OPTIMIZING APPLICATION PERFORMANCE WITH CUDA PROFILING TOOLS

Mayank Jain, 11 May 2017
• CUDA Profiling Tools
• Unified Memory Profiling
• NVLink Profiling
• PC Sampling
• MPI Profiling
• Multi-hop Remote Profiling
• Volta Support
• Other Improvements
CUDA PROFILING TOOLS

• NVIDIA® Visual Profiler 🐧_WINDOWS_APPLE

• nvprof 🐧_WINDOWS_APPLE_ANDROID *

• NVIDIA® Nsight™ Visual Studio Edition 🌐

* Android CUDA APK profiling not supported (yet)
3\textsuperscript{RD} PARTY PROFILING TOOLS

- TAU Performance System ®
- VampirTrace
- PAPI CUDA Component
- HPC Toolkit
UNIFIED MEMORY PROFILING
UNIFIED MEMORY EVENTS
SEGMENT MODE TIMELINE OPTIONS

Option to enable segment mode

Option to specify number of segments
SEGMENT MODE TIMELINE

Segment mode interval

Heat map for CPU page faults
SWITCH TO NON-SEGMENT VIEW

- Uncheck
- Select settings view
- Select this tab
- Load data within a specific time range
NON-SEGMENTED MODE TIMELINE
CPU PAGE FAULT SOURCE CORRELATION

Selected interval

Source location

Properties

CPU Page Faults

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>440.45958 ms (440,459,581 ns)</td>
</tr>
<tr>
<td>Memory Access Type</td>
<td>Write</td>
</tr>
<tr>
<td>Virtual Address</td>
<td>0x9001000000</td>
</tr>
<tr>
<td>Source Location</td>
<td><a href="mailto:main@Jacobi.cu">main@Jacobi.cu</a>:130</td>
</tr>
<tr>
<td>Process</td>
<td>25684</td>
</tr>
</tbody>
</table>
CPU PAGE FAULT SOURCE CORRELATION

Source line causing CPU page fault:

```plaintext
a[i] = a[i] - delta*i
```
CPU PAGE FAULT SOURCE CORRELATION

Unguided Analysis

Summary of all CPU page faults

Option to collect Unified Memory information
NEW UNIFIED MEMORY EVENTS

Page throttling, Memory thrashing, Remote map
FILTER AND ANALYZE

1. Select unified memory in the unguided analysis section

2. Select required events and click on ‘Filter and Analyze’

Summary of filtered intervals
FILTER AND ANALYZE

Unfiltered
**FILTER AND ANALYZE**

**Filtered**

<table>
<thead>
<tr>
<th>Process &quot;vecAdd_manager&quot;</th>
<th>233.5 ms</th>
<th>234 ms</th>
<th>234.5 ms</th>
<th>235 ms</th>
<th>235.5 ms</th>
<th>236 ms</th>
<th>236.5 ms</th>
<th>237 ms</th>
<th>237.5 ms</th>
<th>238 ms</th>
<th>238.5 ms</th>
<th>239 ms</th>
<th>239.5 ms</th>
<th>240 ms</th>
<th>240.5 ms</th>
<th>241 ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread 3890149184</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Runtime API</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Driver API</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Profiling Overhead</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unified Memory</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU Page Faults</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[0] Graphics Device</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unified Memory</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Migration [DtoH]</td>
<td></td>
<td>Data ...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GPU Page Faults</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Migration [HtoD]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Context 1 (CUDA)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100.0% vectorAdd...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Streams</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CPU Page Faults**
- **Access Type:**
  - [ ] Read
  - [ ] Write

**GPU Page Faults**
- **Access Type:**
  - [ ] Read
  - [ ] Write
  - [ ] Atomic
  - [ ] Prefetch

**HtoD Migrations**
- **Reason:**
  - [ ] User
  - [ ] Coherence
  - [ ] Prefetch

**DtoH Migrations**
- **Reason:**
  - [ ] User
  - [ ] Coherence
  - [ ] Prefetch
  - [ ] Eviction

**Filtered intervals**
UNOPTIMIZED APPLICATION

Memory Thrashing

12.2 ms

Read access page faults

Analyze read access page faults and thrashing
OPTIMIZATION

OLD

```c
int threadsPerBlock = 256;
int numBlocks = (length + threadsPerBlock - 1) / threadsPerBlock;

kernel<<<numBlocks, threadsPerBlock>>>(A, B, C, length);
```

