BEST PRACTICES WHEN BENCHMARKING CUDA APPLICATIONS

Bill Fiser - Senior System Software Engineer
Sebastian Jodłowski - Senior System Software Engineer
AGENDA

Peak performance vs. Stable performance
AGENDA

Peak performance vs. Stable performance
AGENDA

System stability
  • CPU Frequency Scaling
  • NUMA
  • GPU clocks

Measuring the right thing
  • JIT cache
  • CUDA events
  • API contention
SYSTEM STABILITY
#include <chrono>
#include <iostream>

using namespace std;
using namespace std::chrono;

__global__ void empty() {}

int main() {
    const int iters = 1000;

    cudaFree(0);
    empty<<<1,1>>>(0);
    cudaDeviceSynchronize();

    // Warmup phase
    for (int i = 0; i < 10; ++i) {
        empty<<<1,1>>>(0);
    }

    // Benchmark phase
    auto start = steady_clock::now();
    for (int i = 0; i < iters; ++i) {
        empty<<<1,1>>>(0);
    }
    auto end = steady_clock::now();

    auto usecs = duration_cast<duration<float, microseconds::period>>(end - start);
    cout << usecs.count() / iters << endl;
}
CPU FREQUENCY SCALING
Achieving Stable CPU Benchmarks: launch latency

Average Launch Latency - 2.70 us
Relative Standard Deviation - 16%

DGX-1V, Intel Xeon E5-2698 @ 2.20GHz
CPU FREQUENCY SCALING
Achieving Stable CPU Benchmarks: launch latency

CPU clocks can fluctuate significantly
- This can be a result of CPU idling
- This can be a result of thermal or power throttling
- Can potentially cause unstable benchmark results

Average Launch Latency - 2.70 us
Relative Standard Deviation - 16%

DGX-1V, Intel Xeon E5-2698 @ 2.20GHz
CPU FREQUENCY SCALING

Monitoring Clocks and Policies

Using cpupower to monitor clocks while the test is running can reveal what is happening

user@dgx-1v:~$ cpupower monitor -m Mperf

<table>
<thead>
<tr>
<th>PKG</th>
<th>CORE</th>
<th>CPU</th>
<th>C0</th>
<th>Cx</th>
<th>Freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>99.13</td>
<td>0.87</td>
<td>3575</td>
</tr>
<tr>
<td>0</td>
<td>40</td>
<td>0</td>
<td>0.07</td>
<td>99.93</td>
<td>3360</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>9.64</td>
<td>90.36</td>
<td>3568</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>41</td>
<td>41.55</td>
<td>58.45</td>
<td>3576</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0.05</td>
<td>99.95</td>
<td>2778</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>42</td>
<td>0.14</td>
<td>99.86</td>
<td>3249</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0.06</td>
<td>99.94</td>
<td>2789</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>43</td>
<td>0.07</td>
<td>99.93</td>
<td>2835</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0.07</td>
<td>99.93</td>
<td>2867</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>44</td>
<td>0.06</td>
<td>99.94</td>
<td>2912</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>5</td>
<td>0.05</td>
<td>99.95</td>
<td>2793</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>45</td>
<td>0.07</td>
<td>99.93</td>
<td>2905</td>
</tr>
</tbody>
</table>

user@dgx-1v:~$ cpupower frequency-info

analyzing CPU 0:

driver: intel_pstate

CPUs which run at the same hardware frequency: 0
CPUs which need to have their frequency coordinated by software: 0
maximum transition latency: Cannot determine or is not supported.
hardware limits: 1.20 GHz - 3.60 GHz
available cpufreq governors: performance powersave
current policy: frequency should be within 1.20 GHz and 3.60 GHz.
The governor "powersave" may decide which speed to use within this range.
current CPU frequency: Unable to call hardware
current CPU frequency: 1.31 GHz (asserted by call to kernel)
boost state support:
  Supported: yes
  Active: yes
CPU FREQUENCY SCALING

Monitoring Clocks and Policies

CPU frequency scaling enables the operating system to scale the CPU frequency up or down in order to increase performance or save power.

Scaling Governor set to “powersave” can result in CPU being underclocked longer than expected.

Turbo Boost set to enabled can result in CPU being overclocked and eventually throttle.

user@dgx-1v:~$ cpupower frequency-info
analyzing CPU 0:
   driver: intel_pstate
   CPUs which run at the same hardware frequency: 0
   CPUs which need to have their frequency coordinated by software: 0
   maximum transition latency: Cannot determine or is not supported.
   hardware limits: 1.20 GHz - 3.60 GHz
   available cpufreq governors: performance powersave
   current policy: frequency should be within 1.20 GHz and 3.60 GHz. The governor "powersave" may decide which speed to use within this range.
   current CPU frequency: Unable to call hardware
   current CPU frequency: 1.31 GHz (asserted by call to kernel)
   boost state support: Supported: yes
   Active: yes
CPU FREQUENCY SCALING
Achieving Stable CPU Benchmarks

With intel_pstate driver user cannot directly control CPU clocks

Use “performance” scaling governor and disable Turbo Boost for more stable benchmarking

user@d gx-1v:~$ # Set the Frequency Scaling Governor to Performance
user@d gx-1v:~$ sudo cpupower frequency-set -g performance
Setting cpu: 0
...
Setting cpu: 79
user@d gx-1v:~$ # Disable Turbo Boost
user@d gx-1v:~$ echo "1" | sudo tee /sys/devices/system/cpu/intel_pstate/no_turbo
1
CPU FREQUENCY SCALING

