# Analysis Report

**pollardKernel(unsigned int const *, unsigned int*)**

| | |
|---|---|
| Duration | 3,85 s (3 850 292 108 ns) |
| Grid Size | [ 1024,1,1 ] |
| Block Size | [ 100,1,1 ] |
| Registers/Thread | 43 |
| Shared  Memory/Block | 0 B |
| Shared Memory Requested | 16 KiB |
| Shared Memory Executed | 16 KiB |
| Shared Memory Bank Size | 4 B |

[0] GeForce GTX 780 Ti

| | |
|---|---|
| GPU UUID | GPU-6dfac189-49e4-701d-5da2-17d8a2ae1f8b |
| Compute Capability | 3.5 |
| Max. Threads per Block | 1024 |
| Max. Threads per Multiprocessor | 2048 |
| Max. Shared Memory per Block | 48 KiB |
| Max. Shared Memory per Multiprocessor | 48 KiB |
| Max. Registers per Block | 65536 |
| Max. Registers per Multiprocessor | 65536 |
| Max. Grid Dimensions | [ 2147483647, 65535, 65535 ] |
| Max. Block Dimensions | [ 1024, 1024, 64 ] |
| Max. Warps per Multiprocessor | 64 |
| Max. Blocks per Multiprocessor | 16 |
| Single Precision FLOP/s | 6,172 TeraFLOP/s |
| Double Precision FLOP/s | 257,16 GigaFLOP/s |
| Number of Multiprocessors | 15 |
| Multiprocessor Clock Rate | 1,071 GHz |
| Concurrent Kernel | true |
| Max IPC | 7 |
| Threads per Warp | 32 |
| Global Memory Bandwidth | 336 GB/s |
| Global Memory Size | 2,95 GiB |
| Constant Memory Size | 64 KiB |
| L2 Cache Size | 1,5 MiB |
| Memcpy Engines | 1 |
| PCIe Generation | 3 |
| PCIe Link Rate | 8 Gbit/s |
| PCIe Link Width | 16 |

**1. Compute, Bandwidth, or Latency Bound**

The first step in analyzing an individual kernel is to determine if the performance of the kernel is bounded by computation, memory bandwidth, or instruction/memory latency. Unfortunately, the device executing this kernel can not provide the profile data needed for this analysis.

**2. Instruction and Memory Latency**

Instruction and memory latency limit the performance of a kernel when the GPU does not have enough work to keep busy. Unfortunately, the device executing this kernel can not provide the profile data needed for this analysis.

## 3. Compute Resources

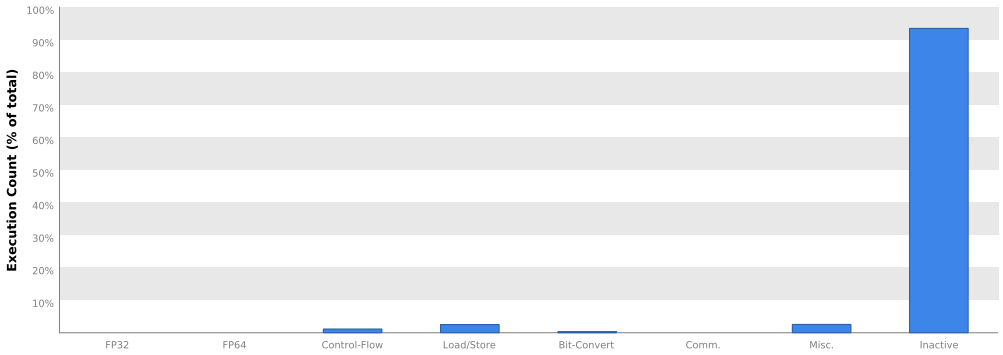
GPU compute resources limit the performance of a kernel when those resources are insufficient or poorly utilized. Compute resources are used most efficiently when all threads in a warp have the same branching and predication behavior. The results below indicate that a significant fraction of the available compute performance is being wasted because branch and predication behavior is differing for threads within a warp.

### 3.1. Divergent Branches

Compute resource are used most efficiently when all threads in a warp have the same branching behavior. When this does not occur the branch is said to be divergent. Divergent branches lower warp execution efficiency which leads to inefficient use of the GPU's compute resources.
*Optimization: Each entry below points to a divergent branch within the kernel. For each branch reduce the amount of intra-warp divergence.*

### 3.2. Instruction Execution Counts

The following chart shows the mix of instructions executed by the kernel. The instructions are grouped into classes and for each class the chart shows the percentage of thread execution cycles that were devoted to executing instructions in that class. The "Inactive" result shows the thread executions that did not execute any instruction because the thread was predicated or inactive due to divergence.




### 3.3. Floating-Point Operation Counts

The following chart shows the mix of floating-point operations executed by the kernel. The operations are grouped into classes and for each class the chart shows the percentage of thread execution cycles that were devoted to executing operations in that class. The results do not sum to 100% because non-floating-point operations executed by the kernel are not shown in this chart.

**4. Memory Bandwidth**

Memory bandwidth limits the performance of a kernel when one or more memories in the GPU cannot provide data at the rate requested by the kernel. Unfortunately, the device executing this kernel can not provide the profile data needed for this analysis.