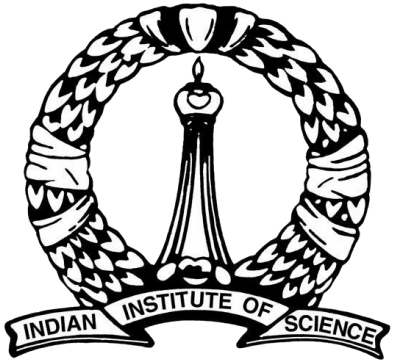
RLWS: A Reinforcement Learning Based GPU Warp Scheduler

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



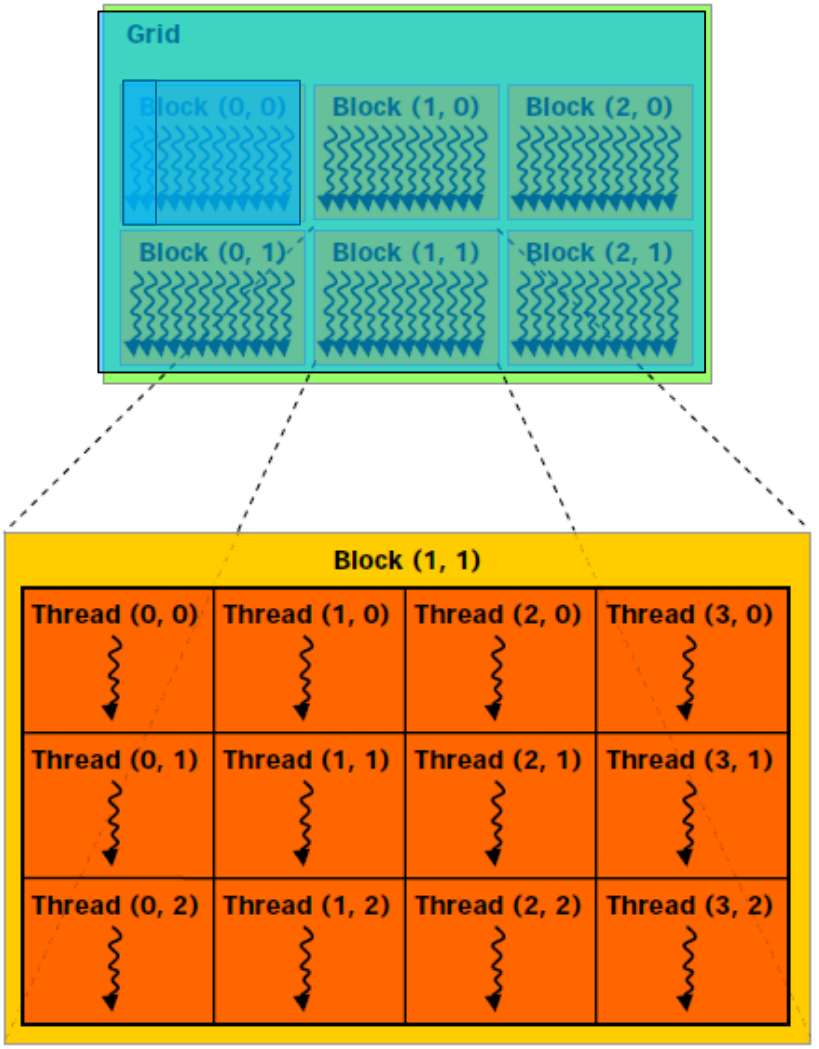
RLWS: A Reinforcement Learning based GPU Warp Scheduler

- **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
- **Objective:**
  - Minimize the number of stalls