Achieving Stable CPU Benchmarks

This helps keeping CPU clocks in more stable state

---

<table>
<thead>
<tr>
<th>PKG</th>
<th>CORE</th>
<th>CPU</th>
<th>C0</th>
<th>Cx</th>
<th>Freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>93.43</td>
<td>6.57</td>
<td>2192</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.75</td>
<td>99.25</td>
<td>2185</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>41</td>
<td>0.60</td>
<td>99.40</td>
<td>2193</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2.71</td>
<td>97.29</td>
<td>2192</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0.52</td>
<td>99.48</td>
<td>2193</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0.46</td>
<td>99.54</td>
<td>2193</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>44</td>
<td>0.56</td>
<td>99.44</td>
<td>2186</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>5</td>
<td>0.48</td>
<td>99.52</td>
<td>2193</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>45</td>
<td>0.54</td>
<td>99.46</td>
<td>2193</td>
</tr>
</tbody>
</table>

user@dgx-1v:~$ cpupower monitor -m Mperf

user@dgx-1v:~$ cpupower frequency-info

analyzing CPU 0:

driver: intel_pstate

CPUs which run at the same hardware frequency: 0
CPUs which need to have their frequency coordinated by software: 0
maximum transition latency: Cannot determine or is not supported.
hardware limits: 1.20 GHz - 3.60 GHz
available cpufreq governors: performance powersave
current policy: frequency should be within 1.20 GHz and 2.20 GHz.
   The governor "performance" may decide which speed to use within this range.
current CPU frequency: Unable to call hardware
current CPU frequency: 2.19 GHz (asserted by call to kernel)
boost state support:
   Supported: yes
   Active: yes
CPU FREQUENCY SCALING

Achieving Stable CPU Benchmarks: launch latency

Average Launch Latency - 2.61 us
Relative Standard Deviation - 3%

Better stability with “performance” scaling governor

DGX-1V, Intel Xeon E5-2698 @ 2.20GHz
NUMA
Achieving Stable Memory Benchmarks: pageable copies

Host-to-device pageable memcopy:
Average Bandwidth - 4.5 GB/s
Relative Standard Deviation - 1%

Device-to-host pageable memcopy:
Average Bandwidth - 6.1 GB/s
Relative Standard Deviation - 15%
Achieving Stable Memory Benchmarks: pageable copies

Low or unstable bandwidth might be caused by CPU migrations or accesses to non-local memory.

Host-to-device pageable memcopy:
Average Bandwidth - 4.5 GB/s
Relative Standard Deviation - 1%

Device-to-host pageable memcopy:
Average Bandwidth - 6.1 GB/s
Relative Standard Deviation - 15%

DGX-1V, Intel Xeon E5-2698 @ 2.20GHz
Non-Uniform Memory Access (NUMA) allows system memory to be divided into zones (nodes).

NUMA nodes are allocated to particular CPUs or sockets.

Memory bandwidth and latencies between NUMA nodes might not be the same.
Non-Uniform Memory Access (NUMA) allows system memory to be divided into zones (nodes). NUMA nodes are allocated to particular CPUs or sockets. Memory bandwidth and latencies between NUMA nodes might not be the same.
Non-Uniform Memory Access (NUMA) allows system memory to be divided into zones (nodes).

NUMA nodes are allocated to particular CPUs or sockets.

Memory bandwidth and latencies between NUMA nodes might not be the same.
Non-Uniform Memory Access (NUMA) allows system memory to be divided into zones (nodes).

NUMA nodes are allocated to particular CPUs or sockets.

Memory bandwidth and latencies between NUMA nodes might not be the same.
NUMA
Querying NUMA configuration

Use numactl to check NUMA nodes configuration

```bash
user@dgx-1v:~$ numactl --hardware
available: 2 nodes (0-1)
node 0 cpus:   0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59
node 0 size: 257844 MB
node 0 free:  255674 MB
node 1 cpus:  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  74  75  76  77  78  79
node 1 size: 258039 MB
node 1 free:  256220 MB
node distances:
  node   0   1
  0:    10  21
  1:    21  10
```
Use numactl to check NUMA nodes configuration

```
user@dgx-1v:~$ numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59
node 0 size: 257844 MB
node 0 free: 255674 MB
node 1 cpus: 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79
node 1 size: 258039 MB
node 1 free: 256220 MB
node distances:
  node   0   1
  0:  10  21
  1:  21  10
```
NUMA

Querying NUMA configuration

Use numactl to check NUMA nodes configuration

```bash
user@dgx-1v:~$ numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57
58 59
node 0 size: 257844 MB
node 0 free: 255674 MB
node 1 cpus: 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 60 61 62 63 64 65 66 67 68 69 70 71 72 73
74 75 76 77 78 79
node 1 size: 258039 MB
node 1 free: 256220 MB
node distances:
node   0   1
 0:  10  21
 1:  21  10
```
NUMA

