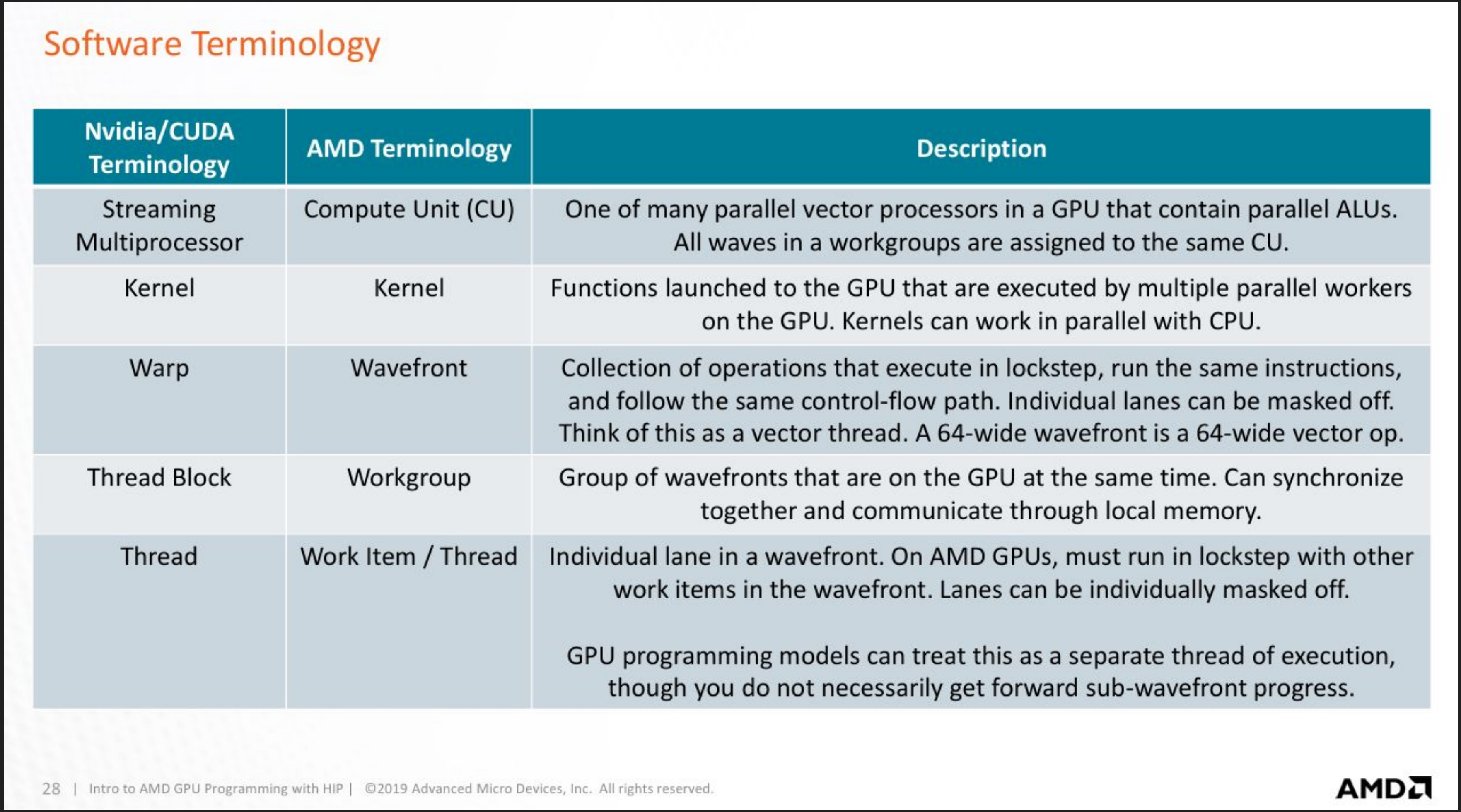
GPU Occupancy Analysis with Gem5

Objectives

- Edit GPU occupancy analysis script from Module 6 to search for the correct stat names in gem5 stats files
- Adjust occupancy formulas to match with occupancy definitions
- Sweep through HIP parameters using HIP-samples code
- Compare GPU occupancy results between runs

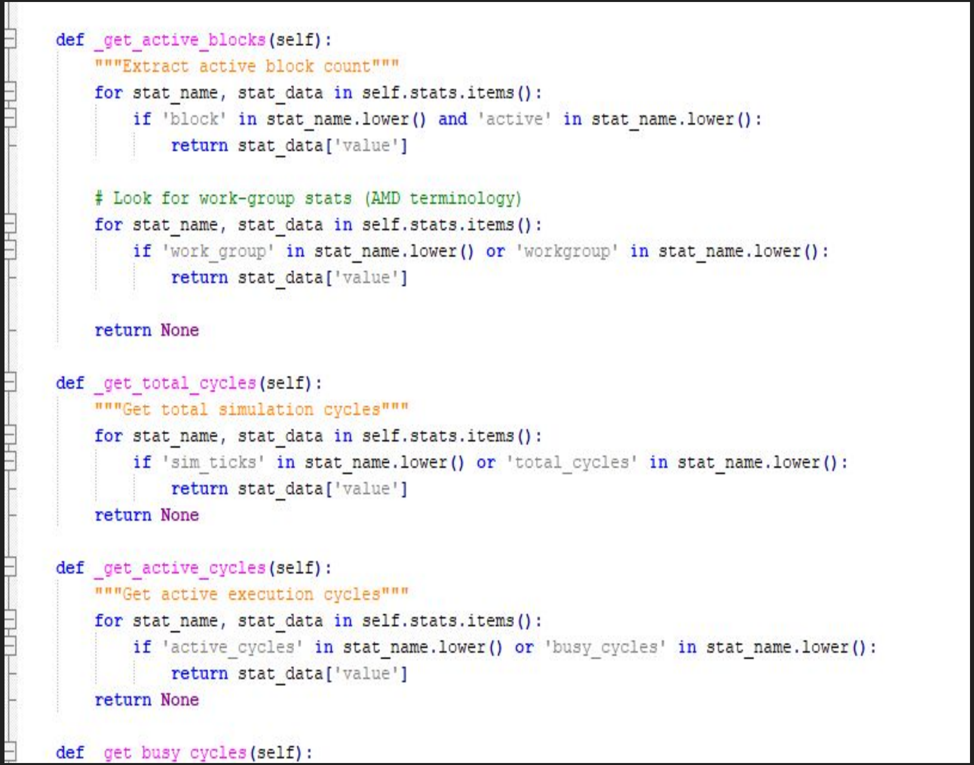


Software Terminology

Nvidia/CUDA Terminology	AMD Terminology	Description
Streaming Multiprocessor	Compute Unit (CU)	One of many parallel vector processors in a GPU that contain parallel ALUs. All waves in a workgroups are assigned to the same CU.
Kernel	Kernel	Functions launched to the GPU that are executed by multiple parallel workers on the GPU. Kernels can work in parallel with CPU.
Warp	Wavefront	Collection of operations that execute in lockstep, run the same instructions, and follow the same control-flow path. Individual lanes can be masked off. Think of this as a vector thread. A 64-wide wavefront is a 64-wide vector op.
Thread Block	Workgroup	Group of wavefronts that are on the GPU at the same time. Can synchronize together and communicate through local memory.
Thread	Work Item / Thread	Individual lane in a wavefront. On AMD GPUs, must run in lockstep with other work items in the wavefront. Lanes can be individually masked off. GPU programming models can treat this as a separate thread of execution, though you do not necessarily get forward sub-wavefront progress.

https://www.olcf.ornl.gov/wp-content/uploads/2019/09/AMD_GPU_HIP_training_20190906.pdf

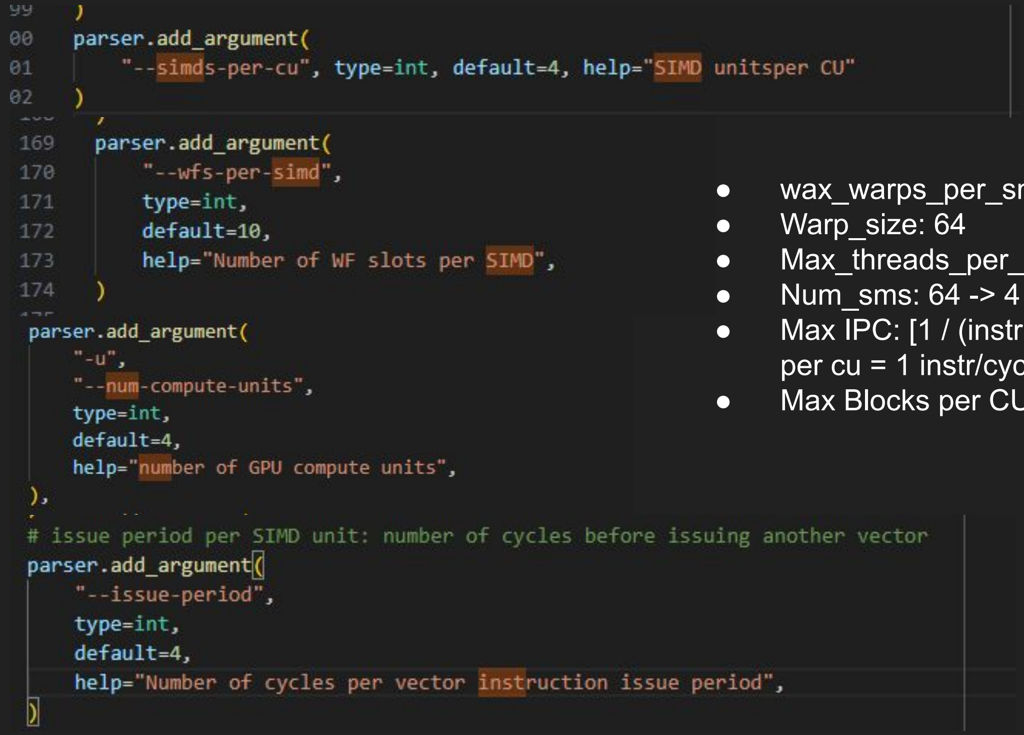
Original GPU_occupancy_analyzer.py Issues



```
    def _get_active_blocks(self):
        """Extract active block count"""
        for stat_name, stat_data in self.stats.items():
            if 'block' in stat_name.lower() and 'active' in stat_name.lower():
                return stat_data['value']

        # Look for work-group stats (AMD terminology)
        for stat_name, stat_data in self.stats.items():
            if 'work_group' in stat_name.lower() or 'workgroup' in stat_name.lower():
                return stat_data['value']

        return None

    def _get_total_cycles(self):
        """Get total simulation cycles"""
        for stat_name, stat_data in self.stats.items():
            if 'sim_ticks' in stat_name.lower() or 'total_cycles' in stat_name.lower():
                return stat_data['value']
        return None

    def _get_active_cycles(self):
        """Get active execution cycles"""
        for stat_name, stat_data in self.stats.items():
            if 'active_cycles' in stat_name.lower() or 'busy_cycles' in stat_name.lower():
                return stat_data['value']
        return None

    def _get_busy_cycles(self):
```

