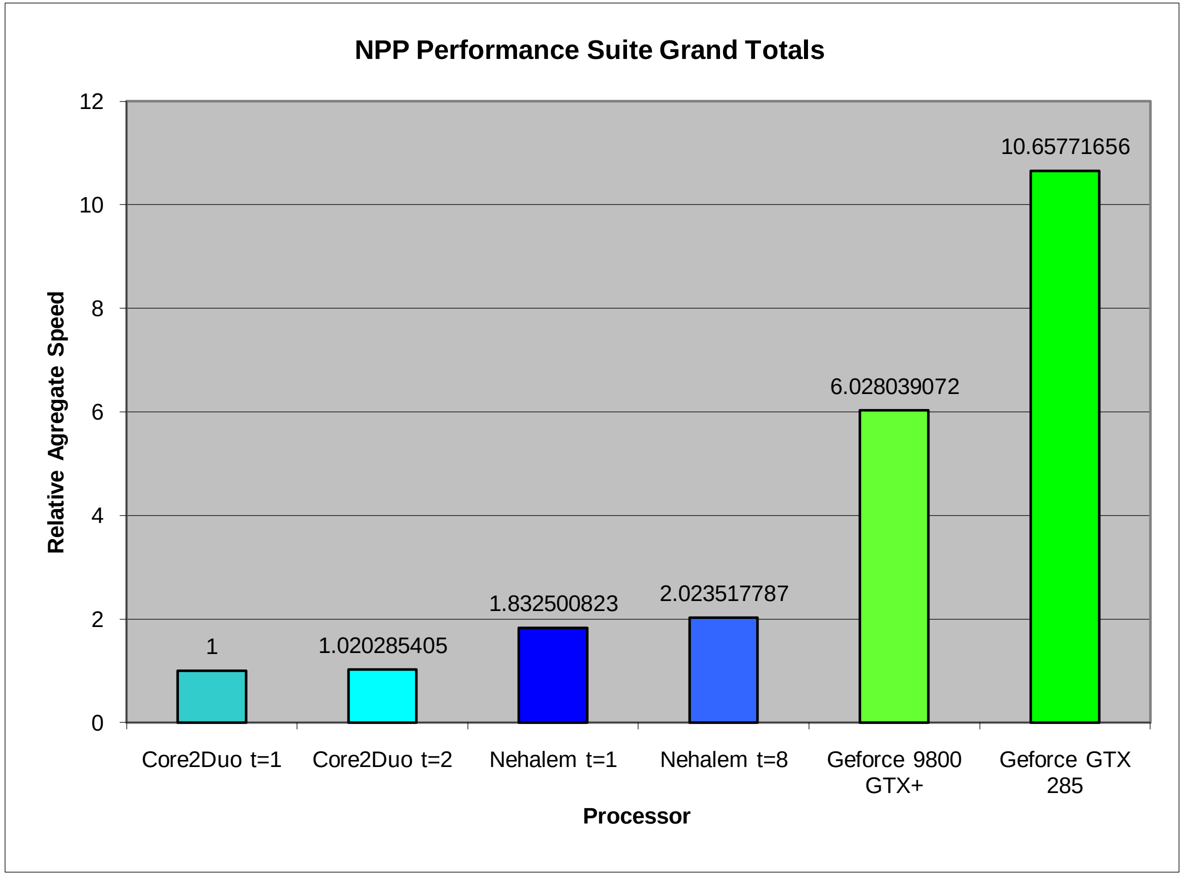
Scalability With Number of Cores (4)



- Full range of image sizes on CPU
  - Not clear how many threads are the best configuration for max performance.
- **CPU does not scale with number of Cores.**

Aggregate Performance Numbers (1)

- Average over 2800 performance tests.
  - each test gathers IPP and NPP processing times
  - performance tests are repurposed functional tests
  - run on 720p and 2k x 2k frames (mostly)



Aggregate Performance Numbers (2)

- Put into perspective:
  - NPP is 1.0 release
  - has been developed in 6 months
  - no processor specific optimizations*
    - all code compiled for compute 1.0 or 1.1
  - for the most part only optimized for memory coalescing
- Intel Core i7 vs. GTX 285
  - really different generations (GTX 285 uses 1.5 year old arch)
- That means there's still a lot of room for improvement.

* Exception: some statistics functions use atomics from compute capability 1.1.