Querying NUMA configuration

Use numactl to check NUMA nodes configuration

user@dgx-1v:~$ numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59
node 0 size: 257844 MB
node 0 free: 255674 MB
node 1 cpus: 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79
node 1 size: 258039 MB
node 1 free: 256220 MB
node distances:
node   0   1
0:  10  21
1:  21  10
**NUMA**

**Querying System Topology**

Use `nvidia-smi` to check which CPU is the closest to the given GPU.

```
user@dgx-1v:~$ nvidia-smi topo -mp
```

<table>
<thead>
<tr>
<th></th>
<th>GPU0</th>
<th>GPU1</th>
<th>GPU2</th>
<th>GPU3</th>
<th>GPU4</th>
<th>GPU5</th>
<th>GPU6</th>
<th>GPU7</th>
<th>mlx5_1</th>
<th>mlx5_2</th>
<th>mlx5_3</th>
<th>mlx5_0</th>
<th>CPU Affinity</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU0</td>
<td>X</td>
<td>PIX</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>SYS</td>
<td>SYS</td>
<td>PIX</td>
<td>0-19,40-59</td>
</tr>
<tr>
<td>GPU1</td>
<td>PIX</td>
<td>X</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>SYS</td>
<td>SYS</td>
<td>PIX</td>
<td>0-19,40-59</td>
</tr>
<tr>
<td>GPU2</td>
<td>PHB</td>
<td>PHB</td>
<td>X</td>
<td>PIX</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PIX</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>0-19,40-59</td>
</tr>
<tr>
<td>GPU3</td>
<td>PHB</td>
<td>PHB</td>
<td>PIX</td>
<td>X</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PIX</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>0-19,40-59</td>
</tr>
<tr>
<td>GPU4</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>X</td>
<td>PIX</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
<td>PX</td>
<td>PHB</td>
<td>SYS</td>
<td>20-39,60-79</td>
</tr>
<tr>
<td>GPU5</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PIX</td>
<td>X</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
<td>PX</td>
<td>PHB</td>
<td>SYS</td>
<td>20-39,60-79</td>
</tr>
<tr>
<td>GPU6</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>PHB</td>
<td>X</td>
<td>PIX</td>
<td>SYS</td>
<td>PHB</td>
<td>PX</td>
<td>SYS</td>
<td>20-39,60-79</td>
</tr>
<tr>
<td>GPU7</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>PHB</td>
<td>X</td>
<td>PIX</td>
<td>SYS</td>
<td>PHB</td>
<td>PX</td>
<td>SYS</td>
<td>20-39,60-79</td>
</tr>
<tr>
<td>mlx5_1</td>
<td>PHB</td>
<td>PHB</td>
<td>PIX</td>
<td>PIX</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>X</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>0-19,40-59</td>
</tr>
<tr>
<td>mlx5_2</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PIX</td>
<td>PIX</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
<td>X</td>
<td>PHB</td>
<td>SYS</td>
<td>0-19,40-59</td>
</tr>
<tr>
<td>mlx5_3</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>PHB</td>
<td>PIX</td>
<td>PIX</td>
<td>SYS</td>
<td>PX</td>
<td>PHB</td>
<td>X</td>
<td>20-39,60-79</td>
</tr>
<tr>
<td>mlx5_0</td>
<td>PIX</td>
<td>PIX</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>SYS</td>
<td>SYS</td>
<td>X</td>
<td>20-39,60-79</td>
</tr>
</tbody>
</table>
**NUMA**

**Querying System Topology**

Use `nvidia-smi` to check which CPU is the closest to the given GPU

```bash
user@dgx-1v-~$ nvidia-smi topo -mp
```

<table>
<thead>
<tr>
<th></th>
<th>GPU0</th>
<th>GPU1</th>
<th>GPU2</th>
<th>GPU3</th>
<th>GPU4</th>
<th>GPU5</th>
<th>GPU6</th>
<th>GPU7</th>
<th>mlx5_1</th>
<th>mlx5_2</th>
<th>mlx5_3</th>
<th>mlx5_0</th>
<th>CPU Affinity</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU0</td>
<td>X</td>
<td>PIX</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>SYS</td>
<td>SYS</td>
<td>PIX</td>
</tr>
<tr>
<td>GPU1</td>
<td>PIX</td>
<td>X</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>SYS</td>
<td>SYS</td>
<td>PIX</td>
</tr>
<tr>
<td>GPU2</td>
<td>PHB</td>
<td>PHB</td>
<td>X</td>
<td>PIX</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>0-19,40-59</td>
</tr>
<tr>
<td>GPU3</td>
<td>PHB</td>
<td>PHB</td>
<td>PIX</td>
<td>X</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
</tr>
<tr>
<td>GPU4</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>X</td>
<td>PIX</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
<td>PIX</td>
<td>PHB</td>
<td>SYS</td>
</tr>
<tr>
<td>GPU5</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PIX</td>
<td>X</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
<td>PIX</td>
<td>PHB</td>
<td>SYS</td>
</tr>
<tr>
<td>GPU6</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>PHB</td>
<td>X</td>
<td>PIX</td>
<td>SYS</td>
<td>PHB</td>
<td>PIX</td>
<td>SYS</td>
</tr>
<tr>
<td>GPU7</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>PHB</td>
<td>X</td>
<td>PIX</td>
<td>SYS</td>
<td>PHB</td>
<td>PIX</td>
<td>SYS</td>
</tr>
<tr>
<td>mlx5_1</td>
<td>PHB</td>
<td>PHB</td>
<td>PIX</td>
<td>PIX</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>X</td>
<td>SYS</td>
<td>X</td>
<td>PHB</td>
<td>SYS</td>
</tr>
<tr>
<td>mlx5_2</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PIX</td>
<td>PIX</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
<td>X</td>
<td>PHB</td>
<td>SYS</td>
<td></td>
</tr>
<tr>
<td>mlx5_3</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>PHB</td>
<td>PIX</td>
<td>PIX</td>
<td>SYS</td>
<td>PHB</td>
<td>X</td>
<td>SYS</td>
<td></td>
</tr>
<tr>
<td>mlx5_0</td>
<td>PIX</td>
<td>PIX</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>SYS</td>
<td>SYS</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
**NUMA**