- Matching stat names did not exist, or did exist but did not represent what we wanted to look for

Changes to GPU_config



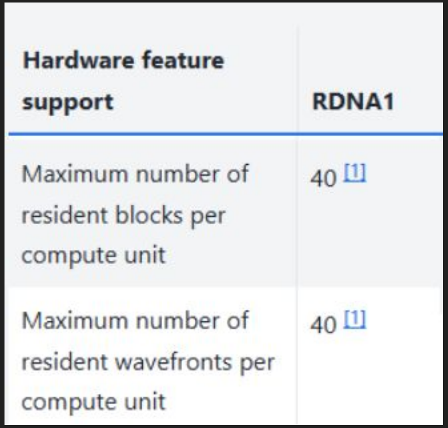
```
)
parser.add_argument(
    "--simds-per-cu", type=int, default=4, help="SIMD unitsper CU"
)
```

```
)
parser.add_argument(
    "--wfs-per-simd",
    type=int,
    default=10,
    help="Number of WF slots per SIMD",
)
```

```
parser.add_argument(
    "-u",
    "--num-compute-units",
    type=int,
    default=4,
    help="number of GPU compute units",
),
```

```
# issue period per SIMD unit: number of cycles before issuing another vector
parser.add_argument(
    "--issue-period",
    type=int,
    default=4,
    help="Number of cycles per vector instruction issue period",
)
```

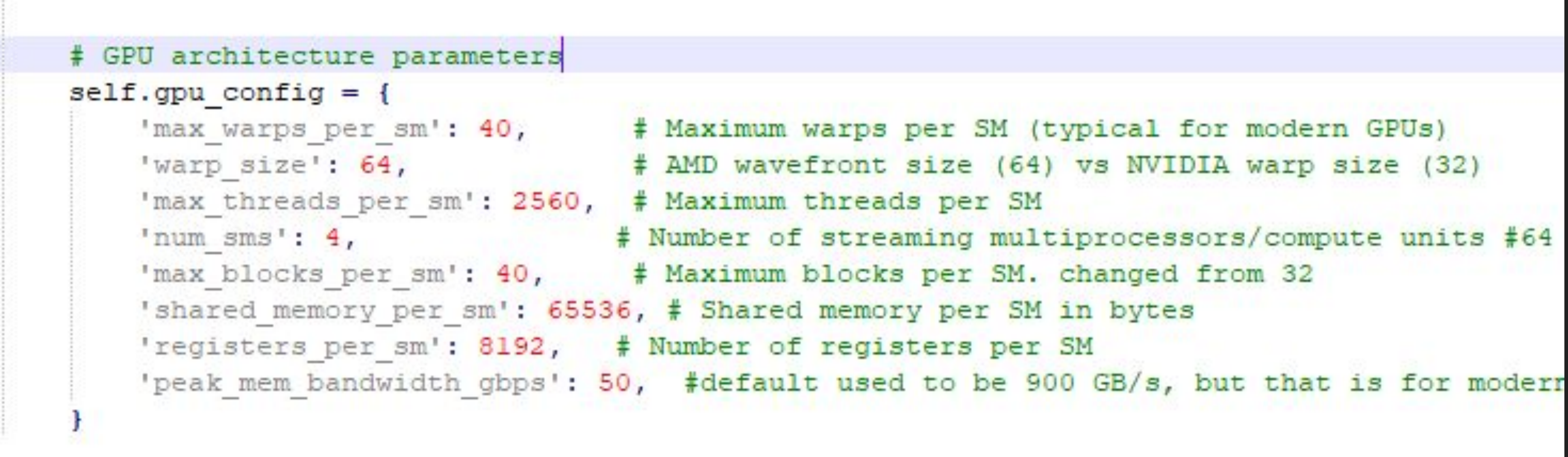
- wax_warps_per_sm : 64 -> 40
- Warp_size: 64
- Max_threads_per_sm: 2048 -> 2560
- Num_sms: 64 -> 4
- Max IPC: [1 / (instruction issue period)] times SIMDs per cu = 1 instr/cycle
- Max Blocks per CU: 40



Hardware feature support	RDNA1
Maximum number of resident blocks per compute unit	40 [1]
Maximum number of resident wavefronts per compute unit	40 [1]

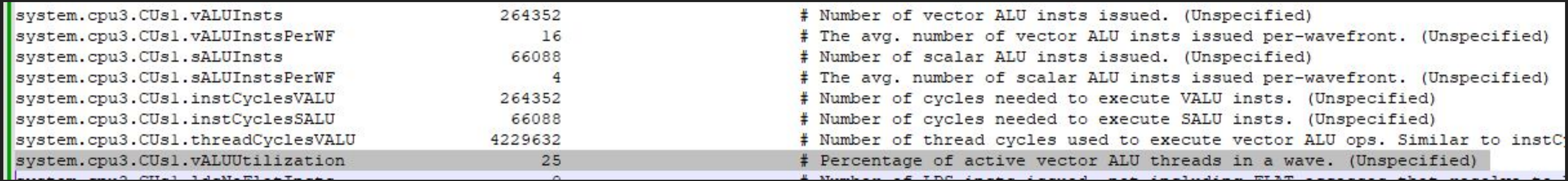
Hardware features — HIP 7.1.52802 Documentation

GPU Config

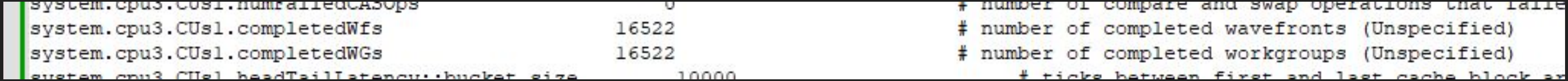


```
# GPU architecture parameters
self.gpu_config = {
    'max_warps_per_sm': 40,      # Maximum warps per SM (typical for modern GPUs)
    'warp_size': 64,             # AMD wavefront size (64) vs NVIDIA warp size (32)
    'max_threads_per_sm': 2560,  # Maximum threads per SM
    'num_sms': 4,               # Number of streaming multiprocessors/compute units #64
    'max_blocks_per_sm': 40,     # Maximum blocks per SM. changed from 32
    'shared_memory_per_sm': 65536, # Shared memory per SM in bytes
    'registers_per_sm': 8192,   # Number of registers per SM
    'peak_mem_bandwidth_gbps': 50,  #default used to be 900 GB/s, but that is for modern
}
```

Thread-based Occupancy



```
system.cpu3.CUs1.vALUInsts                  264352                        # Number of vector ALU insts issued. (Unspecified)
system.cpu3.CUs1.vALUInstsPerWF                 16                        # The avg. number of vector ALU insts issued per-wavefront. (Unspecified)
system.cpu3.CUs1.sALUInsts                   66088                        # Number of scalar ALU insts issued. (Unspecified)
system.cpu3.CUs1.sALUInstsPerWF                  4                        # The avg. number of scalar ALU insts issued per-wavefront. (Unspecified)
system.cpu3.CUs1.instCyclesVALU             264352                        # Number of cycles needed to execute VALU insts. (Unspecified)
system.cpu3.CUs1.instCyclesSALU              66088                        # Number of cycles needed to execute SALU insts. (Unspecified)
system.cpu3.CUs1.threadCyclesVALU          4229632                        # Number of thread cycles used to execute vector ALU ops. Similar to instC
system.cpu3.CUs1.vALUUtilization                25                        # Percentage of active vector ALU threads in a wave. (Unspecified)
system.cpu3.CUs1.ldsNoFlatInsts                  0                        # Number of LDS insts issued, not including FLAT accesses that resolve to
```

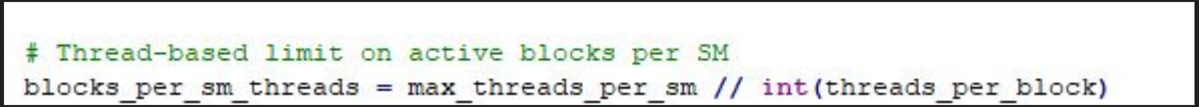


```
system.cpu3.CUs1.numFailedCASOps                   0                      # number of compare and swap operations that faile
system.cpu3.CUs1.completedWfs                  16522                      # number of completed wavefronts (Unspecified)
system.cpu3.CUs1.completedWGs                  16522                      # number of completed workgroups (Unspecified)
system.cpu3.CUs1.headTailLatency::bucket_size        10000                   # ticks between first and last cache block ar
```

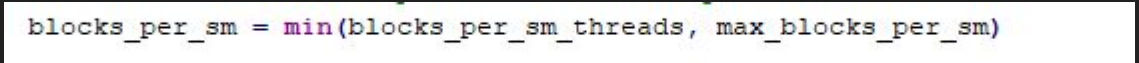
- Thread occupancy = active threads per CU / max threads per CU
- Thread occupancy = [(%active_threads)*(wavefront_size)*(wavefronts_per_block)*(blocks_per_cu)] **/** (max_threads_per_cu)