Summary

- NPP
  - easy to integrate
  - provides substantial performance gains over highly optimized x86 code
  - 300 functions
- GPU/NPP Performance
  - scales extremely well with problem sizes and GPU type
  - room for performance improvements
  - particularly well suited for larger image sizes
- For questions regarding NPP please contact:
  - npp@nvidia.com

GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE

Video Codecs on GPU

Fairmont Hotel, San Jose | 10.01.2009 | Anton Obukhov

NVIDIA.

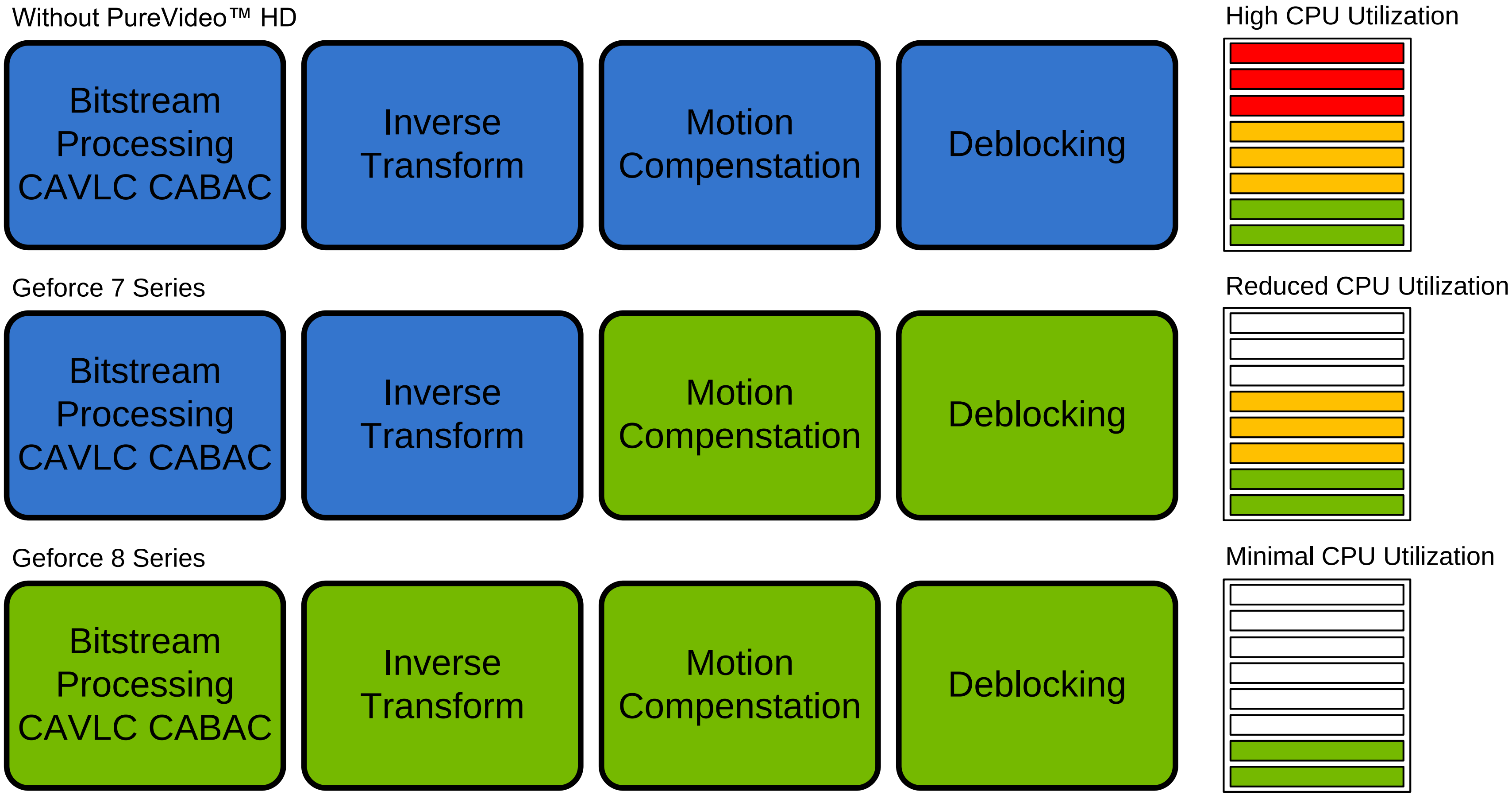
Motivation for the talk

Video encoding and decoding tasks require speedups as never before:

Encoding hi-res movie takes tens of hours on modern desktops

Portable and mobile devices have unveiled processing power

Video capabilities evolution



Video encoding with NVIDIA GPU

Facilities:

- SW H.264 codec designed for CUDA
  - Baseline profile
  - Main profile
  - High profile

Interfaces:

- C library (NVCUVENC)
- Direct Show API
- Win7 MFT

Video decoding with NVIDIA GPU

Facilities:

- HW GPU acceleration of
  - H.264
  - VC1
  - MPEG2
- SW MPEG2 decoder designed for CUDA

Interfaces:

- C library (NVCUVID), HW & SW
- DXVA and Win7 MFT, HW only
- VDPAU library, HW only

Video processing with NVIDIA GPU

Facilities:

- SW pre- and post-processing library designed for CUDA
  - Noise Reduction
  - Deinterlacing
  - Polyphase Scaling
  - Color Processing
  - Deblocking
  - Detail enhance

Interfaces:

- VMR/EVR API

Benefits of Decoding with GPU

~100% Offload of 3 Major Video CODECs

Without NVIDIA GPU — High CPU Utilization

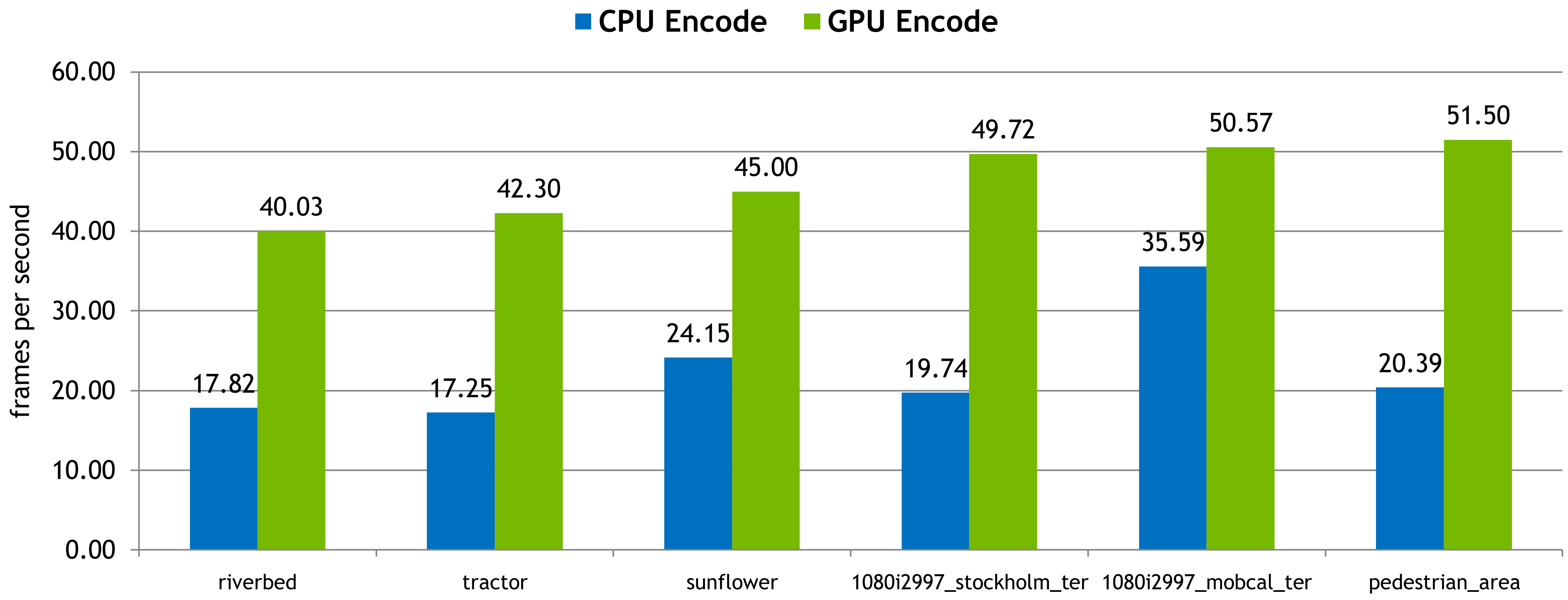
H.264	Bitstream Processing CAVLC CABAC	Inverse Transform	Motion Compensation	Deblocking
VC-1	Bitstream Processing CAVLC CABAC	Inverse Transform	Motion Compensation	Deblocking
MPEG-2	Bitstream Processing CAVLC CABAC	Inverse Transform	Motion Compensation	