**Querying System Topology**

Use `nvidia-smi` to check which peer-GPUs belong to a different NUMA node

```bash
user@dgx-1v:~$ nvidia-smi topo -mp
```

<table>
<thead>
<tr>
<th></th>
<th>GPU0</th>
<th>GPU1</th>
<th>GPU2</th>
<th>GPU3</th>
<th>GPU4</th>
<th>GPU5</th>
<th>GPU6</th>
<th>GPU7</th>
<th>mlx5_1</th>
<th>mlx5_2</th>
<th>mlx5_3</th>
<th>mlx5_0</th>
<th>CPU Affinity</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU0</td>
<td>X</td>
<td>PIX</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PIX</td>
</tr>
<tr>
<td>GPU1</td>
<td>PIX</td>
<td>X</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PIX</td>
</tr>
<tr>
<td>GPU2</td>
<td>PHB</td>
<td>PHB</td>
<td>X</td>
<td>PIX</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PIX</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
</tr>
<tr>
<td>GPU3</td>
<td>PHB</td>
<td>PHB</td>
<td>PIX</td>
<td>X</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PIX</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
</tr>
<tr>
<td>GPU4</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PIX</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
<td>SYS</td>
<td>PIX</td>
<td>PHB</td>
<td>SYS</td>
</tr>
<tr>
<td>GPU5</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PIX</td>
<td>X</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
<td>SYS</td>
<td>PIX</td>
<td>PHB</td>
<td>SYS</td>
</tr>
<tr>
<td>GPU6</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>PHB</td>
<td>X</td>
<td>PIX</td>
<td>SYS</td>
<td>PHB</td>
<td>PIX</td>
<td>SYS</td>
</tr>
<tr>
<td>GPU7</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>PHB</td>
<td>X</td>
<td>SYS</td>
<td>PHB</td>
<td>PHB</td>
<td>X</td>
<td>SYS</td>
</tr>
<tr>
<td>mlx5_1</td>
<td>PHB</td>
<td>PHB</td>
<td>PIX</td>
<td>PIX</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>X</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
</tr>
<tr>
<td>mlx5_2</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PIX</td>
<td>PIX</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
<td>SYS</td>
<td>X</td>
<td>PHB</td>
<td>SYS</td>
</tr>
<tr>
<td>mlx5_3</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>PHB</td>
<td>PIX</td>
<td>PIX</td>
<td>SYS</td>
<td>PHB</td>
<td>X</td>
<td>SYS</td>
<td></td>
</tr>
<tr>
<td>mlx5_0</td>
<td>PIX</td>
<td>PIX</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>SYS</td>
<td>SYS</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
**NUMA**