- %active_threads = vALUUtilization value in stats file
- Wavefront_size = warp_size in gpu_config = 64
- Wavefronts_per_block = # completed wavefronts / # completed workgroups
- Blocks_per_cu = min(blocks_per_sm_threads, max_blocks_per_sm)
- Blocks_per_sm_threads = max_threads_per_sm / threads_per_block
- Threads_per_block = wavefronts_per_block * wavefront_size
- Max_blocks_per_sm = parameter in gpu_config = 40
- Max_threads_per_sm = parameter in gpu_config = 2560

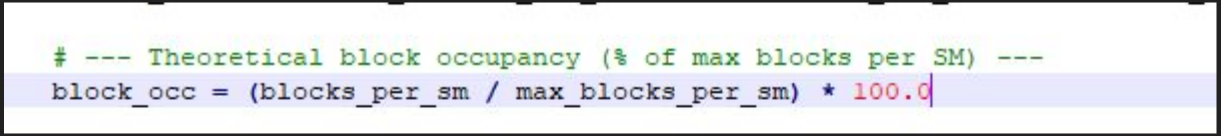
Block-based Occupancy



```
# Thread-based limit on active blocks per SM
blocks_per_sm_threads = max_threads_per_sm // int(threads_per_block)
```



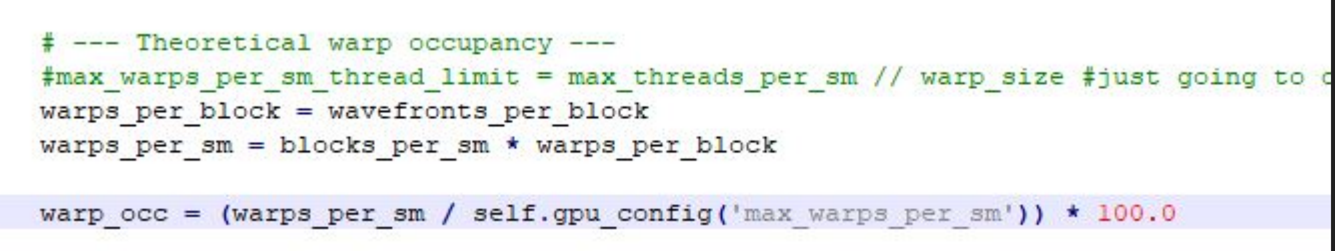
```
blocks_per_sm = min(blocks_per_sm_threads, max_blocks_per_sm)
```



```
  # --- Theoretical block occupancy (% of max blocks per SM) ---
  block_occ = (blocks_per_sm / max_blocks_per_sm) * 100.0
```

- Block occupancy = blocks per CU / max blocks per CU
-
- Wavefront_size = warp_size in gpu_config = 64
- Wavefronts_per_block = # completed wavefronts / # completed workgroups
- Blocks_per_cu = min(blocks_per_sm_threads, max_blocks_per_sm)
- Blocks_per_sm_threads = max_threads_per_sm / threads_per_block
- Threads_per_block = wavefronts_per_block * wavefront_size
- Max_blocks_per_sm = parameter in gpu_config = 40
- Max_threads_per_sm = parameter in gpu_config = 2560

Theoretical Warp-based Occupancy

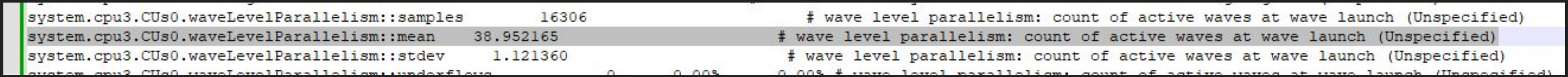


```
# --- Theoretical warp occupancy ---
#max_warps_per_sm_thread_limit = max_threads_per_sm // warp_size #just going to c
warps_per_block = wavefronts_per_block
warps_per_sm = blocks_per_sm * warps_per_block

warp_occ = (warps_per_sm / self.gpu_config('max_warps_per_sm')) * 100.0
```

- Warp occupancy = active warps / max warps per CU

- Warps_per_sm = blocks_per_cu * wavefronts_per_block
- Wavefront_size = warp_size in gpu_config = 64
- Wavefronts_per_block = # completed wavefronts / # completed workgroups
- Blocks_per_cu = min(blocks_per_sm_threads, max_blocks_per_sm)
- Blocks_per_sm_threads = max_threads_per_sm / threads_per_block
- Threads_per_block = wavefronts_per_block * wavefront_size
- Max_blocks_per_sm = parameter in gpu_config = 40
- Max_threads_per_sm = parameter in gpu_config = 2560

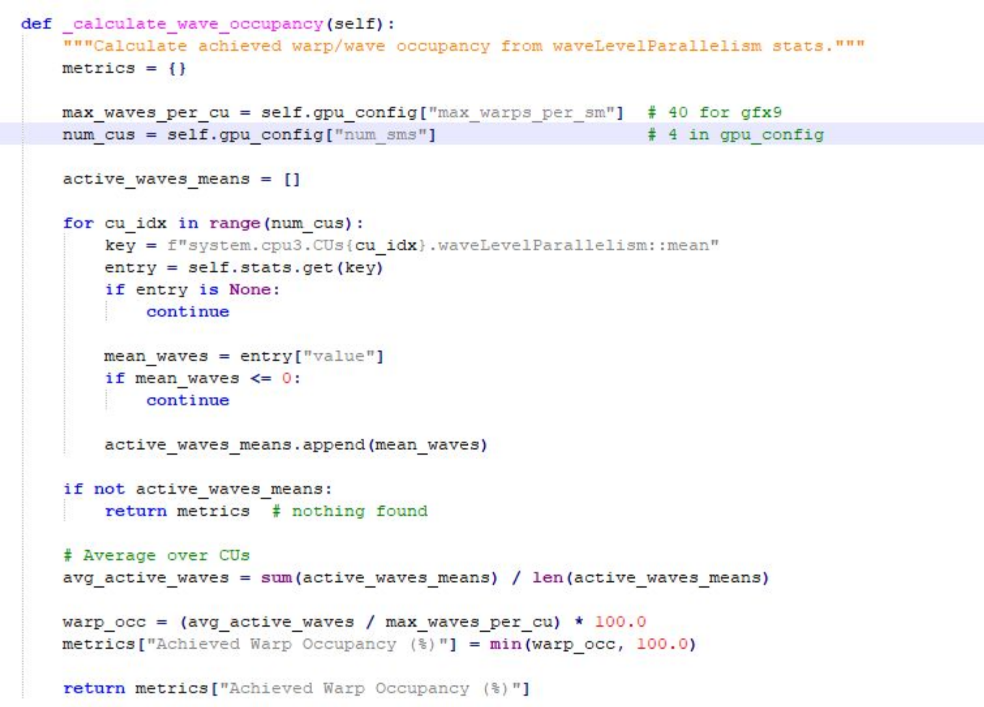
Achieved Warp-based Occupancy



```
system.cpu3.CUs0.waveLevelParallelism::samples          16306                              # wave level parallelism: count of active waves at wave launch (Unspecified)
system.cpu3.CUs0.waveLevelParallelism::mean    38.952165                              # wave level parallelism: count of active waves at wave launch (Unspecified)
system.cpu3.CUs0.waveLevelParallelism::stdev     1.121360                              # wave level parallelism: count of active waves at wave launch (Unspecified)
system.cpu3.CUs0.waveLevelParallelism::underflows            0      0.00%      0.00% # wave level parallelism: count of active waves at wave launch (Unspecified)
```

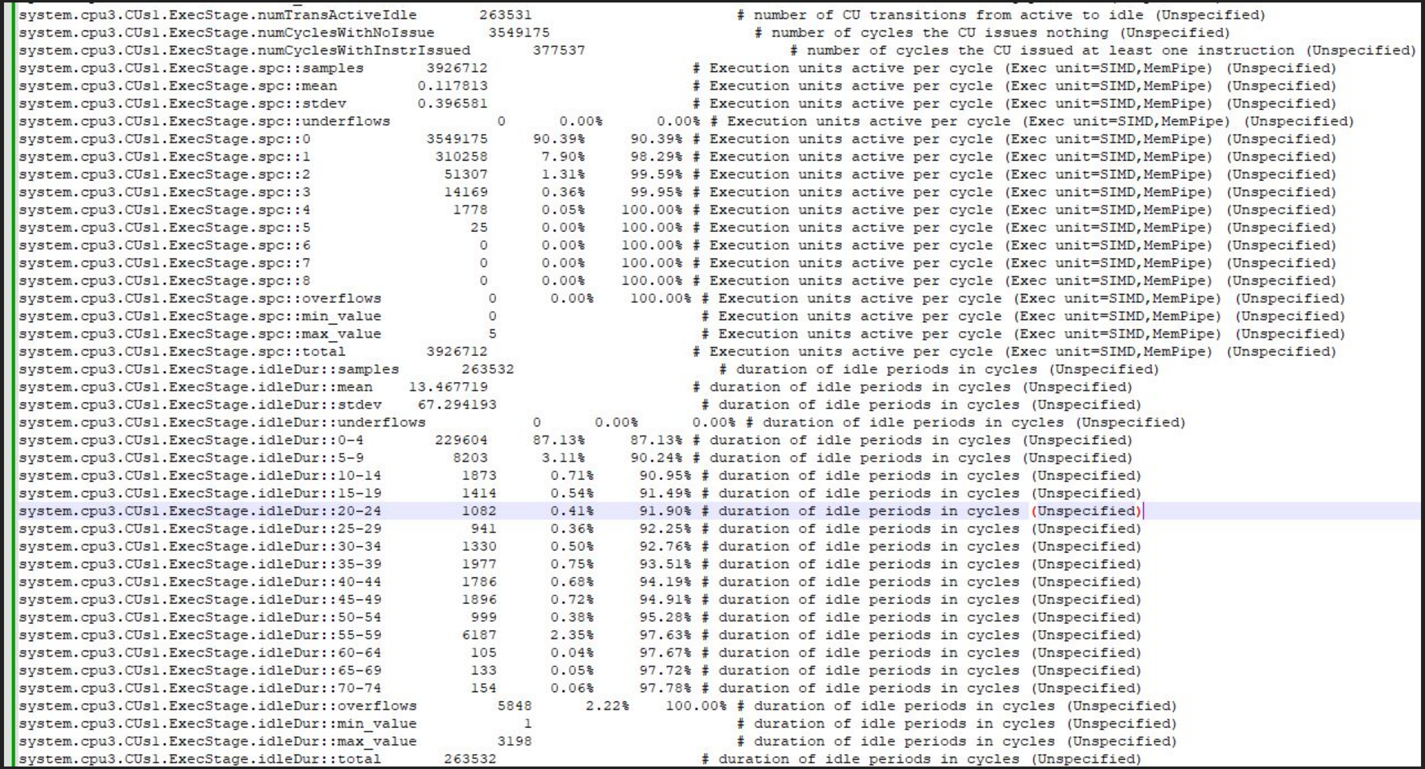
- Achieved Warp occupancy = achieved active warps / max warps per CU