NEW

```c
int threadsPerBlock = 256;
int numBlocks = (length + threadsPerBlock - 1) / threadsPerBlock;

cudaMemAdvise(A, size, cudaMemAdviseSetReadMostly, 0);
cudaMemAdvise(B, size, cudaMemAdviseSetReadMostly, 0);

kernel<<<numBlocks, threadsPerBlock>>>(A, B, C, length);
```
OPTIMIZED APPLICATION

No DtoH Migrations and thrashing

Speedup 4x (2.9 vs 12.2)
NVLINK VISUALIZATION

Visual Profiler

Unguided Analysis

Option to collect NVLink information

Color codes for NVLink

Topology

Selected NVLink

Static properties

Runtime values

NVLink Analysis

The following NVLink topology diagram shows logical NVLink connections between GPUs and CPUs. A logical NVLink can contain one or more physical links. When two devices A and B are connected by an NVLink, the receive throughput of device A is the same as the transmit throughput of device B. The tables on the right-hand side show the properties for each logical NVLink.

*NVLink utilization may vary in accuracy, because any activity within the sampling period is treated as active, even though most of that period could be idle.

Achieved throughput

Selected NVLink
DGX-1V NVLINK TOPOLOGY
NVLink Events on Timeline

MemCpy API

Color Coding of NVLink Events
NVLINK ANALYSIS

Stage I: Data Movement Over PCIe

216 milliseconds
# NVLINK ANALYSIS

## Stage II: Data Movement Over NVLink

<table>
<thead>
<tr>
<th>Context 4 (CUDA)</th>
<th>Compute</th>
<th>Streams</th>
<th>Default</th>
</tr>
</thead>
</table>

### Speedup 4x (216 vs 65)

- Under-utilized NVLink
- Minimal-Unused NVLinks

### 65 milliseconds

<table>
<thead>
<tr>
<th>5.71s</th>
<th>5.72s</th>
<th>5.73s</th>
<th>5.75s</th>
<th>5.76s</th>
<th>5.77s</th>
<th>5.78s</th>
</tr>
</thead>
<tbody>
<tr>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
</tr>
<tr>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
</tr>
<tr>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
</tr>
<tr>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
</tr>
<tr>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
</tr>
<tr>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
</tr>
<tr>
<td>174GBs</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
<td>0B/s</td>
</tr>
</tbody>
</table>

**Logical NVLink Throughput**

| GPU0 -> CPU0 | GPU0 < CPU0 | GPU0 < GPU1 | GPU0 < GPU2 | GPU0 < GPU3 | GPU1 -> CPU0 | GPU1 < CPU0 | GPU1 < GPU2 | GPU1 < GPU3 | GPU2 -> CPU0 | GPU2 < CPU0 | GPU2 < GPU3 | GPU3 -> CPU0 | GPU3 < CPU0 | GPU3 < GPU1 | GPU3 < GPU2 | GPU3 < GPU3 |
|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       |
| 816.309 KB/s | 372.575 KB/s | 29.14 KB/s  | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       |
| 64.750 KB/s  | 29.14 KB/s  | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       |
| 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       |
| 777.593 KB/s | 351.521 KB/s | 31.789 KB/s | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       |
| 79.474 KB/s  | 31.789 KB/s | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       |
| 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       |
| 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       |
| 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       |
| 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       |
| 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       |
| 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       |
| 1.71 GB/s   | 1.71 GB/s   | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       |
| 26.606 KB/s  | 26.606 KB/s | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       |
| 11.402 KB/s  | 11.402 KB/s | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       | 0 B/s       |
NVLINK ANALYSIS
Stage III: Data Movement Over NVLink with Streams

20 milliseconds

Speedup 3x (65 vs 20)

Changed color based on selection
INSTRUCTION LEVEL PROFILING (PC SAMPLING)
PC SAMPLING

PC sampling feature is available for device with CC >= 5.2

Provides CPU PC sampling parity + additional information for warp states/stalls reasons for GPU kernels

Effective in optimizing large kernels, pinpoints performance bottlenecks at specific lines in source code or assembly instructions

Samples warp states periodically in round robin order over all active warps

No overheads in kernel runtime, CPU overheads to parse the records
PC SAMPLING UI