**Querying System Topology**

Use `nvidia-smi` to check which peer-GPUs belong to a different NUMA node

<table>
<thead>
<tr>
<th>User@dgx-1v:~$ nvidia-smi topo -mp</th>
<th>GPU0</th>
<th>GPU1</th>
<th>GPU2</th>
<th>GPU3</th>
<th>GPU4</th>
<th>GPU5</th>
<th>GPU6</th>
<th>GPU7</th>
<th>mlx5_1</th>
<th>mlx5_2</th>
<th>mlx5_3</th>
<th>mlx5_0</th>
<th>CPU Affinity</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU0</td>
<td>X</td>
<td>PIX</td>
<td>PHB</td>
<td>PHB</td>
<td>SYST</td>
<td>SYST</td>
<td>SYST</td>
<td>SYST</td>
<td>PHBST</td>
<td>SYST</td>
<td>SYST</td>
<td>SYST</td>
<td>PIXST</td>
</tr>
<tr>
<td>GPU1</td>
<td>PIX</td>
<td>X</td>
<td>PHB</td>
<td>PHB</td>
<td>SYST</td>
<td>SYST</td>
<td>SYST</td>
<td>SYST</td>
<td>PHBST</td>
<td>SYST</td>
<td>SYST</td>
<td>SYST</td>
<td>PIXST</td>
</tr>
<tr>
<td>GPU2</td>
<td>PHB</td>
<td>PHB</td>
<td>X</td>
<td>PIX</td>
<td>SYST</td>
<td>SYST</td>
<td>SYST</td>
<td>SYST</td>
<td>PIXST</td>
<td>SYST</td>
<td>SYST</td>
<td>SYST</td>
<td>PHBST</td>
</tr>
<tr>
<td>GPU3</td>
<td>PHB</td>
<td>PHB</td>
<td>PIX</td>
<td>X</td>
<td>SYST</td>
<td>SYST</td>
<td>SYST</td>
<td>SYST</td>
<td>PIXST</td>
<td>SYST</td>
<td>SYST</td>
<td>SYST</td>
<td>PHBST</td>
</tr>
<tr>
<td>GPU4</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PIX</td>
<td>X</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
<td>PIX</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
</tr>
<tr>
<td>GPU5</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PIX</td>
<td>X</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
<td>PIX</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
</tr>
<tr>
<td>GPU6</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>PHB</td>
<td>X</td>
<td>PIX</td>
<td>SYST</td>
<td>PHB</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
</tr>
<tr>
<td>GPU7</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>PHB</td>
<td>X</td>
<td>PIX</td>
<td>SYST</td>
<td>PHB</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
</tr>
<tr>
<td>mlx5_1</td>
<td>PHB</td>
<td>PHB</td>
<td>PIX</td>
<td>PIX</td>
<td>SYST</td>
<td>SYST</td>
<td>SYST</td>
<td>SYST</td>
<td>X</td>
<td>SYST</td>
<td>SYST</td>
<td>SYST</td>
<td>PHBST</td>
</tr>
<tr>
<td>mlx5_2</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PIX</td>
<td>PIX</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
<td>X</td>
<td>PHB</td>
<td>PHB</td>
<td>SYS</td>
</tr>
<tr>
<td>mlx5_3</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>SYS</td>
<td>PHB</td>
<td>PHB</td>
<td>PIX</td>
<td>PIX</td>
<td>SYS</td>
<td>PHB</td>
<td>X</td>
<td>SYS</td>
<td>SYST</td>
</tr>
<tr>
<td>mlx5_0</td>
<td>PIX</td>
<td>PIX</td>
<td>PHB</td>
<td>PHB</td>
<td>SYST</td>
<td>SYST</td>
<td>SYST</td>
<td>SYST</td>
<td>PHBST</td>
<td>SYST</td>
<td>SYST</td>
<td>SYST</td>
<td>X</td>
</tr>
</tbody>
</table>
NUMA
Achieving Stable Benchmarks

Use closest NUMA node for best stability (…and highest performance)

With numactl, you can set both:

• which NUMA node the application is executed on
• which NUMA node the application allocates memory from

user@dgx-1v:~$ numactl --cpunodebind=0 --membind=0 ./bandwidthTest --device=0
NUMA
Achieving Stable Memory Benchmarks: pageable copies

Better stability (...and performance) with correct NUMA setting

Host-to-device pageable memcopy:
Average Bandwidth - 8.3 GB/s
Relative Standard Deviation - 1%

Device-to-host pageable memcopy:
Average Bandwidth - 11.3 GB/s
Relative Standard Deviation - 0%

DGX-1V, Intel Xeon E5-2698 @ 2.20GHz
NUMA
Achieving Stable CPU Benchmarks: launch latency

Average Launch Latency - 2.47 us
Relative Standard Deviation - 1%

Better stability (...and performance) with “performance” scaling governor and correct NUMA settings
GPU CLOCK SETTINGS

Achieving Stable GPU Benchmarks

compute_gemm() kernel runtimes - RTX 4000

Mean Kernel Runtime - 4.27 ms
Relative Standard Deviation - 3.67%
GPU CLOCK SETTINGS

Achieving Stable GPU Benchmarks

Unrestricted, the GPU’s clock settings can fluctuate significantly

- This can be a result of thermal or power throttling
- Can potentially cause unstable benchmark results

Mean Kernel Runtime - 4.27 ms
Relative Standard Deviation - 3.67%
GPU CLOCK SETTINGS
Monitoring Clocks and Throttling

Using nvidia-smi to monitor clocks while the test is running can reveal what is happening

- ‘nvidia-smi -q -d PERFORMANCE’ will show current Performance State and throttling
- ‘nvidia-smi dmon’ will scroll the current clock of the GPU

