



# **NVIDIA Tools For Profiling And Monitoring**

**David Goodwin**



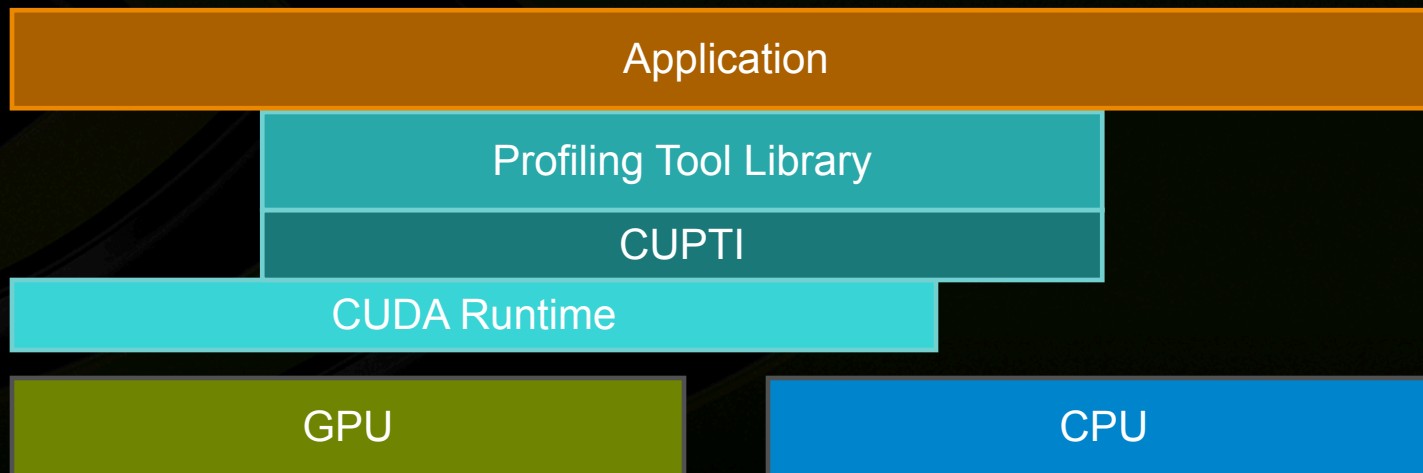
# Outline



- **CUDA Profiling and Monitoring**
  - Libraries
  - Tools
- **Technologies**
- **Directions**

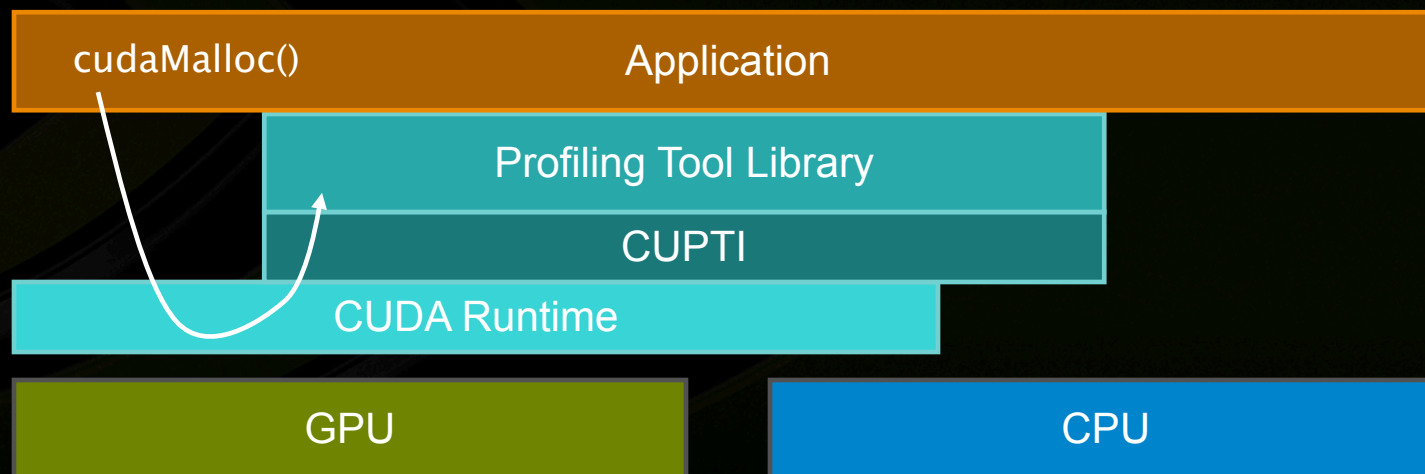
# CUDA Profiling Tools Interface (CUPTI)