Pie chart for sample distribution for a CUDA function

Sample distribution

Source-Assembly view
MPI PROFILING
MPI PROFILING

nvprof

$ mpirun -n 4 nvprof --process-name "MPI Rank %q{OMPI_COMM_WORLD_RANK}" --context-name "MPI Rank %q{OMPI_COMM_WORLD_RANK}" -o timeline.%q{OMPI_COMM_WORLD_RANK}.pdm ./simpleMPI

Running on 4 nodes
==21977== NVPROF is profiling process 21977, command: ./simpleMPI
==21983== NVPROF is profiling process 21983, command: ./simpleMPI
==21979== NVPROF is profiling process 21979, command: ./simpleMPI
==21982== NVPROF is profiling process 21982, command: ./simpleMPI

<program output>
==21982== Generated result file: timeline.0.pdm
==21977== Generated result file: timeline.3.pdm
==21983== Generated result file: timeline.1.pdm
==21979== Generated result file: timeline.2.pdm
MPI PROFILING
nvprof daemon mode

1. $ nvprof --profile-all-processes
   -o out.%h.%p.%q{OMPI_COMM_WORLD_RANK}
   <nvprof listens in daemon mode>

2. $ mpirun -n 4 ./simpleMPI

3. <profiling data is generated>
MPI PROFILING
Importing into the Visual Profiler

1. Open a new session in the Visual Profiler.

2. Click on "Import..." to import the profile data.

3. Select "Multiple processes" from the available options.

4. Choose the profile files containing the timeline data for multiple processes.
MPI PROFILING

Visual Profiler

MPI Rank-based naming

NVTX Markers & Ranges
MPI PROFILING
MPI + NVTX

Manual mode

nvtxEventAttributes_t range = {0};
rang.message.ascii = "MPI_Scatter";
nvtxRangePushEx(range);
int result = MPI_Scatter(...);
nvtxRangePop();

Interception mode

1. Auto-generate mpi_interception.so

2. LD_PRELOAD=mpi_interception.so

3. Run your MPI app with nvprof.
MPI calls will be auto-annotated using NVTX.
MPI PROFILING

Interception

int res = MPI_Scatter(...);


MPI app

Interception library (LD_PRELOAD)

int MPI_Scatter(...)
{
    nvtxRangePushEx(range);
    int res = PMPI_Scatter(...);
    nvtxRangePop();
    nvtxRangePop();
    return res;
}

MPI library

int PMPI_Scatter(...)

REMOTE PROFILING
NVVP: MULTI-HOP REMOTE PROFILING

Visual Profiler → ssh → Login Node → ssh → CUDA Application

Host

Script

Login Node

Compute Node
NVVP: MULTI-HOP REMOTE PROFILING

One-Time Setup

1. Configure script on the login node

   ![Script]
   ![Login Node]

2. Connect Visual Profiler to the login node

   ![Remote Connections]

3. Use the custom script option

   ![CUDA Toolkits on tk@10.24.204.242]

   ![Custom Script]
NVVP: MULTI-HOP REMOTE PROFILING

Application Profiling

1. Select custom script, then create a remote session as usual

2. Application transparently runs on compute node and profiling data is displayed in the Visual Profiler
VOLTA SUPPORT
VOLTA SUPPORT

GPU Trace

GV100 Device Attributes
OTHER IMPROVEMENTS
OTHER IMPROVEMENTS

Tracing and profiling of Cooperative Kernel launches is supported.

The Visual Profiler supports remote profiling to systems supporting ssh algorithms with a key length of 2048 bits.

OpenACC profiling is now supported on systems without CUDA setup.

nvprof flushes all profiling data when a SIGINT or SIGKILL signal is encountered.
REFERENCES

NVIDIA toolkit documentation: http://docs.nvidia.com


Other GTC 2017 sessions:

S7824 - DEVELOPER TOOLS UPDATE IN CUDA 9
S7519 - DEVELOPER TOOLS FOR AUTOMOTIVE, DRONES AND INTELLIGENT CAMERAS APPLICATIONS
S7445 - WHAT THE PROFILER IS TELLING YOU: OPTIMIZING WHOLE APPLICATION PERFORMANCE
S7444 - WHAT THE PROFILER IS TELLING YOU: OPTIMIZING GPU KERNELS