- waveLevelParallelism::mean avg over all CUs = achieved active wavefronts



```python
def _calculate_wave_occupancy(self):
    """Calculate achieved warp/wave occupancy from waveLevelParallelism stats."""
    metrics = {}

    max_waves_per_cu = self.gpu_config["max_warps_per_sm"]  # 40 for gfx9
    num_cus = self.gpu_config["num_sms"]                      # 4 in gpu_config

    active_waves_means = []

    for cu_idx in range(num_cus):
        key = f"system.cpu3.CUs{cu_idx}.waveLevelParallelism::mean"
        entry = self.stats.get(key)
        if entry is None:
            continue

        mean_waves = entry["value"]
        if mean_waves <= 0:
            continue

        active_waves_means.append(mean_waves)

    if not active_waves_means:
        return metrics  # nothing found

    # Average over CUs
    avg_active_waves = sum(active_waves_means) / len(active_waves_means)

    warp_occ = (avg_active_waves / max_waves_per_cu) * 100.0
    metrics["Achieved Warp Occupancy (%)"] = min(warp_occ, 100.0)

    return metrics["Achieved Warp Occupancy (%)"]
```

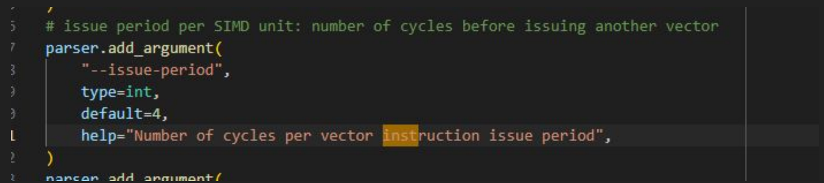
Cycle-based Occupancy / SM Utilization



```
system.cpu3.CUs1.ExecStage.numTransActiveIdle       263531                       # number of CU transitions from active to idle (Unspecified)
system.cpu3.CUs1.ExecStage.numCyclesWithNoIssue      3549175                       # number of cycles the CU issues nothing (Unspecified)
system.cpu3.CUs1.ExecStage.numCyclesWithInstrIssued       377537                       # number of cycles the CU issued at least one instruction (Unspecified)
system.cpu3.CUs1.ExecStage.spc::samples       3926712                       # Execution units active per cycle (Exec unit=SIMD,MemPipe) (Unspecified)
system.cpu3.CUs1.ExecStage.spc::mean         0.117813                       # Execution units active per cycle (Exec unit=SIMD,MemPipe) (Unspecified)
system.cpu3.CUs1.ExecStage.spc::stdev        0.396581                       # Execution units active per cycle (Exec unit=SIMD,MemPipe) (Unspecified)
system.cpu3.CUs1.ExecStage.spc::underflows            0      0.00%      0.00% # Execution units active per cycle (Exec unit=SIMD,MemPipe) (Unspecified)
system.cpu3.CUs1.ExecStage.spc::0             3549175     90.39%     90.39% # Execution units active per cycle (Exec unit=SIMD,MemPipe) (Unspecified)
system.cpu3.CUs1.ExecStage.spc::1              310258      7.90%     98.29% # Execution units active per cycle (Exec unit=SIMD,MemPipe) (Unspecified)
system.cpu3.CUs1.ExecStage.spc::2               51307      1.31%     99.59% # Execution units active per cycle (Exec unit=SIMD,MemPipe) (Unspecified)
system.cpu3.CUs1.ExecStage.spc::3               14169      0.36%     99.95% # Execution units active per cycle (Exec unit=SIMD,MemPipe) (Unspecified)
system.cpu3.CUs1.ExecStage.spc::4                1778      0.05%    100.00% # Execution units active per cycle (Exec unit=SIMD,MemPipe) (Unspecified)
system.cpu3.CUs1.ExecStage.spc::5                  25      0.00%    100.00% # Execution units active per cycle (Exec unit=SIMD,MemPipe) (Unspecified)
system.cpu3.CUs1.ExecStage.spc::6                   0      0.00%    100.00% # Execution units active per cycle (Exec unit=SIMD,MemPipe) (Unspecified)
system.cpu3.CUs1.ExecStage.spc::7                   0      0.00%    100.00% # Execution units active per cycle (Exec unit=SIMD,MemPipe) (Unspecified)
system.cpu3.CUs1.ExecStage.spc::8                   0      0.00%    100.00% # Execution units active per cycle (Exec unit=SIMD,MemPipe) (Unspecified)
system.cpu3.CUs1.ExecStage.spc::overflows            0      0.00%    100.00% # Execution units active per cycle (Exec unit=SIMD,MemPipe) (Unspecified)
system.cpu3.CUs1.ExecStage.spc::min_value            0                       # Execution units active per cycle (Exec unit=SIMD,MemPipe) (Unspecified)
system.cpu3.CUs1.ExecStage.spc::max_value            5                       # Execution units active per cycle (Exec unit=SIMD,MemPipe) (Unspecified)
system.cpu3.CUs1.ExecStage.spc::total         3926712                       # Execution units active per cycle (Exec unit=SIMD,MemPipe) (Unspecified)
system.cpu3.CUs1.ExecStage.idleDur::samples       263532                        # duration of idle periods in cycles (Unspecified)
system.cpu3.CUs1.ExecStage.idleDur::mean    13.467719                       # duration of idle periods in cycles (Unspecified)
system.cpu3.CUs1.ExecStage.idleDur::stdev    67.294193                       # duration of idle periods in cycles (Unspecified)
system.cpu3.CUs1.ExecStage.idleDur::underflows            0      0.00%      0.00% # duration of idle periods in cycles (Unspecified)
system.cpu3.CUs1.ExecStage.idleDur::0-4        229604     87.13%     87.13% # duration of idle periods in cycles (Unspecified)
system.cpu3.CUs1.ExecStage.idleDur::5-9          8203      3.11%     90.24% # duration of idle periods in cycles (Unspecified)
system.cpu3.CUs1.ExecStage.idleDur::10-14         1873      0.71%     90.95% # duration of idle periods in cycles (Unspecified)
system.cpu3.CUs1.ExecStage.idleDur::15-19         1414      0.54%     91.49% # duration of idle periods in cycles (Unspecified)
system.cpu3.CUs1.ExecStage.idleDur::20-24         1082      0.41%     91.90% # duration of idle periods in cycles (Unspecified)
system.cpu3.CUs1.ExecStage.idleDur::25-29          941      0.36%     92.25% # duration of idle periods in cycles (Unspecified)
system.cpu3.CUs1.ExecStage.idleDur::30-34         1330      0.50%     92.76% # duration of idle periods in cycles (Unspecified)
system.cpu3.CUs1.ExecStage.idleDur::35-39         1977      0.75%     93.51% # duration of idle periods in cycles (Unspecified)
system.cpu3.CUs1.ExecStage.idleDur::40-44         1786      0.68%     94.19% # duration of idle periods in cycles (Unspecified)
system.cpu3.CUs1.ExecStage.idleDur::45-49         1896      0.72%     94.91% # duration of idle periods in cycles (Unspecified)
system.cpu3.CUs1.ExecStage.idleDur::50-54          999      0.38%     95.28% # duration of idle periods in cycles (Unspecified)
system.cpu3.CUs1.ExecStage.idleDur::55-59         6187      2.35%     97.63% # duration of idle periods in cycles (Unspecified)
system.cpu3.CUs1.ExecStage.idleDur::60-64          105      0.04%     97.67% # duration of idle periods in cycles (Unspecified)
system.cpu3.CUs1.ExecStage.idleDur::65-69          133      0.05%     97.72% # duration of idle periods in cycles (Unspecified)
system.cpu3.CUs1.ExecStage.idleDur::70-74          154      0.06%     97.78% # duration of idle periods in cycles (Unspecified)
system.cpu3.CUs1.ExecStage.idleDur::overflows         5848      2.22%    100.00% # duration of idle periods in cycles (Unspecified)
system.cpu3.CUs1.ExecStage.idleDur::min_value            1                       # duration of idle periods in cycles (Unspecified)
system.cpu3.CUs1.ExecStage.idleDur::max_value         3198                       # duration of idle periods in cycles (Unspecified)
system.cpu3.CUs1.ExecStage.idleDur::total       263532                       # duration of idle periods in cycles (Unspecified)
```

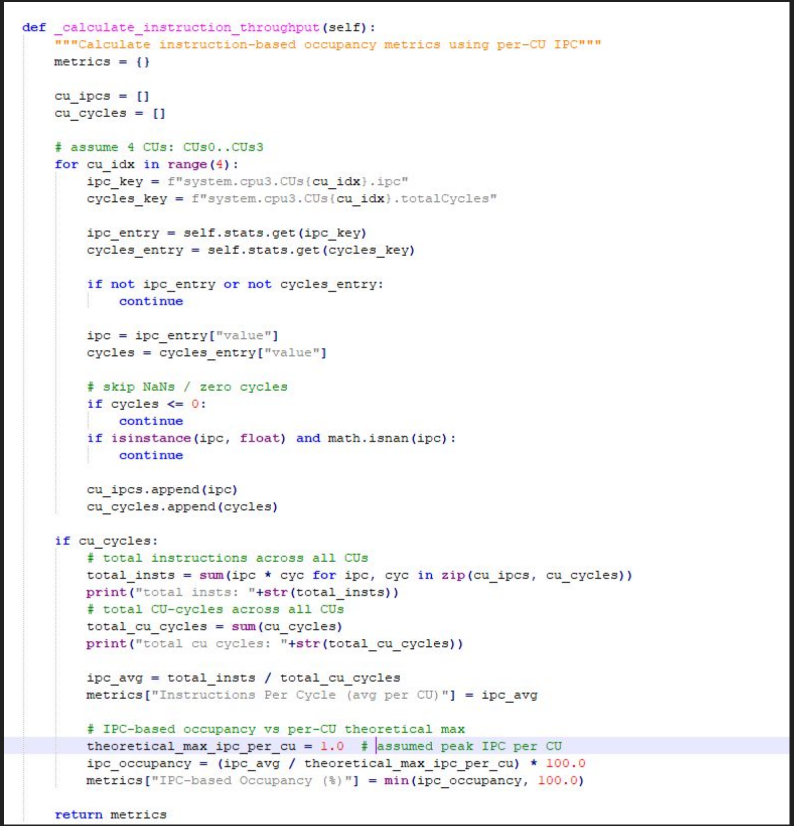
Cycle-based occupancy = cycles with instruction issued / total cycles