- **CUPTI only supported interface for CUDA profiling**
  - 3<sup>rd</sup> party profilers: Vampir, Tau, PAPI, ...
  - Internal: NVIDIA Visual Profiler, nvprof



## CUPTI Callbacks

- Profiling tool can register for callback on every CUDA API call
  - Callback at entry and exit
  - Callback get access to function parameters and return code
- Also for CUDA object create/destroy, synchronization

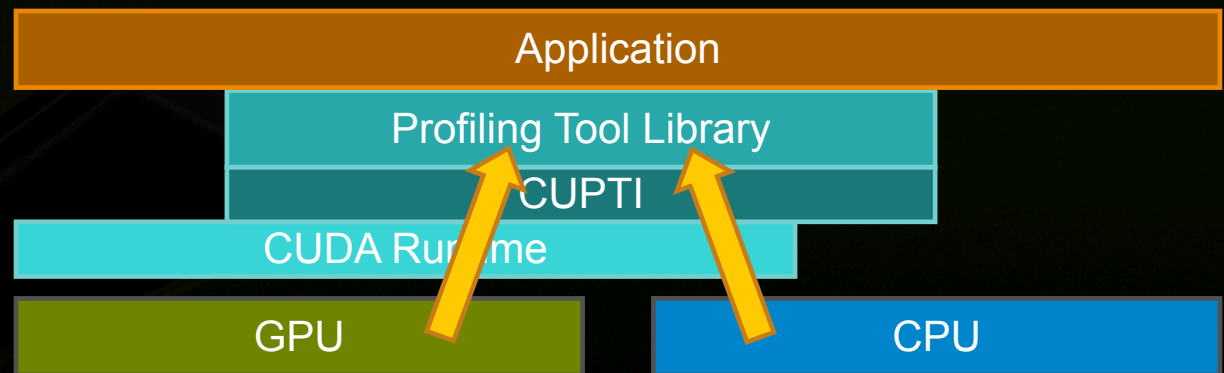


# CUPTI Events and Metrics

- **Events: low-level raw counts**
  - Sample
  - Collect over duration of kernel execution
  - HW counters
  - SW patches
  - Replay kernel execution to collect large amount of data
- **Metrics: typical profiling values**
  - E.g. IPC, throughput, hit-rate, etc.
  - Derived from event values and system properties
  - More actionable

