Page: Details | Launch: 0 - 492 - kernel_account_balance | Add Baseline | Apply Rules | Occupancy Calculator | Save as PDF

	Launch	Time	Cycles	Regs	GPU	SM Frequency	CC	Process
Current	492 - kernel_account_balance (79, 1, 1)x(256, 1, 1)	22.84 msecond	32,227,825	30	0 - NVIDIA A100-SXM4-40GB MIG 1g.5gb	1.41 cycle/nsecond	8.0	[584413] account_savings

Warning: Data collection happened without GPU frequencies fixed by the profiler. Some results may be inconsistent.

GPU Speed Of Light Throughput

High-level overview of the throughput for compute and memory resources of the GPU. For each unit, the throughput reports the achieved percentage of utilization with respect to the theoretical maximum. Breakdowns show the throughput for each individual sub-metric of Compute and Memory to clearly identify the highest contributor. High-level overview of the utilization for compute and memory resources of the GPU presented as a roofline chart.

Compute (SM) Throughput [%]	6.23	Duration [msecond]	22.84
Memory Throughput [%]	71.97	Elapsed Cycles [cycle]	32,227,825
L1/TEX Cache Throughput [%]	14.01	SM Active Cycles [cycle]	31,834,682.07
L2 Cache Throughput [%]	20.19	SM Frequency [cycle/nsecond]	1.41
DRAM Throughput [%]	71.97	DRAM Frequency [cycle/nsecond]	1.22

High Memory Throughput Memory is more heavily utilized than Compute: Look at the Memory Workload Analysis report section to see where the memory system bottleneck is. Check memory replay (coalescing) metrics to make sure you're efficiently utilizing the bytes transferred. Also consider whether it is possible to do more work per memory access (kernel fusion) or whether there are values you can (re)compute.

Roofline Analysis The ratio of peak float (fp32) to double (fp64) performance on this device is 2:1. The kernel achieved 0% of this device's fp32 peak performance and 0% of its fp64 peak performance. See the Kernel Profiling Guide for mode details on roofline analysis.

Compute Workload Analysis

Detailed analysis of the compute resources of the streaming multiprocessors (SM), including the achieved instructions per clock (IPC) and the utilization of each available pipeline. Pipelines with very high utilization might limit the overall performance.

Executed Ipc Elapsed [inst/cycle]	0.20	SM Busy [%]	5.13
Executed Ipc Active [inst/cycle]	0.21	Issue Slots Busy [%]	5.13
Issued Ipc Active [inst/cycle]	0.21		

Low Utilization All pipelines are under-utilized. Either this kernel is very small or it doesn't issue enough warps per scheduler. Check the Launch Statistics and Scheduler Statistics sections for further details.

Memory Workload Analysis

Detailed analysis of the memory resources of the GPU. Memory can become a limiting factor for the overall kernel performance when fully utilizing the involved hardware units (Mem Busy), exhausting the available communication bandwidth between those units (Max Bandwidth), or by reaching the maximum throughput of issuing memory instructions (Mem Pipes Busy). Detailed chart of the memory units. Detailed tables with data for each memory unit.

Memory Throughput [Gbyte/second]	140.03	Mem Busy [%]	16.20
L1/TEX Hit Rate [%]	11.11	Max Bandwidth [%]	71.97
L2 Hit Rate [%]	50.00	Mem Pipes Busy [%]	6.23
L2 Compression Success Rate [%]	0	L2 Compression Ratio	0

Scheduler Statistics

Summary of the activity of the schedulers issuing instructions. Each scheduler maintains a pool of warps that it can issue instructions for. The upper bound of warps in the pool (Theoretical Warps) is limited by the launch configuration. On every cycle each scheduler checks the state of the allocated warps in the pool (Active Warps). Active warps that are not stalled (Eligible Warps) are ready to issue their next instruction. From the set of eligible warps the scheduler selects a single warp from which to issue one or more instructions (Issued Warp). On cycles with no eligible warps, the issue slot is skipped and no instruction is issued. Having many skipped issue slots indicates poor latency hiding.

Active Warps Per Scheduler [warp]	10.83	No Eligible [%]	94.87
Eligible Warps Per Scheduler [warp]	0.09	One or More Eligible [%]	5.13
Issued Warp Per Scheduler	0.05		

Issue Slot Utilization Every scheduler is capable of issuing one instruction per cycle, but for this kernel each scheduler only issues an instruction every 19.5 cycles. This might leave hardware resources underutilized and may lead to less optimal performance. Out of the maximum of 16 warps per scheduler, this kernel allocates an average of 10.83 active warps per scheduler, but only an average of 0.09 warps were eligible per cycle. Eligible warps are the subset of active warps that are ready to issue their next instruction. Every cycle with no eligible warp results in no instruction being issued and the issue slot remains unused. To increase the number of eligible warps, reduce the time the active warps are stalled by inspecting the top stall reasons on the Warp State Statistics and Source Counters sections.