IPC-based Occupancy

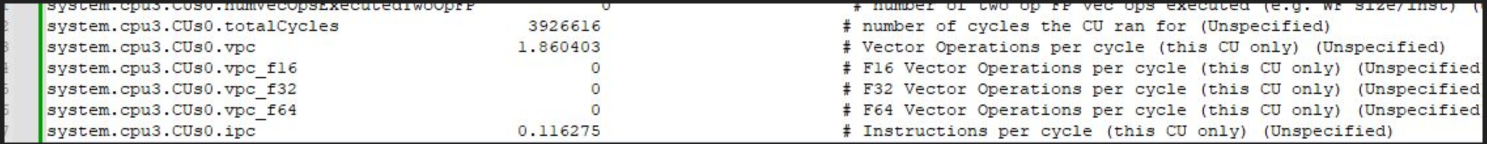


```
    ,
    # issue period per SIMD unit: number of cycles before issuing another vector
    parser.add_argument(
        "--issue-period",
        type=int,
        default=4,
        help="Number of cycles per vector instruction issue period",
    )
    parser.add_argument(
```

- Max IPC per CU = (1 / SIMD issue period) * (# SIMDs / CU) = (¼)(4) = 1 instr/cycle
- Achieved IPC = sum[(ipc * totalCycles) for each CU] / sum(totalcycles all CUs)



```
def _calculate_instruction_throughput(self):
    """Calculate instruction-based occupancy metrics using per-CU IPC"""
    metrics = {}

    cu_ipcs = []
    cu_cycles = []

    # assume 4 CUs: CUs0..CUs3
    for cu_idx in range(4):
        ipc_key = f"system.cpu3.CUs{cu_idx}.ipc"
        cycles_key = f"system.cpu3.CUs{cu_idx}.totalCycles"

        ipc_entry = self.stats.get(ipc_key)
        cycles_entry = self.stats.get(cycles_key)

        if not ipc_entry or not cycles_entry:
            continue

        ipc = ipc_entry["value"]
        cycles = cycles_entry["value"]

        # skip NaNs / zero cycles
        if cycles <= 0:
            continue
        if isinstance(ipc, float) and math.isnan(ipc):
            continue

        cu_ipcs.append(ipc)
        cu_cycles.append(cycles)

    if cu_cycles:
        # total instructions across all CUs
        total_insts = sum(ipc * cyc for ipc, cyc in zip(cu_ipcs, cu_cycles))
        print("total insts: "+str(total_insts))
        # total CU-cycles across all CUs
        total_cu_cycles = sum(cu_cycles)
        print("total cu cycles: "+str(total_cu_cycles))

        ipc_avg = total_insts / total_cu_cycles
        metrics["Instructions Per Cycle (avg per CU)"] = ipc_avg

        # IPC-based occupancy vs per-CU theoretical max
        theoretical_max_ipc_per_cu = 1.0  # assumed peak IPC per CU
        ipc_occupancy = (ipc_avg / theoretical_max_ipc_per_cu) * 100.0
        metrics["IPC-based Occupancy (%)"] = min(ipc_occupancy, 100.0)

    return metrics
```



```
system.cpu3.CUs0.numVecOpsExecutedTwoOpFP           0                          # number of two op FP vec ops executed (e.g. WF size/inst) (
system.cpu3.CUs0.totalCycles                  3926616                          # number of cycles the CU ran for (Unspecified)
system.cpu3.CUs0.vpc                         1.860403                          # Vector Operations per cycle (this CU only) (Unspecified)
system.cpu3.CUs0.vpc_f16                            0                          # F16 Vector Operations per cycle (this CU only) (Unspecified
system.cpu3.CUs0.vpc_f32                            0                          # F32 Vector Operations per cycle (this CU only) (Unspecified
system.cpu3.CUs0.vpc_f64                            0                          # F64 Vector Operations per cycle (this CU only) (Unspecified
system.cpu3.CUs0.ipc                         0.116275                          # Instructions per cycle (this CU only) (Unspecified)
```

MatrixTranspose with Varying Block Dimensions

(using gpu_occupancy_analyzer_v9.py)

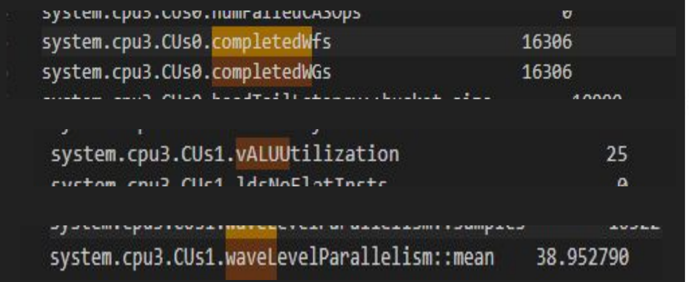
MatrixTranspose

Width=1024

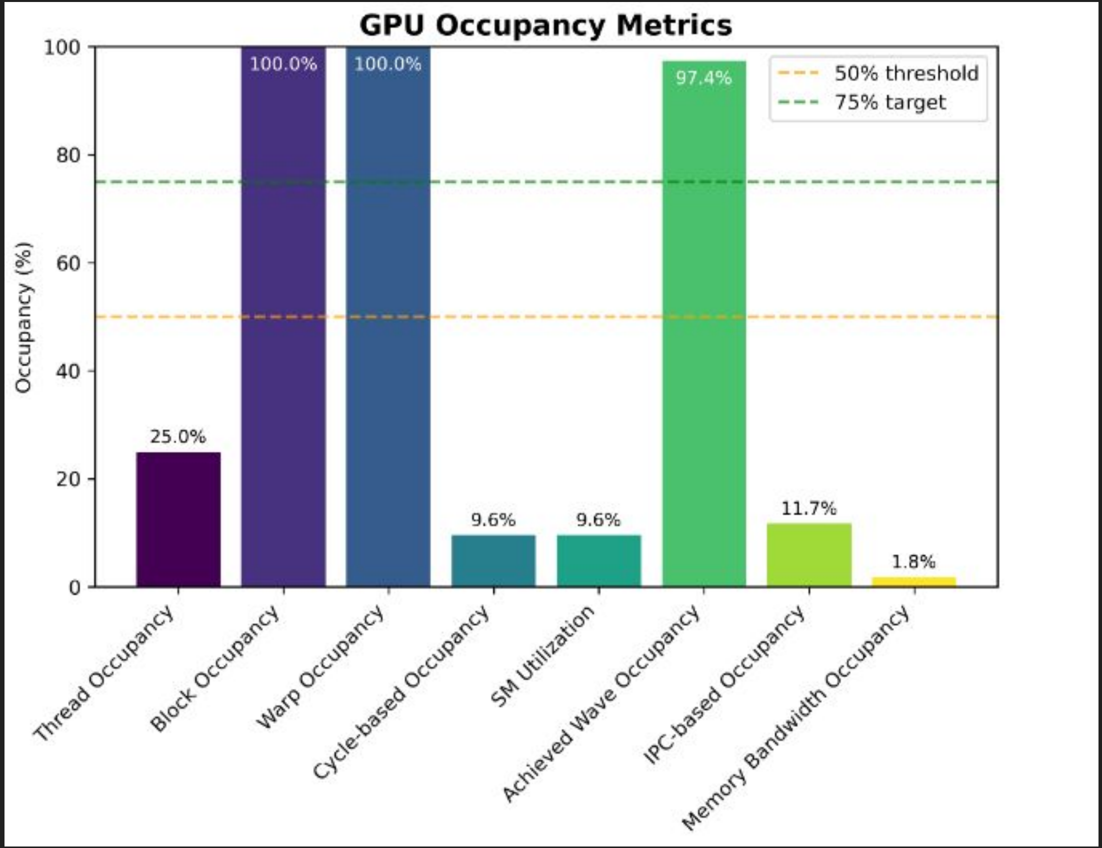
Threads per block X = 4

Threads per block Y = 4

Threads per block Z = 1



```
system.cpu3.CUs0.numFailedCASOps                    0
system.cpu3.CUs0.completedWfs                   16306
system.cpu3.CUs0.completedWGs                   16306
system.cpu3.CUs0.headTailLatency::bucket_size           10000

 system.cpu3.CUs1.vALUUtilization                          25
 system.cpu3.CUs1.ldsNoFlatInsts                            0

 system.cpu3.CUs1.waveLevelParallelism::samples         16322
 system.cpu3.CUs1.waveLevelParallelism::mean     38.952790
```



