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
  - 3rd party profilers: Vampir, Tau, PAPI, ...
  - Internal: NVIDIA Visual Profiler, nvprof

Diagram:
- Application
- Profiling Tool Library
- CUPTI
- CUDA Runtime
- GPU
- CPU
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

```
cudaMalloc()
```

CUDA Runtime

- GPU
- CPU

Profiling Tool Library

- Profiling Tool Library
- CUPTI
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

CScADS Summer 2012 Workshop on Performance Tools for Extreme Scale Computing
Automated Analysis

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

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.

CScADS Summer 2012 Workshop on Performance Tools for Extreme Scale Computing
Automated Analysis - Expert System

- Identify bottlenecks

- Multiprocessor
  - Compute, latency, memory bound
  - FU, register, etc. resource limiters

- Memory subsytems
  - 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.”
nvprof

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

```bash
$ nvprof dct8x8
```

```
Profiling result:
<table>
<thead>
<tr>
<th>Time(%)</th>
<th>Time</th>
<th>Calls</th>
<th>Avg</th>
<th>Min</th>
<th>Max</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>49.52</td>
<td>9.36ms</td>
<td>101</td>
<td>92.68us</td>
<td>92.31us</td>
<td>94.31us</td>
<td>CUDAkernel2DCT(float*, float*, int)</td>
</tr>
<tr>
<td>37.47</td>
<td>7.08ms</td>
<td>10</td>
<td>708.31us</td>
<td>707.99us</td>
<td>708.50us</td>
<td>CUDAkernel1DCT(float*,int, int,int)</td>
</tr>
<tr>
<td>3.75</td>
<td>708.42us</td>
<td>1</td>
<td>708.42us</td>
<td>708.42us</td>
<td>708.42us</td>
<td>CUDAkernel1IDCT(float*,int,int,int)</td>
</tr>
<tr>
<td>1.84</td>
<td>347.99us</td>
<td>2</td>
<td>173.99us</td>
<td>173.59us</td>
<td>174.40us</td>
<td>CUDAkernelQuantizationFloat()</td>
</tr>
<tr>
<td>1.75</td>
<td>331.37us</td>
<td>2</td>
<td>165.69us</td>
<td>165.67us</td>
<td>165.70us</td>
<td>[CUDA memcpy DtoH]</td>
</tr>
<tr>
<td>1.41</td>
<td>266.70us</td>
<td>2</td>
<td>133.35us</td>
<td>89.70us</td>
<td>177.00us</td>
<td>[CUDA memcpy HtoD]</td>
</tr>
<tr>
<td>1.00</td>
<td>189.64us</td>
<td>1</td>
<td>189.64us</td>
<td>189.64us</td>
<td>189.64us</td>
<td>CUDAkernelShortDCT(short*, int)</td>
</tr>
<tr>
<td>0.94</td>
<td>176.87us</td>
<td>1</td>
<td>176.87us</td>
<td>176.87us</td>
<td>176.87us</td>
<td>[CUDA memcpy HtoA]</td>
</tr>
</tbody>
</table>
```
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 3rd 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?