<table>
<thead>
<tr>
<th>#</th>
<th>GPU</th>
<th>pwr</th>
<th>gtemp</th>
<th>mtemp</th>
<th>sm</th>
<th>mem</th>
<th>enc</th>
<th>dec</th>
<th>mclk</th>
<th>pclk</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>22</td>
<td>59</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>405</td>
<td>300</td>
</tr>
<tr>
<td>0</td>
<td>57</td>
<td>61</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>650</td>
<td>1215</td>
</tr>
<tr>
<td>0</td>
<td>131</td>
<td>66</td>
<td>-</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>650</td>
<td>1575</td>
</tr>
<tr>
<td>0</td>
<td>130</td>
<td>68</td>
<td>-</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>650</td>
<td>1530</td>
</tr>
<tr>
<td>0</td>
<td>130</td>
<td>69</td>
<td>-</td>
<td>66</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>650</td>
<td>1530</td>
</tr>
<tr>
<td>0</td>
<td>129</td>
<td>70</td>
<td>-</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>650</td>
<td>1515</td>
</tr>
<tr>
<td>0</td>
<td>131</td>
<td>70</td>
<td>-</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>650</td>
<td>1580</td>
</tr>
<tr>
<td>0</td>
<td>129</td>
<td>70</td>
<td>-</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>650</td>
<td>1485</td>
</tr>
<tr>
<td>0</td>
<td>130</td>
<td>71</td>
<td>-</td>
<td>64</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>650</td>
<td>1485</td>
</tr>
<tr>
<td>0</td>
<td>130</td>
<td>72</td>
<td>-</td>
<td>64</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>650</td>
<td>1485</td>
</tr>
<tr>
<td>0</td>
<td>128</td>
<td>72</td>
<td>-</td>
<td>64</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>650</td>
<td>1485</td>
</tr>
<tr>
<td>0</td>
<td>130</td>
<td>73</td>
<td>-</td>
<td>62</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>650</td>
<td>1485</td>
</tr>
<tr>
<td>0</td>
<td>130</td>
<td>74</td>
<td>-</td>
<td>62</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>650</td>
<td>1485</td>
</tr>
<tr>
<td>0</td>
<td>128</td>
<td>74</td>
<td>-</td>
<td>62</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>650</td>
<td>1485</td>
</tr>
<tr>
<td>0</td>
<td>128</td>
<td>75</td>
<td>-</td>
<td>62</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>650</td>
<td>1410</td>
</tr>
<tr>
<td>0</td>
<td>128</td>
<td>75</td>
<td>-</td>
<td>62</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>650</td>
<td>1410</td>
</tr>
<tr>
<td>0</td>
<td>128</td>
<td>76</td>
<td>-</td>
<td>61</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>650</td>
<td>1965</td>
</tr>
<tr>
<td>0</td>
<td>128</td>
<td>76</td>
<td>-</td>
<td>61</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>650</td>
<td>1965</td>
</tr>
<tr>
<td>0</td>
<td>62</td>
<td>73</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>650</td>
<td>420</td>
</tr>
</tbody>
</table>
GPU CLOCK SETTINGS
Monitoring Clocks and Throttling

Using nvidia-smi to monitor clocks while the test is running can reveal what is happening.
GPU CLOCK SETTINGS
Achieving Stable GPU Benchmarks

To achieve **stable** results, best practice is to lock the GPU’s clock to default

- Clocks higher than default can be chosen, but monitor throttling with `nvidia-smi`
- ‘`nvidia-smi -q -d SUPPORTED_CLOCKS’’ lists available clock settings
- ‘`nvidia-smi -q -d CLOCK’’ shows current GPU clocks

```
==============NVSMI LOG==============
Timestamp : Fri Feb 22 11:27:21 2019
Driver Version : 418.39
CUDA Version : 10.1
Attached GPUs : 1
GPU 00000000:01:00.0
Clocks
Graphics : 300 MHz
SM : 300 MHz
Memory : 405 MHz
Video : 540 MHz
Applications Clocks
Graphics : 1215 MHz
Memory : 6501 MHz
Default Applications Clocks
Graphics : 1215 MHz
Memory : 6501 MHz
Max Clocks
Graphics : 2100 MHz
SM : 2100 MHz
Memory : 6501 MHz
Video : 1950 MHz
```
GPU CLOCK SETTINGS

Achieving Stable GPU Benchmarks

Use the values from “Default Application Clocks” for more stable benchmarking

- `nvidia-smi -ac <Default Memory Clock>,<Default Graphics Clock>` to lock the clocks while an application is running on the GPU

For Volta+

- `nvidia-smi -lgc <Default Graphics Clock>` to lock the GPU clocks regardless of if an application is running

Note that persistence mode must be enabled for the setting to stick
Note that absolute performance may be lower at default clocks, but we’re after stable rather than peak performance.
GPU CLOCK SETTINGS
Monitoring Clocks and Throttling

Clocks Locked to Default

- Clock (MHz)
- Power (W) and Temp (°C)
- Clock - Locked
- Clock - Unlocked
- Pwr - Locked
- Pwr - Unlocked
- Temp - Locked
- Temp - Unlocked
- Power Cap
GPU CLOCK SETTINGS

Monitoring Clocks and Throttling

‘nvidia-smi dmon’ output for both runs

<table>
<thead>
<tr>
<th>#</th>
<th>gpu</th>
<th>pwr</th>
<th>gtemp</th>
<th>mtemp</th>
<th>sm</th>
<th>mem</th>
<th>enc</th>
<th>dec</th>
<th>mclk</th>
<th>pclk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idx</td>
<td>W</td>
<td>C</td>
<td>C</td>
<td>%</td>
<td>%</td>
<td>%</td>
<td>MHz</td>
<td>MHz</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>22</td>
<td>59</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>405</td>
<td>300</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>57</td>
<td>61</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>131</td>
<td>66</td>
<td>-</td>
<td>22</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1575</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>130</td>
<td>68</td>
<td>-</td>
<td>100</td>
<td>66</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1510</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>130</td>
<td>69</td>
<td>-</td>
<td>100</td>
<td>66</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1530</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>129</td>
<td>70</td>
<td>-</td>
<td>100</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1515</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>131</td>
<td>70</td>
<td>-</td>
<td>100</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1500</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>128</td>
<td>70</td>
<td>-</td>
<td>100</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1495</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>130</td>
<td>71</td>
<td>-</td>
<td>100</td>
<td>64</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1455</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>130</td>
<td>72</td>
<td>-</td>
<td>100</td>
<td>64</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1485</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>128</td>
<td>72</td>
<td>-</td>
<td>100</td>
<td>64</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1455</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>130</td>
<td>72</td>
<td>-</td>
<td>100</td>
<td>63</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1440</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>130</td>
<td>73</td>
<td>-</td>
<td>100</td>
<td>62</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1455</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>130</td>
<td>74</td>
<td>-</td>
<td>100</td>
<td>62</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1440</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>128</td>
<td>74</td>
<td>-</td>
<td>100</td>
<td>62</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1455</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>129</td>
<td>74</td>
<td>-</td>
<td>100</td>
<td>62</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1440</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>130</td>
<td>75</td>
<td>-</td>
<td>100</td>
<td>62</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1410</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>128</td>
<td>75</td>
<td>-</td>
<td>100</td>
<td>62</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1410</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>128</td>
<td>76</td>
<td>-</td>
<td>100</td>
<td>61</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1965</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>128</td>
<td>76</td>
<td>-</td>
<td>60</td>
<td>38</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1965</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>62</td>
<td>73</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>420</td>
<td></td>
</tr>
</tbody>
</table>