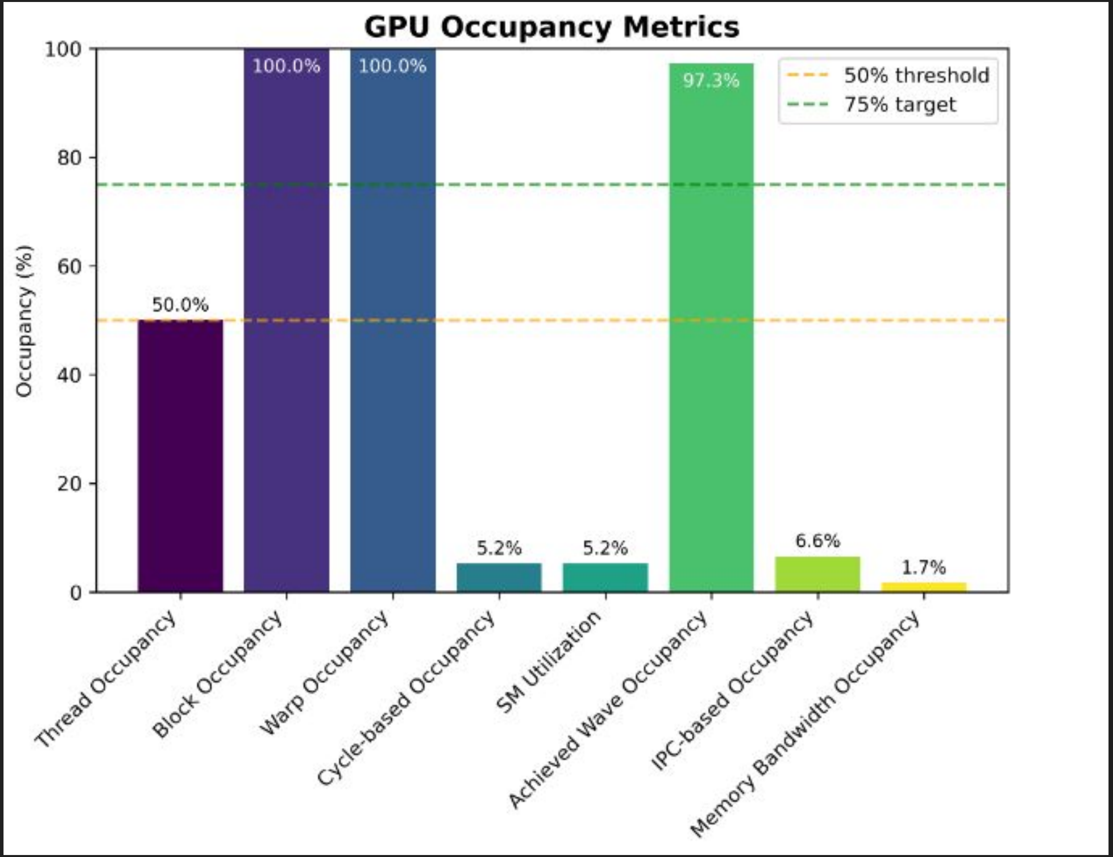
MatrixTranspose

Width=1024

Threads per block X = 8

Threads per block Y = 4

Threads per block Z = 1



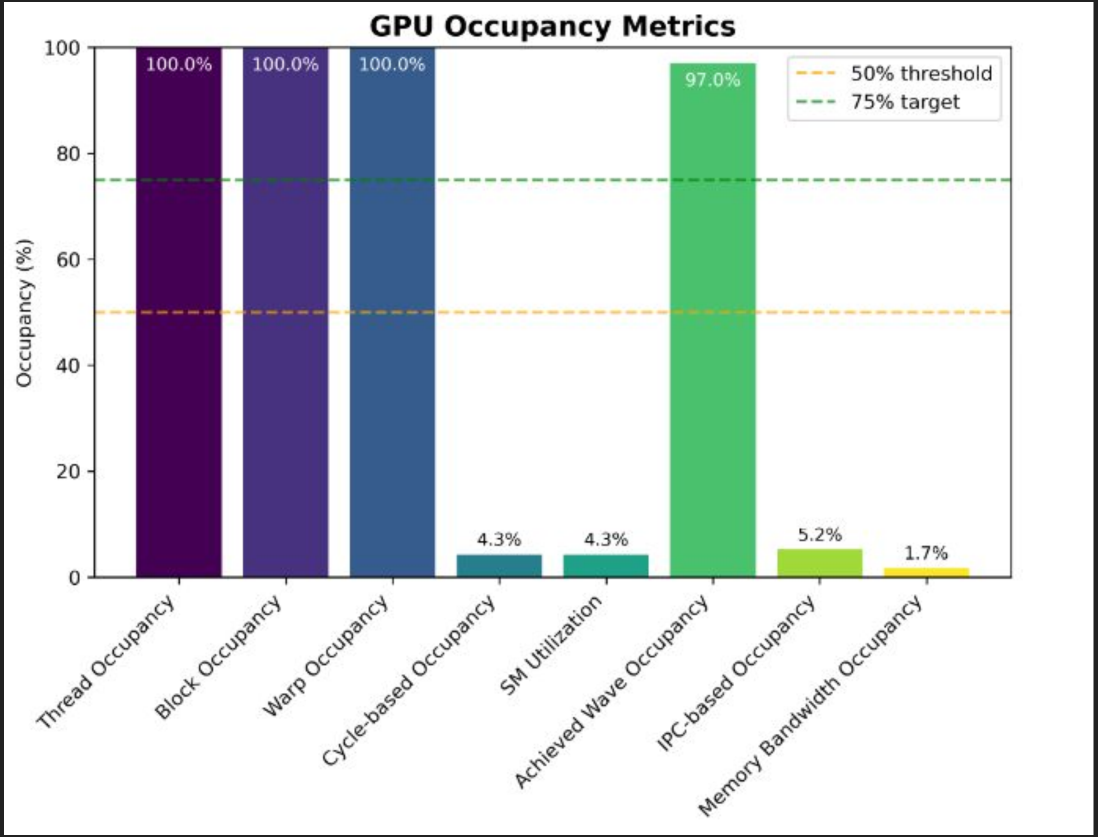
MatrixTranspose

Width=1024

Threads per block X = 8

Threads per block Y = 8

Threads per block Z = 1



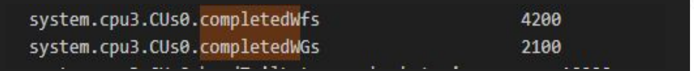
MatrixTranspose

Width=1024

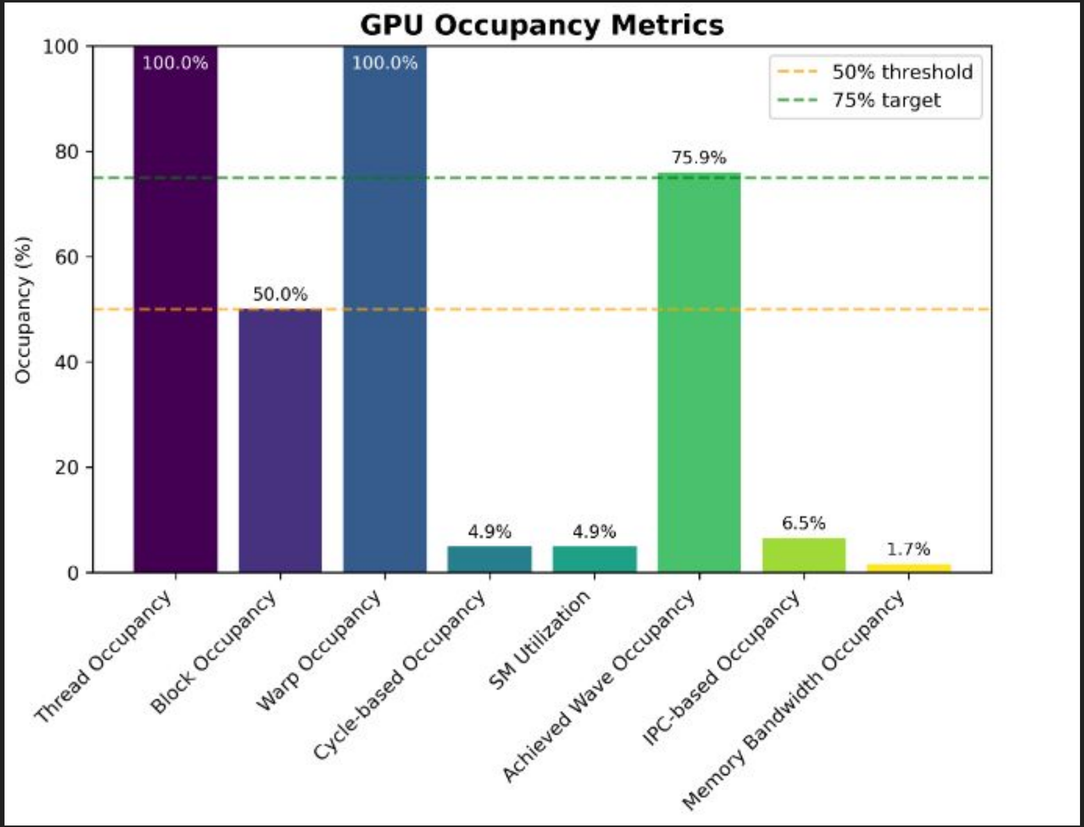
Threads per block X = 16

Threads per block Y = 8

