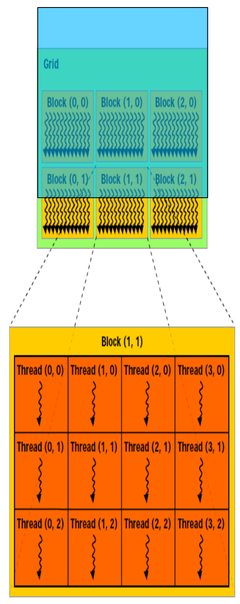
RLWS: A Reinforcement Learning Based GPU Warp Scheduler

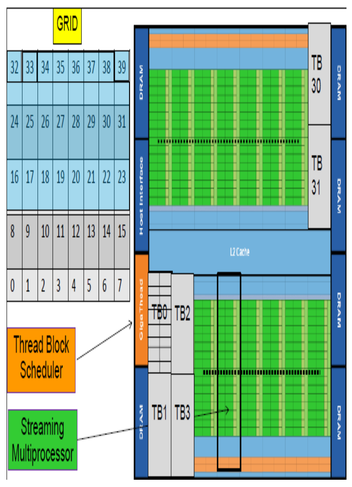
Jayvant Anantpur, Nagendra G. D.
Shivaram Kalyankrishnan, R. Govindarajan

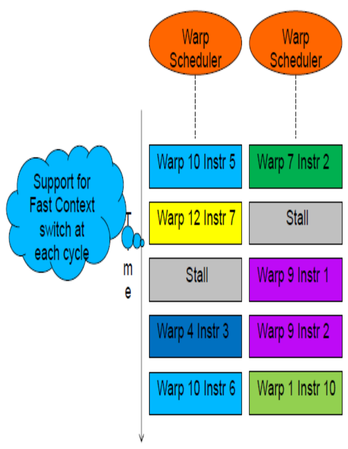
Problem:

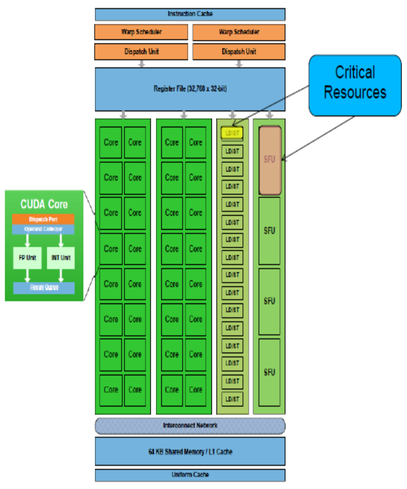
- At each cycle, schedule a warp from a pool of ready warps (satisfying Dependency and Resource Constraints)
- If no warp can be scheduled, the processor stalls
- Minimize Number of Stalls

- Code is executed on the GPU through Kernel calls
- Kernel calls specify execution configuration called Grid
- Grid specifies number of Thread Blocks (TB) and size of a TB
- Threads of a TB partitioned into groups of threads called Warps

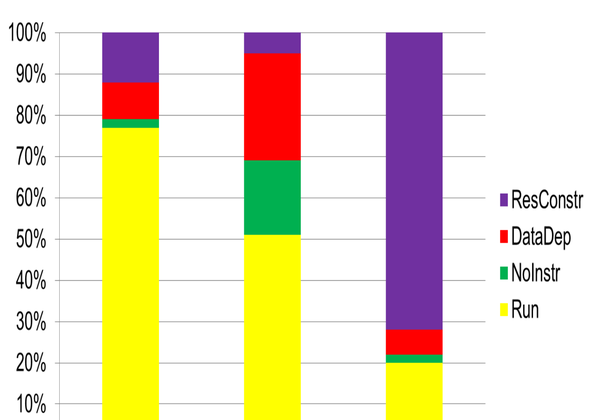






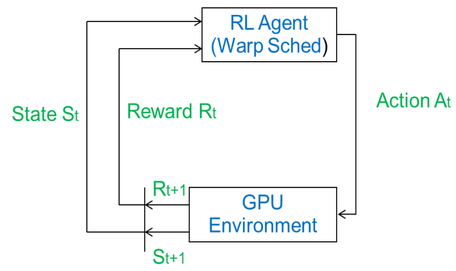


- Select a warp from a pool of ready warps
- Max pool size = 24, but ready warps may be fewer (due to dependence and resource constraints)
- 1 Instruction from a ready warp issued every cycle
- If no ready warp, then processor stalls
- Decision to be made every cycle
- Selecting a warp in each cycle, depends on the next instruction to be executed in the ready warps
- 3 different types of instruction pipelines
  - Memory (latency 300+ cycles for global mem)
  - Special Function (Latency ~20 – 100 cycles)
  - ALU (~10 cycles)
- **Objective:** Reduce the total runtime.