Unlocked:

<table>
<thead>
<tr>
<th>#</th>
<th>gpu</th>
<th>pwr</th>
<th>gtemp</th>
<th>mtemp</th>
<th>sm</th>
<th>mem</th>
<th>enc</th>
<th>dec</th>
<th>mclk</th>
<th>pclk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idx</td>
<td>W</td>
<td>C</td>
<td>C</td>
<td>%</td>
<td>%</td>
<td>%</td>
<td>MHz</td>
<td>MHz</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>27</td>
<td>59</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>810</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>57</td>
<td>60</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>102</td>
<td>66</td>
<td>-</td>
<td>41</td>
<td>22</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>102</td>
<td>64</td>
<td>-</td>
<td>100</td>
<td>54</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>102</td>
<td>64</td>
<td>-</td>
<td>100</td>
<td>54</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>104</td>
<td>67</td>
<td>-</td>
<td>100</td>
<td>54</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>105</td>
<td>67</td>
<td>-</td>
<td>100</td>
<td>54</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>106</td>
<td>68</td>
<td>-</td>
<td>100</td>
<td>54</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>107</td>
<td>69</td>
<td>-</td>
<td>100</td>
<td>54</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>108</td>
<td>69</td>
<td>-</td>
<td>100</td>
<td>54</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>109</td>
<td>70</td>
<td>-</td>
<td>100</td>
<td>54</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>110</td>
<td>70</td>
<td>-</td>
<td>100</td>
<td>54</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>111</td>
<td>71</td>
<td>-</td>
<td>100</td>
<td>54</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>111</td>
<td>72</td>
<td>-</td>
<td>100</td>
<td>54</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>111</td>
<td>72</td>
<td>-</td>
<td>100</td>
<td>54</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>111</td>
<td>72</td>
<td>-</td>
<td>100</td>
<td>54</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>111</td>
<td>73</td>
<td>-</td>
<td>100</td>
<td>54</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>111</td>
<td>74</td>
<td>-</td>
<td>100</td>
<td>54</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>111</td>
<td>74</td>
<td>-</td>
<td>100</td>
<td>54</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>111</td>
<td>75</td>
<td>-</td>
<td>100</td>
<td>54</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>111</td>
<td>75</td>
<td>-</td>
<td>100</td>
<td>54</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>111</td>
<td>75</td>
<td>-</td>
<td>100</td>
<td>54</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>111</td>
<td>76</td>
<td>-</td>
<td>78</td>
<td>41</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>63</td>
<td>78</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6500</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>38</td>
<td>68</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>810</td>
<td>1215</td>
<td></td>
</tr>
</tbody>
</table>

Locked to Default:
MEASURING THE RIGHT THING
CUDA JIT COMPILATION

Performance Considerations

When a CUDA fat binary doesn’t include code for the architecture to be executed, the PTX (if available) is just-in-time compiled by the driver.

In order to reduce CUDA module load time, JIT results are cached on the filesystem.

Default locations for JIT cache:

- Linux - ~/.nv/ComputeCache
- Windows - %APPDATA\%NVIDIA\ComputeCache
CUDA JIT Compilation

Performance Considerations

For certain environments these default locations can be problematic:

- If the location is a network filesystem, access can be slow.
- If the location is shared across nodes, concurrent access can result in drops in performance.

JIT cache location and usage is configurable:

- Environment variable CUDA_CACHE_PATH can be used to set the location.
- Environment variable CUDA_CACHE_DISABLE can be used to skip the cache entirely.
CUDA JIT COMPILATION

Performance Considerations

JITing can be time consuming, especially on a cache miss.

Can be invoked during module load and runtime initialization.

Avoid timing JITing when benchmarking code unless specifically required.