Threads per block Z = 1

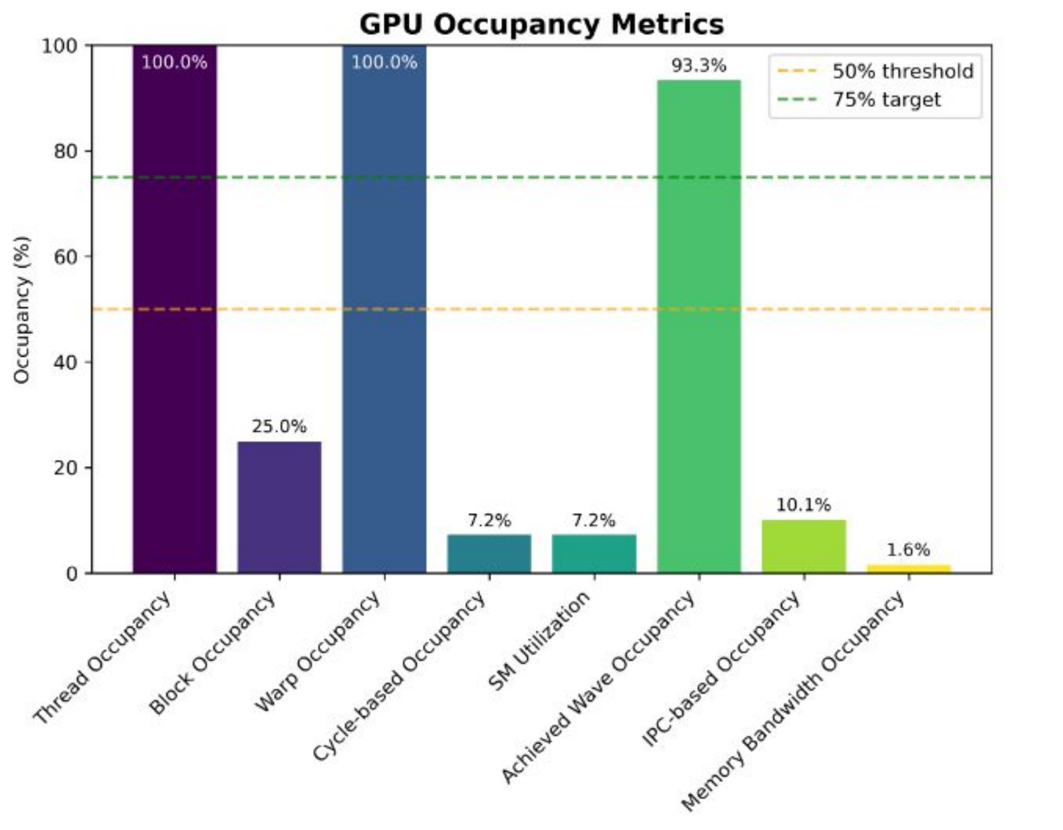


```
system.cpu3.CUs0.completedWfs                          4200
system.cpu3.CUs0.completedWGs                          2100
```



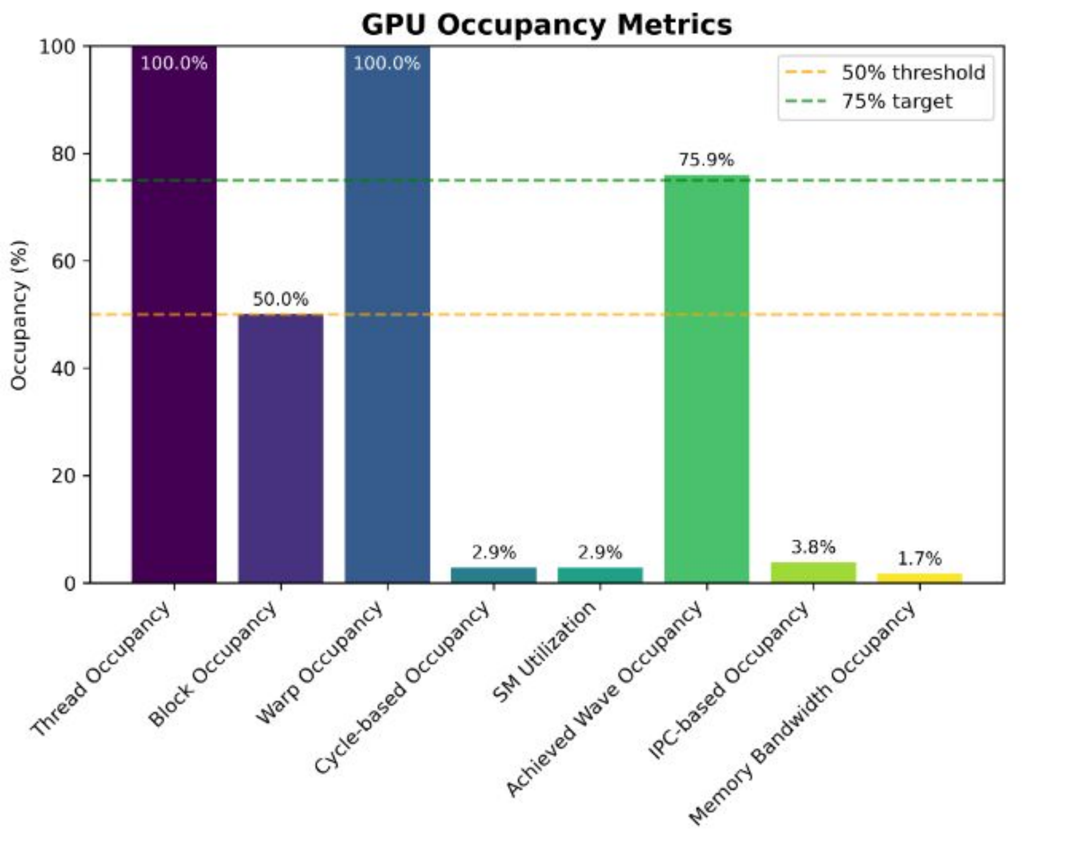
MatrixTranspose

Width=1024
Threads per block X = 16
Threads per block Y = 16
Threads per block Z = 1



MatrixTranspose

Width=1024
Threads per block X = 32
Threads per block Y = 4
Threads per block Z = 1