# CUPTI Activity Trace

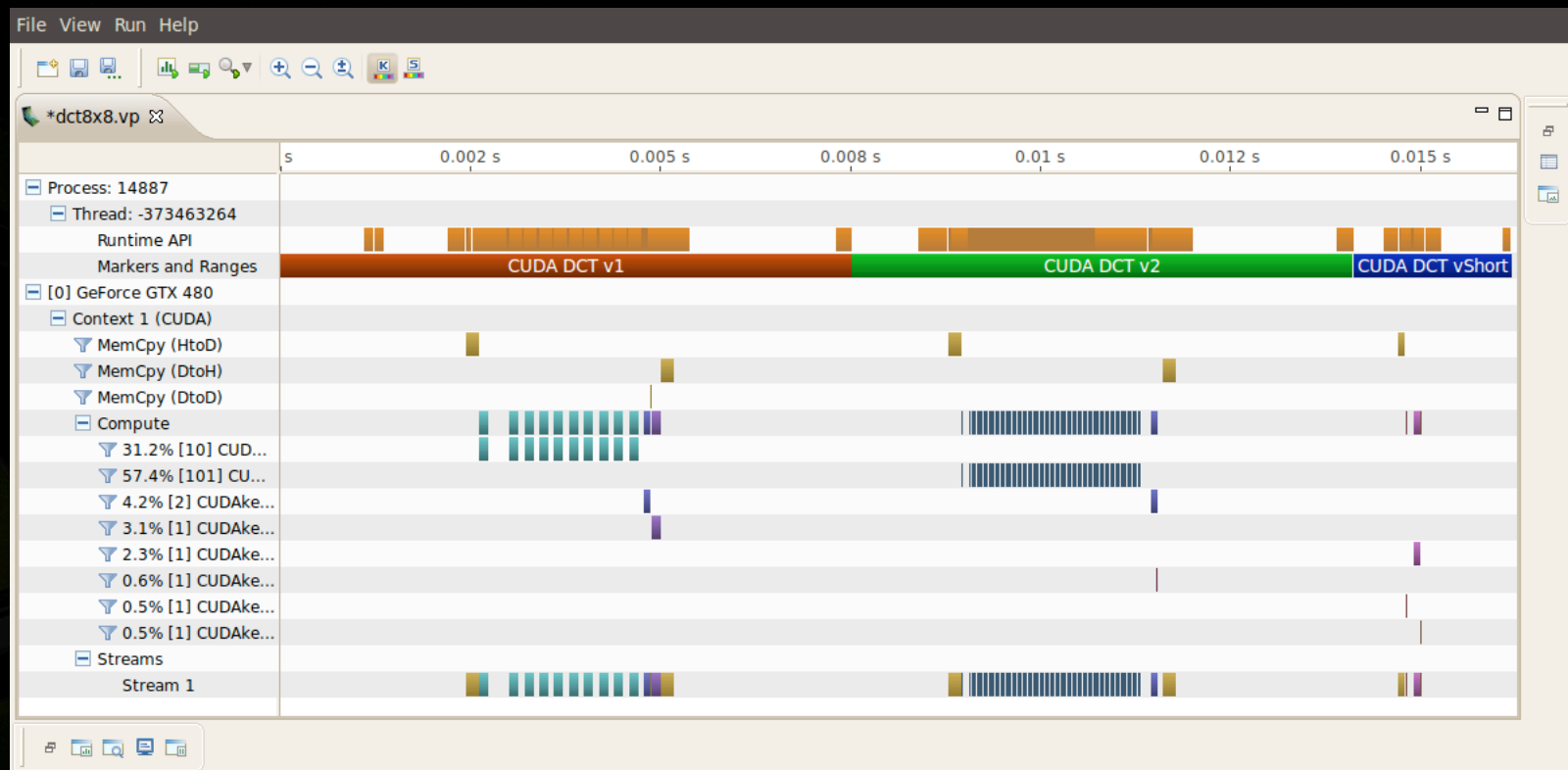
- **Deliver asynchronous stream of system activity**
  - GPU Kernel, memset, memcpy
  - CUDA API invocations
  - Developer-defined activity
  - **Dynamic, Per-PC activity**
    - Behavior of individual branches
    - Behavior of individual loads/stores



# NVIDIA Profiling Tools

- **Visual Profiler**
  - Graphical, Eclipse-based
  - Timeline
  - Automated Analysis
- **nvprof**
  - Command-line
  - Backend for Visual Profiler
  - Built on CUPTI

# Visual Profiler - Timeline





# Automated Analysis



**Low Memcpy Throughput [ 997.19 MB/s avg, for memcpys accounting for 68.1% of all memcpy time ]**  
The memory copies are not fully using the available host to device bandwidth. [More...](#)

- Actionable feedback
- Direct link to more extensive documentation
- Based on both trace and profile information

The screenshot shows a web browser window with a search bar at the top containing the text "Go Scope: All topics". Below the search bar is a navigation menu with the following items: "Visual Profiler Optimization", "Preface", "Parallel Computing with CU", "Performance Metrics", "Memory Optimizations", "Data Transfer Between Host and Device", "Pinned Memory", "Asynchronous Transfer", "Zero Copy", "Device Memory Spaces", "Allocation", "Execution Configuration Options", "Instruction Optimizations", "Control Flow", "Recommendations and Best Practices", and "NVIDIA Compiler Switches". The "Pinned Memory" item is selected and highlighted. The main content area displays the title "Pinned Memory" and the following text: "Page-locked or pinned memory transfers attain the highest bandwidth between the host and the device. On PCIe x16 Gen2 cards, for example, pinned memory can attain greater than 5 GBps transfer rates. Pinned memory is allocated using the cudaMallocHost() or cudaHostAlloc() functions in the Runtime API. The bandwidthTest.cu program in the CUDA SDK shows how to use these functions as well as how to measure memory transfer performance. Pinned memory should not be overused. Excessive use can reduce overall system performance because pinned memory is a scarce resource. How much is too much is difficult to tell in advance, so as with all optimizations, test the applications and the systems they run on for optimal performance parameters. Parent topic: [Data Transfer Between Host and Device](#)". At the bottom of the page, there is a copyright notice: "Copyright © 2011 NVIDIA Corporation | [www.nvidia.com](http://www.nvidia.com)" and the NVIDIA logo.