- Use appropriate architecture flags to create fat binaries to avoid JITing and the JIT cache.
- For Example: `-gencode=arch=compute_75,code=sm_75`
- See `nvcc` documentation for details.
CUDA EVENTS
Timing Event Issues

Using events to time kernels in complex multi-stream cases can result in unexpected results

- Example: Start and end events recorded for each kernel launch across 4 streams

```c
for (int i = 0, j = 0; i < NUM_RUNS / NUM_STREAMS; i++) {
    for (int iStream = 0; iStream < NUM_STREAMS; iStream++, j++) {
        cudaEventRecord(startEvents[j], streams[iStream]);
        compute_gemm<<<gridDim, blockDim, SHMEM_SZ, streams[iStream]>>>(...);
        cudaEventRecord(stopEvents[j], streams[iStream]);
    }
}
```
CUDA EVENTS
Timing Event Issues

The expectation might be that each event pair reports ~5ms (the kernel runtime)

- Events have no affinity to the preceding or subsequent GPU work
- Only ordering within the stream is guaranteed

Expected recorded event times:
CUDA EVENTS
Timing Event Issues

Start and end events have additional work from other streams interleaved

- Per kernel events report 1-4x actual kernel execution time (with 4 streams)
- Default stream events timing the entire run are accurate

Actual recorded event times:
CUDA EVENTS

Timing Event Issues

Even for single stream, other GPU operations can be executed between the start and end event.

Events will record the time the GPU executes the event on the given stream.

- Useful for measuring stream work with respect to the CPU.
- Useful for coarser measurements, but not short running kernels.

Nsight Compute and Nsight Systems are better suited for measuring specific GPU kernels when using multiple streams.

- Both have access driver internals that allow for accurate measurement of GPU operations.
CUDA EVENTS

Other Event Benchmarking Troubles

Events have timing enabled by default

- Recording a time may result in synchronization, potentially reducing concurrency

To use events for explicit synchronization or querying, disable timing when creating the event

- Use cudaEventDisableTiming or CU_EVENT_DISABLE_TIMING flags to disable timing on creation
API OVERHEAD

Latency spikes

4 threads, 1 stream per thread, loop event record + GEMM + event record in each stream

- OS runtime libraries
- CUDA API
- [2426] cudaTensorCoreG
- [2429]
- OS runtime libraries
- CUDA API
- [2431]
- OS runtime libraries
- CUDA API
- [2430]
API OVERHEAD
Latency spikes

4 threads, 1 stream per thread, loop event record + GEMM + event record in each stream

- OS runtime libraries
  - pthread_mutex_lock
  - compute_gemm
  - cudaEventRecord

- CUDA API
  - compute_gemm
  - cudaEventRecord

- [2426] cudaTensorCoreG
  - pthread_mutex_lock
  - compute_gemm
  - cudaEventRecord

- [2431]
  - pthread_mutex_lock
  - compute_gemm
  - cudaEventRecord
  - cudaEventRecord

- [2430]
  - pthread_mutex_lock
  - compute_gemm
  - cudaEventRecord

< 10 us / call
4 threads, 1 stream per thread, loop event record + GEMM + event record in each stream

- **OS runtime libraries**
  - pthread_mutex_lock
  - compute_gemm
  - pthread_mutex_lock
  - cudaEventRecord

  - [622 us]
  - < 10 us / call

- **CUDA API**
  - compute_gemm
  - cudaEventRecord

  - [537 us]
  - < 10 us / call

- **CUDA runtime libraries**
  - pthread_mutex_lock
  - compute_gemm
  - pthread_mutex_lock
  - cudaEventRecord

  - [795 us]
  - < 10 us / call

- **CUDA runtime libraries**
  - pthread_mutex_lock
  - compute_gemm
  - pthread_mutex_lock
  - cudaEventRecord

  - [665 us]
  - < 10 us / call

**API OVERHEAD**

Latency spikes
API OVERHEAD

Lock contention

pthread_mutex is not fair and depends on OS scheduler to select the next thread
# API OVERHEAD

## How to avoid it?

<table>
<thead>
<tr>
<th>Approach</th>
<th>Benefit</th>
<th>More information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submit work from a single CPU worker thread</td>
<td>Eliminates inter-thread lock contention</td>
<td>This presentation</td>
</tr>
<tr>
<td>Batch work submission when using many CPU threads</td>
<td>Eliminates some of inter-thread lock contention</td>
<td>GTC 2019 - CE9147 Connect with the Experts: CUDA Platform</td>
</tr>
<tr>
<td>Try <a href="#">CUDA Graphs</a> to minimize overall API overheads</td>
<td>Reduces overheads by &gt;2x</td>
<td>GTC 2019 - S9240 CUDA: New Features and Beyond</td>
</tr>
<tr>
<td>Combine kernels together to avoid API calls</td>
<td>Single kernel eliminates launch and inter-kernel overheads</td>
<td>Cooperative Groups: Flexible CUDA Thread Programming Devblog</td>
</tr>
<tr>
<td>Go multi-process with Volta+ MPS</td>
<td>Separates launching threads and avoids locks</td>
<td>GTC 2017 - S7798 Inside Volta</td>
</tr>
</tbody>
</table>
SUMMARY
Best Practices when benchmarking CUDA applications

System stability

  CPU Frequency Scaling - Use performance governor and disable Turbo Boost
  NUMA - Use ‘numactl’ to control NUMA behavior
  GPU clocks - Lock GPU clocks for stable benchmarking

Measuring the right thing

  JIT cache - Check the location or avoid JITing entirely
  CUDA events - Use Nsight tools for better measurements
  API contention - Take steps to avoid lock contention