With NVIDIA GPU — Minimal CPU Utilization

H.264	Bitstream Processing CAVLC CABAC	Inverse Transform	Motion Compensation	Deblocking
VC-1	Bitstream Processing CAVLC CABAC	Inverse Transform	Motion Compensation	Deblocking
MPEG-2	Bitstream Processing CAVLC CABAC	Inverse Transform	Motion Compensation	

GPU TECHNOLOGY CONFERENCE

Encode performance



Frame size: 1080p
Platform: 3.2 GHz quad core Nehalem, GeForce GTX 280 (240 core) GPU
CPU encoder is x264
GPU encoder is NVIDIA H.264 CUDA encoder.

Video encoding with NVIDIA GPU

Commercial applications for video transcoding with CUDA

- Badaboom
- Nero Move it
- CyberLink PowerDirector
- Loilo SuperLoiloscope
- *Tons of them!*

Thoughts aloud

- What about and ?
- What about multi-GPU systems?

Thoughts aloud

- What about and ?
  - Linux: only decoding acceleration with VDPAU
  - Mac OSX: QuickTime API

Thoughts aloud

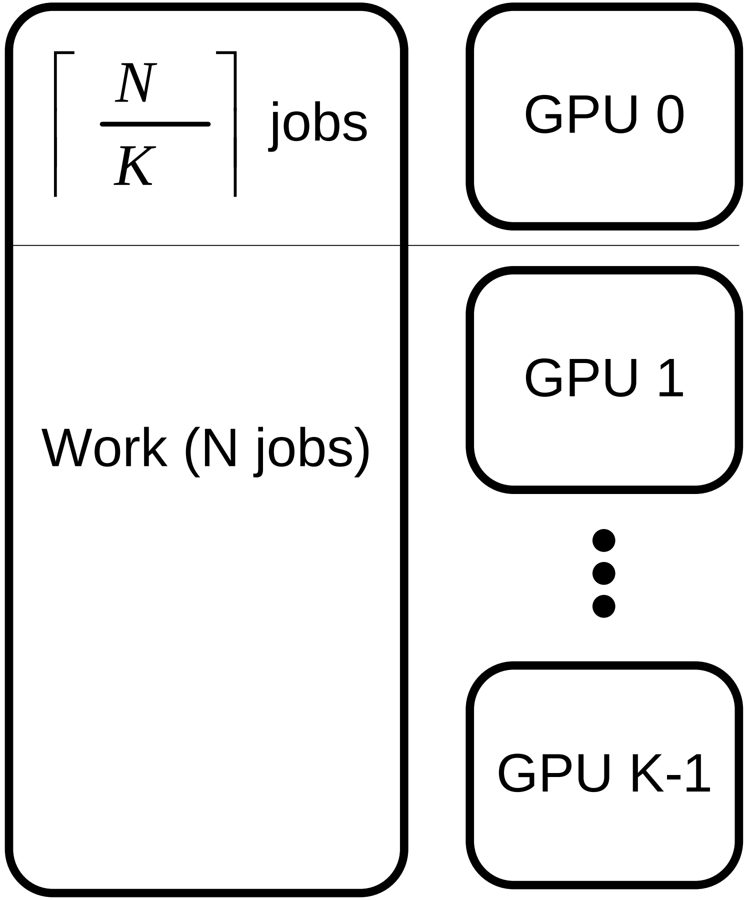
- What about multi-GPU systems?
  - NVIDIA H.264 encoder is going to support dual-GPU systems

Thoughts aloud

- Multi-GPU systems are commodity
- Programming for Multi-GPU systems is challenging

Thoughts aloud

CUDA provides access to every GPU. How to make them all work efficiently?



Thoughts aloud

There is a need for an open-source video codecs that can accelerate the transcoding pipeline using GPUs

Webinar 10/28/2009 9:00 AM - 11:00 AM PDT

- Multi-GPU techniques
- Application for video coding

https://www2.gotomeeting.com/register/628549827

GPU TECHNOLOGY CONFERENCE

Questions & Answers

?

E-mail: aobukhov@nvidia.com

“Introducing a new Multi-GPU framework” webinar, 10/28/2009 9:00 AM - 11:00 AM PDT

https://www2.gotomeeting.com/register/628549827