# Automated Analysis - Expert System

- **Identify bottlenecks**
- **Multiprocessor**
  - Compute, latency, memory bound
  - FU, register, etc. resource limiters
- **Memory subsystems**
  - Global, Shared
  - Texture, Constant
  - Caches

## Analysis Examples

- **“Occupancy can potentially be improved by increasing the number of threads per block.”**
- **“Global memory loads may have a bad access pattern, leading to inefficient use of global memory bandwidth.”**
- **“The kernel is likely memory bound, but it is not fully utilizing the available DRAM bandwidth.”**
- **“Divergent branches are causing significant instruction issue overhead.”**

- **Command-line interface to most CUPTI functionality**
- **Summary and full trace outputs**
- **Collect on headless node, visualize with Visual Profiler**

```
$ nvprof dct8x8
```

```
==== Profiling result:
```

| Time(%) | Time     | Calls | Avg      | Min      | Max      | Name                                |
|---------|----------|-------|----------|----------|----------|-------------------------------------|
| 49.52   | 9.36ms   | 101   | 92.68us  | 92.31us  | 94.31us  | CUDAkernel2DCT(float*, float*, int) |
| 37.47   | 7.08ms   | 10    | 708.31us | 707.99us | 708.50us | CUDAkernel1DCT(float*,int, int,int) |
| 3.75    | 708.42us | 1     | 708.42us | 708.42us | 708.42us | CUDAkernel1IDCT(float*,int,int,int) |
| 1.84    | 347.99us | 2     | 173.99us | 173.59us | 174.40us | CUDAkernelQuantizationFloat()       |
| 1.75    | 331.37us | 2     | 165.69us | 165.67us | 165.70us | [CUDA memcpy DtoH]                  |
| 1.41    | 266.70us | 2     | 133.35us | 89.70us  | 177.00us | [CUDA memcpy HtoD]                  |
| 1.00    | 189.64us | 1     | 189.64us | 189.64us | 189.64us | CUDAkernelShortDCT(short*, int)     |
| 0.94    | 176.87us | 1     | 176.87us | 176.87us | 176.87us | [CUDA memcpy HtoA]                  |

## Future (Profiling)

- **Tighter integration of CPU/GPU profiling**
- **Expand expert system**
  - “upward” – help identify and extract data parallelism
  - “downward” – more precise feedback
- **Auto tuning**
- **PC Sampling**
- **Expose expert system through CUPTI so 3<sup>rd</sup> party tools can take advantage**
- **Standards for higher-level profiling**
  - **OpenACC**

# Monitoring

- **Cluster, not individual nodes**
- **Coarser measurement and control**
- **Zero or little overhead**
  
- **NVIDIA Monitoring Library**
  - **In-band**
- **Proprietary bus protocol**
  - **Out-of-band**

# NVIDIA Monitoring Library (NVML)

- **GPU aggregate utilizations**
  - Compute
  - Bandwidth
  - Memory usage
- **Power**
  - Temperature
  - Clocks
  - Power draw
  - Power states

## Future (Monitoring)

- **Power management**
  - Improved control over clock speeds, power budget
  - Integration with power balancing solutions
  - Exploit variations in GPU power efficiencies
- **Improved profiling capabilities**
  - Support concurrent workloads (per-process reporting vs. per-GPU)
  - System level metrics: network bottlenecks, communication locality, ...
  - Higher accuracy utilization measurement
  - Cluster-level profiling expert system



## Future (Monitoring) – cont.

- In-band vs. out-of-band
- Proprietary vs. standard



**Questions?**