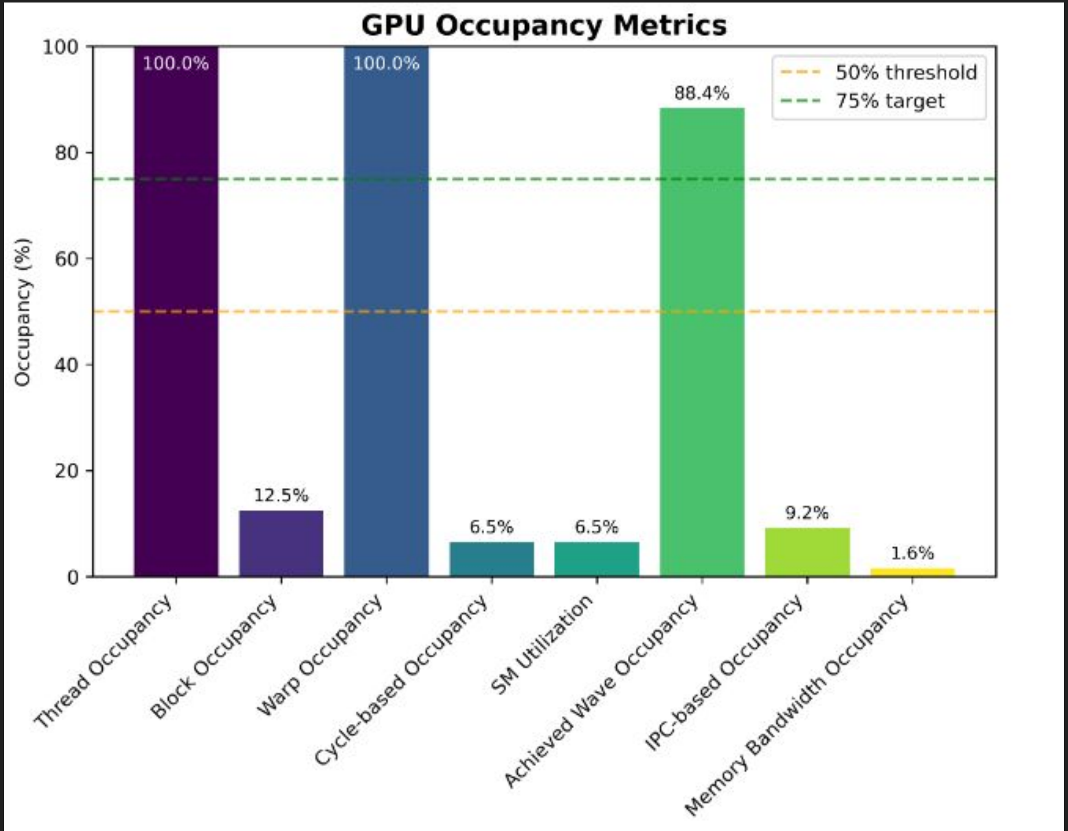
MatrixTranspose

Width=1024

Threads per block X = 32

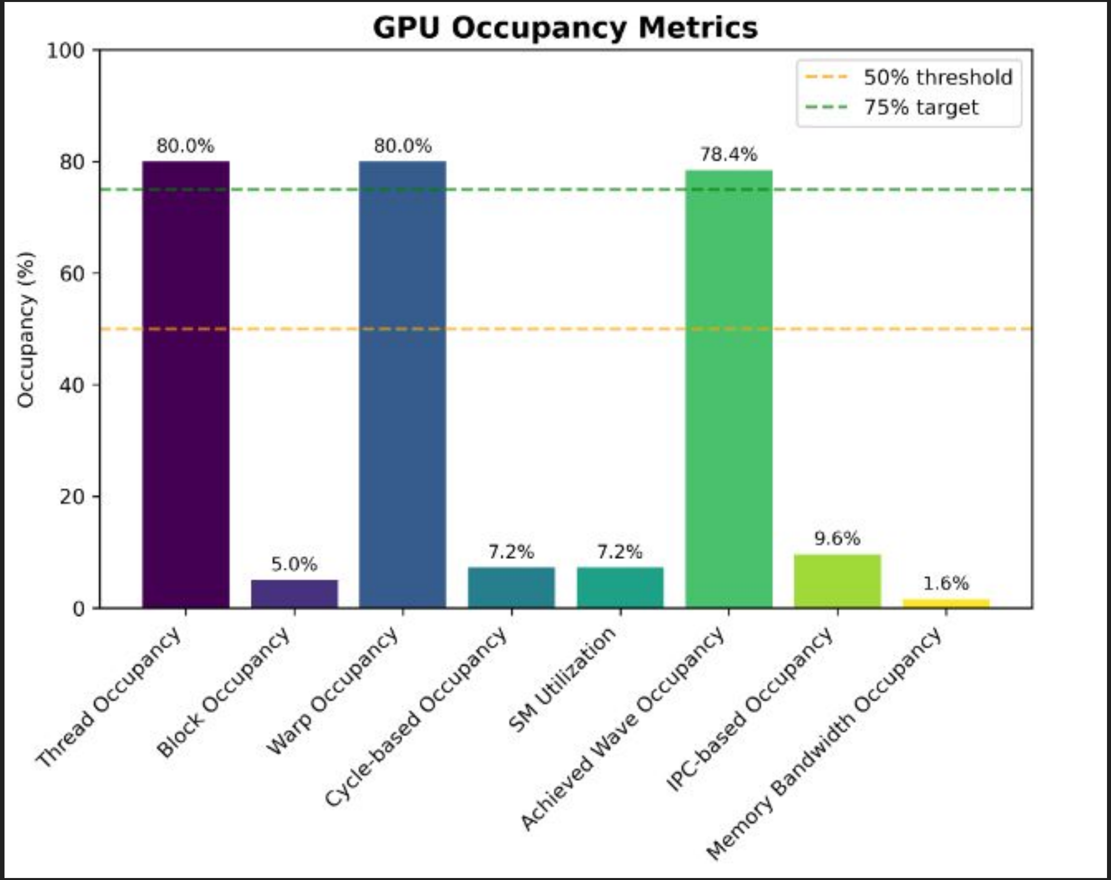
Threads per block Y = 16

Threads per block Z = 1

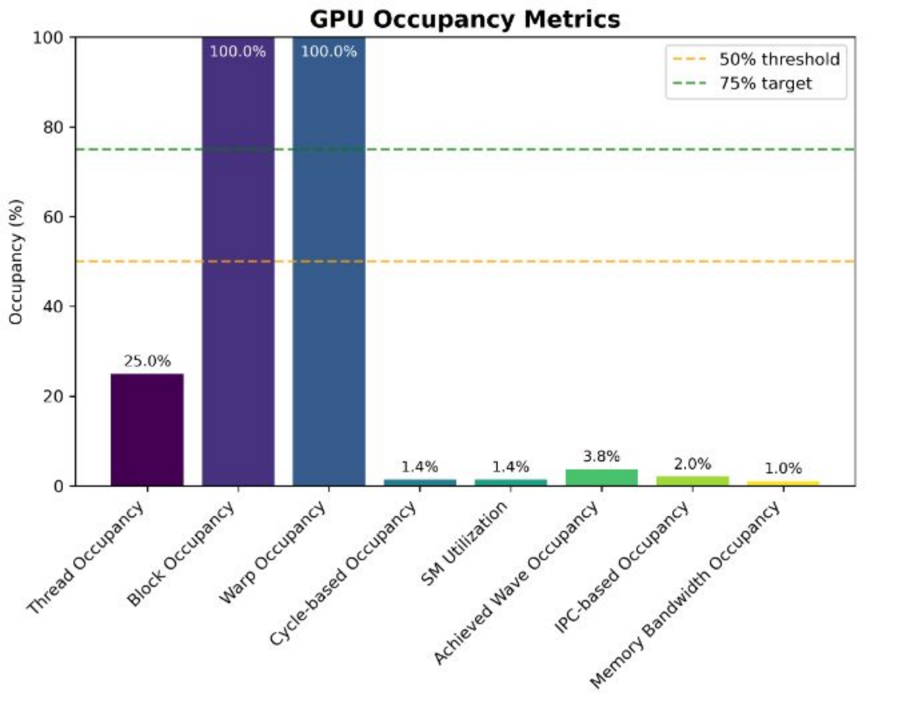


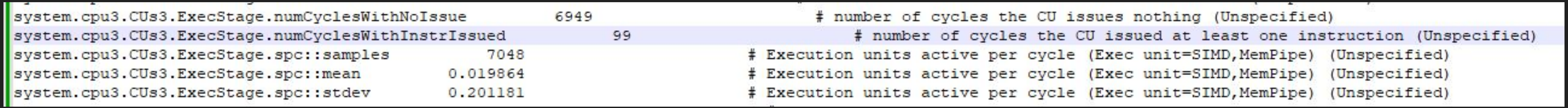
MatrixTranspose

Width=1024
Threads per block X = 32
Threads per block Y = 32
Threads per block Z = 1

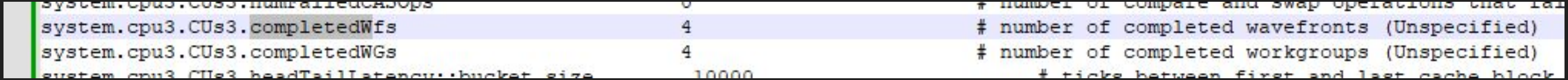


dynamic_shared



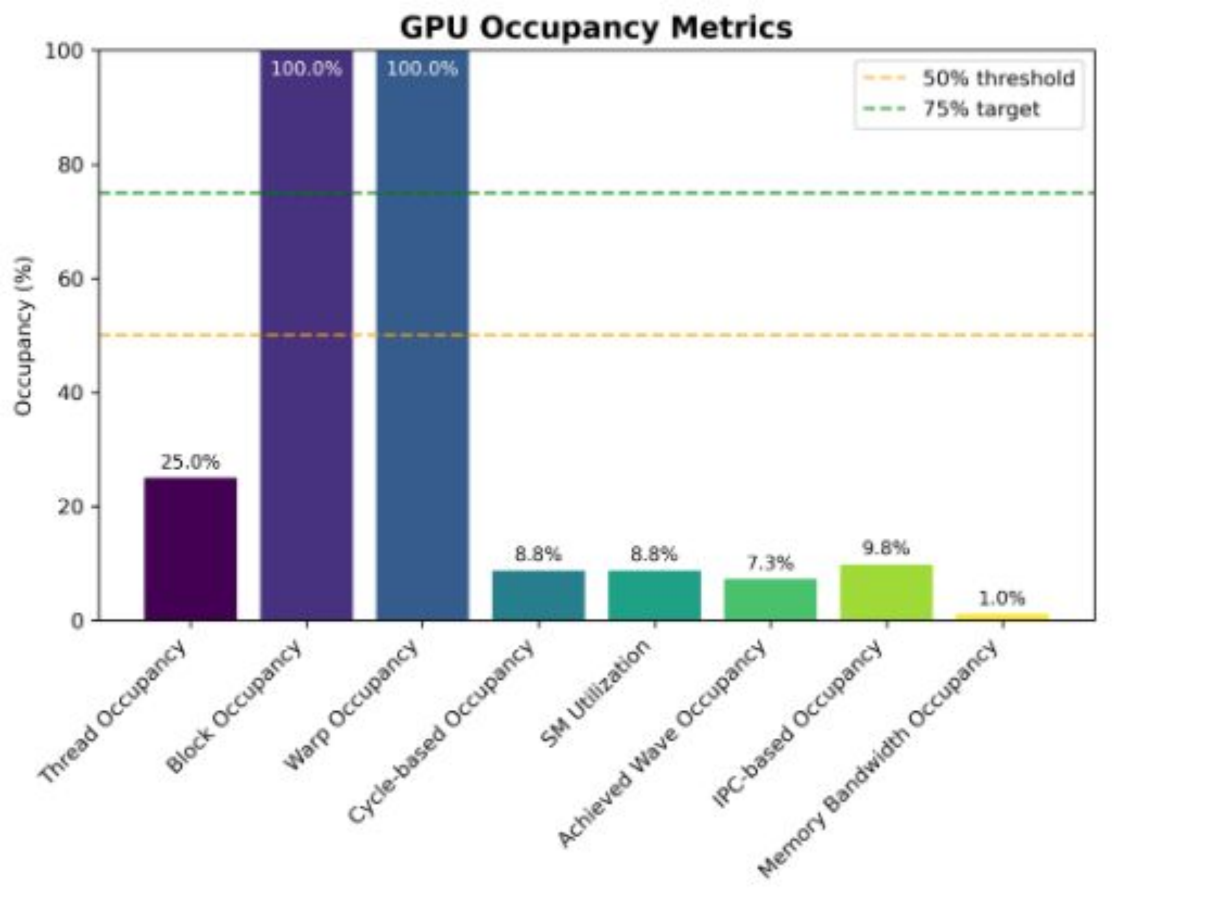


```
system.cpu3.CUs3.ExecStage.numCyclesWithNoIssue          6949                         # number of cycles the CU issues nothing (Unspecified)
system.cpu3.CUs3.ExecStage.numCyclesWithInstrIssued          99                         # number of cycles the CU issued at least one instruction (Unspecified)
system.cpu3.CUs3.ExecStage.spc::samples          7048                         # Execution units active per cycle (Exec unit=SIMD,MemPipe) (Unspecified)
system.cpu3.CUs3.ExecStage.spc::mean          0.019864                         # Execution units active per cycle (Exec unit=SIMD,MemPipe) (Unspecified)
system.cpu3.CUs3.ExecStage.spc::stdev          0.201181                         # Execution units active per cycle (Exec unit=SIMD,MemPipe) (Unspecified)
```



```
system.cpu3.CUs3.numFailedCASOps                     0                    # number of compare and swap operations that fai
system.cpu3.CUs3.completedWfs                        4                    # number of completed wavefronts (Unspecified)
system.cpu3.CUs3.completedWGs                        4                    # number of completed workgroups (Unspecified)
system.cpu3.CUs3.headTailLatency::bucket_size       10000                    # ticks between first and last cache block
```

sharedMemory



Conclusion

- Fixed most of the GPU occupancy calculations and stat parsing in the occupancy analyzer script
- Discussed occupancy formulas and important metrics from gem5 stats file
- Looked at how block size affects occupancy
- Resource utilization affecting occupancy
- Further work:
    - Assessing whether cycle based occupancy is calculated like it should be
    - Find the best metrics in the stats file for showing resource utilization and shared memory or LDS allocation and usage
        - Could help identify resource bottlenecks
    - Compare results of running on AMD hardware with the Radeon GPU profiler
- Github: https://github.com/alex-keist/EN525_712_gem5_gpu_occupancy

Sources

- https://www.olcf.ornl.gov/wp-content/uploads/2019/09/AMD_GPU_HIP_training_20190906.pdf
- https://gpuopen.com/learn/occupancy-explained/
- https://rocm.docs.amd.com/en/latest/reference/gpu-arch-specs.html
- https://rocm.docs.amd.com/projects/HIP/en/latest/index.html
- https://rocm.docs.amd.com/projects/HIP/en/latest/reference/hardware_features.html
- https://www.olcf.ornl.gov/wp-content/uploads/2019/10/ORNL_Application_Readiness_Workshop-AMD_GPU_Basics.pdf
- https://github.com/gem5bootcamp/2024/blob/main/slides/04-GPU-model/gpu-slides.pdf