Warp State Statistics

Analysis of the states in which all warps spent cycles during the kernel execution. The warp states describe a warp's readiness or inability to issue its next instruction. The warp cycles per instruction define the latency between two consecutive instructions. The higher the value, the more warp parallelism is required to hide this latency. For each warp state, the chart shows the average number of cycles spent in that state per issued instruction. Stalls are not always impacting the overall performance nor are they completely avoidable. Only focus on stall reasons if the schedulers fail to issue every cycle. When executing a kernel with mixed library and user code, these metrics show the combined values.

Warp Cycles Per Issued Instruction [cycle]	211.04	Avg. Active Threads Per Warp	32
Warp Cycles Per Executed Instruction [cycle]	211.04	Avg. Not Predicated Off Threads Per Warp	32.00

long_scoreboard On average, each warp of this kernel spends 205.7 cycles being stalled waiting for a scoreboard dependency on a L1TEX (local, global, surface, texture) operation. This represents about 97.5% of the total average of 211.0 cycles between issuing two instructions. To reduce the number of cycles waiting on L1TEX data accesses verify the memory access patterns are optimal for the target architecture, attempt to increase cache hit rates by increasing data locality or by changing the cache configuration, and consider moving frequently used data to shared memory.

Warp Stall Check the Source Counters section for the top stall locations in your source based on sampling data. The Kernel Profiling Guide provides more details on each stall reason.

Instruction Statistics

Statistics of the executed low-level assembly instructions (SASS). The instruction mix provides insight into the types and frequency of the executed instructions. A narrow mix of instruction types implies a dependency on few instruction pipelines, while others remain unused. Using multiple pipelines allows hiding latencies and enables parallel execution. Note that 'Instructions/Opcode' and 'Executed Instructions' are measured differently and can diverge if cycles are spent in system calls.

Executed Instructions [inst]	91,443,167	Avg. Executed Instructions Per Scheduler [inst]	1,632,913.70
Issued Instructions [inst]	91,445,873	Avg. Issued Instructions Per Scheduler [inst]	1,632,962.02

NVLink Topology

NVLink Topology diagram shows logical NVLink connections with transmit/receive throughput.

NVLink Tables

Detailed tables with properties for each NVLink.

Launch Statistics

Summary of the configuration used to launch the kernel. The launch configuration defines the size of the kernel grid, the division of the grid into blocks, and the GPU resources needed to execute the kernel. Choosing an efficient launch configuration maximizes device utilization.

Grid Size	79	Registers Per Thread [register/thread]	30
Block Size	256	Static Shared Memory Per Block [byte/block]	0
Threads [thread]	20,224	Dynamic Shared Memory Per Block [byte/block]	0
Waves Per SM	0.71	Driver Shared Memory Per Block [Kbyte/block]	1.02
Function Cache Configuration	cudaFuncCachePreferNone	Shared Memory Configuration Size [Kbyte]	32.77

Occupancy

Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the hardware's ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencies, resulting in overall performance degradation. Large discrepancies between the theoretical and the achieved occupancy during execution typically indicates highly imbalanced workloads.

Theoretical Occupancy [%]	100	Block Limit Registers [block]	8
Theoretical Active Warps per SM [warp]	64	Block Limit Shared Mem [block]	164
Achieved Occupancy [%]	67.64	Block Limit Warps [block]	8
Achieved Active Warps Per SM [warp]	43.29	Block Limit SM [block]	32

Occupancy Limiters This kernel's theoretical occupancy is not impacted by any block limit. The difference between calculated theoretical (100.0%) and measured achieved occupancy (67.6%) can be the result of warp scheduling overheads or workload imbalances during the kernel execution. Load imbalances can occur between warps within a block as well as across blocks of the same kernel. See the CUDA Best Practices Guide for more details on optimizing occupancy.

Source Counters

Source metrics, including branch efficiency and sampled warp stall reasons. Sampling Data metrics are periodically sampled over the kernel runtime. They indicate when warps were stalled and couldn't be scheduled. See the documentation for a description of all stall reasons. Only focus on stalls if the schedulers fail to issue every cycle.

Branch Instructions [inst]	788,757	Branch Efficiency [%]	100
Branch Instructions Ratio [%]	0.01	Avg. Divergent Branches	0