Why RL based Warp Scheduler

- Different warps (both within a TB and across TBs) execute the same code, i.e., same sequence of instructions
  - Except for data dependent execution paths
- SMs have seen execution of past TB
  - Each SM can hold only a few resident TBs, and new TBs come in the place of old (completed) TBs
- Intelligent scheduling needed to reduce stalls!
  - Need to hide long memory stalls of one warp with useful work from other warps!



State

- Total 33 Variables
- 10 binary – true/false values
- 23 variables can take multiple values
  - Value range is divided into buckets
  - Uniform/Non-uniform bucket sizes
  - Increasing/Decreasing bucket sizes
  - Boundary values (overlapping buckets)
- Number of ready warps of Various instruction types
- Various memory related statistics
- Warp synchronization state
- Number of stalled warps due to various stall reasons

Actions

- Select a type of command to execute
  - MEM, SFU, SP, NOP
- Select a type of TB
  - ANY_TB, FASTEST_TB, SLOWEST_TB, FINISH_TB, BARRIER_TB
- Hybrid scheduler with two agents

Rewards

- 1 if a warp is scheduled and 0 if none
- 1/-1
- Differential reward/penalties:
  - Reward = 3/2/1, 5/3/1, 10/5/1, 16/8/2 depending on the cmd type
  - Penalty = -2/-1/0, -4/-2/0, -10/-5/-1, -16/-8/-2 depending on the type of stall

- Q Value Table and Update Function

- Q value table – index computed using state and action values
  - XOR of lower $\log_2$(QtableSize) bits with upper bits
- Update function (SARSA)
  - $Q \leftarrow (1-\alpha) * Q + \alpha * (R + \gamma * Q_n)$;
- Same Q value table for both Warp Schedulers on an SM
- Various Q value table sizes from 256 to 1 million

- RL and Other Parameters

- ALPHA (Learning Rate)
  - (0.5/0.6/0.7/0.8/0.9) and reduced gradually
  - (0.01/0.03/0.05/0.07/0.09) fixed
- EXPL (Exploration)
  - (0.08/0.12/0.16/0.20) and reduced gradually
  - (0.01/0.03/0.05/0.07/0.09) fixed
- GAMMA (Discount Factor)
  - 0.95-0.99 and 0.999
- Ordering of warps with the same action
  - LRR, GTO, YOUNGEST,
- Number of consecutive no ops
  - Stall as a possible action even when there are ready warps
  - Number of consecutive no ops: 1, 2, 4, 8
- Frequency of selecting action
  - Every 1/2/4/8/16 cycles

Genetic Algorithm

- Very large design space
- To select state variables and their granularity (number of discrete values)
- To select RL and other parameters
- Number of solutions per generation = 100
- First generation solutions randomly chosen
  - Each solution is run on the RL implementation
  - Fitness value = Geometric mean of perf improvement over GTO (kernel times)
- Next generation selection:
  - 90 solutions generated using previous generation solutions using crossover and mutation
  - Parent solutions selected using Roulette wheel method
  - Every 10th generation introduces best 10 solutions seen so far
  - Other generations introduce 10 random solutions

Experimental Evaluation

- GPGPU-SIM to simulate CUDA benchmarks
- CUDA 4.2
- NVIDIA Fermi GPU architecture
- Benchmarks from GPGPU-SIM, Parboil, CUDA SDK and Rodinia benchmark suites

Results

- Used the best 10 RL configurations from GA
  - Used 15 kernels for learning the above configurations
- Ran 59 kernels and compared the speedup (over existing warp schedulers)
- Best RL Configuration gives
  - 5 % improvement over LRR
  - 7 % improvement over TL
  - 1 % slowdown w.r.t GTO
  - Best on 17 and second best on 30 kernels

RLWS Rank	RLWS_LRR	RLWS_GTO	RLWS_TL	Kernels
1	1.09	1.02	1.08	17
2	1.07	0.98	1.10	30
3	0.96	0.98	0.98	7
4	0.99	1.00	0.99	5