CUDA Programming Model

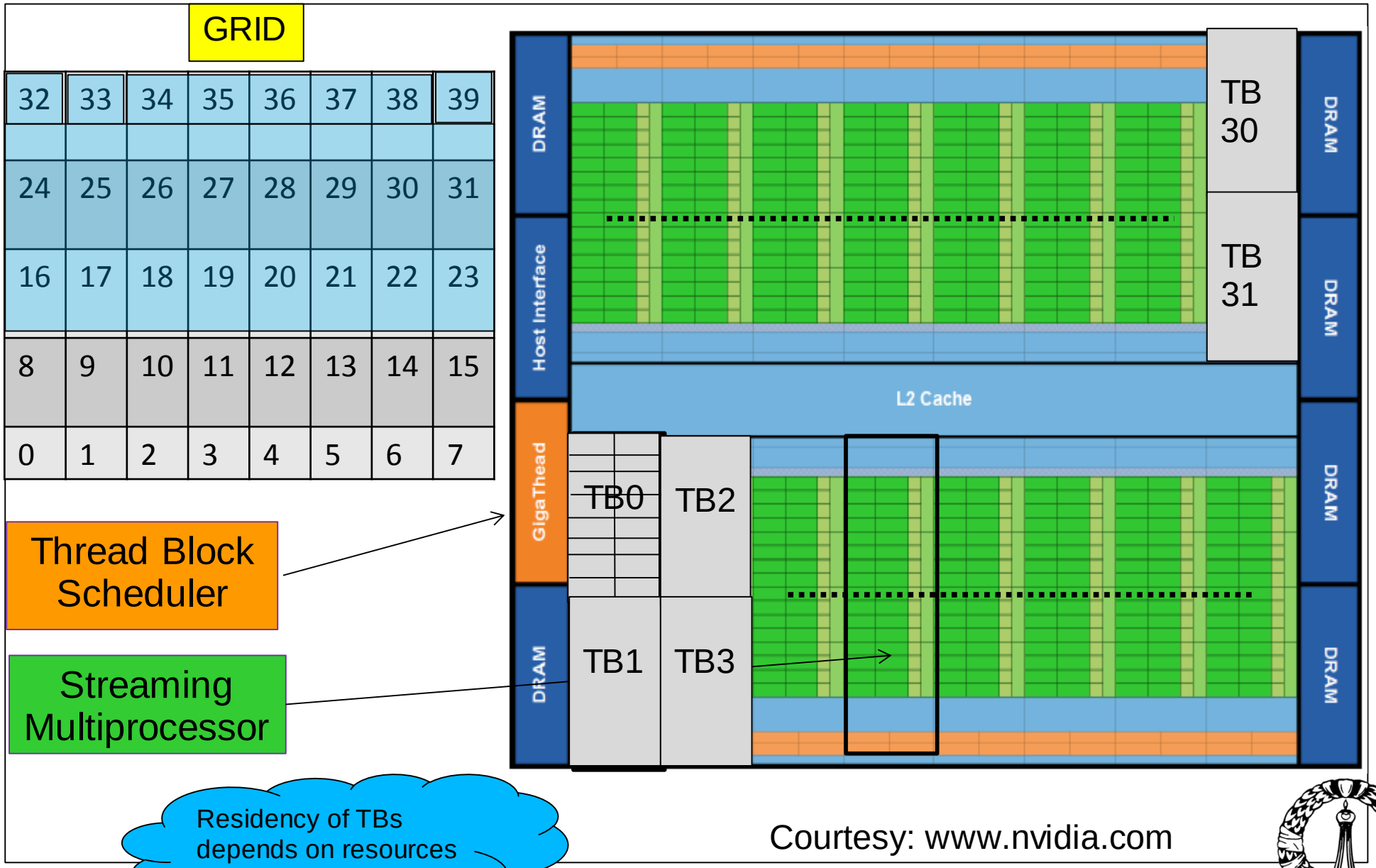
- Code is executed on the GPU through Kernel calls
- Kernel calls specify execution configuration called Grid
- Grid specifies number of Thread Blocks (TB) and size of a TB
- Threads of a TB partitioned into groups of threads called Warps

```
dim3 g(3, 2, 1);
dim3 b(4, 3, 1);
gpuKernel <<<g, b>>>(…);
```

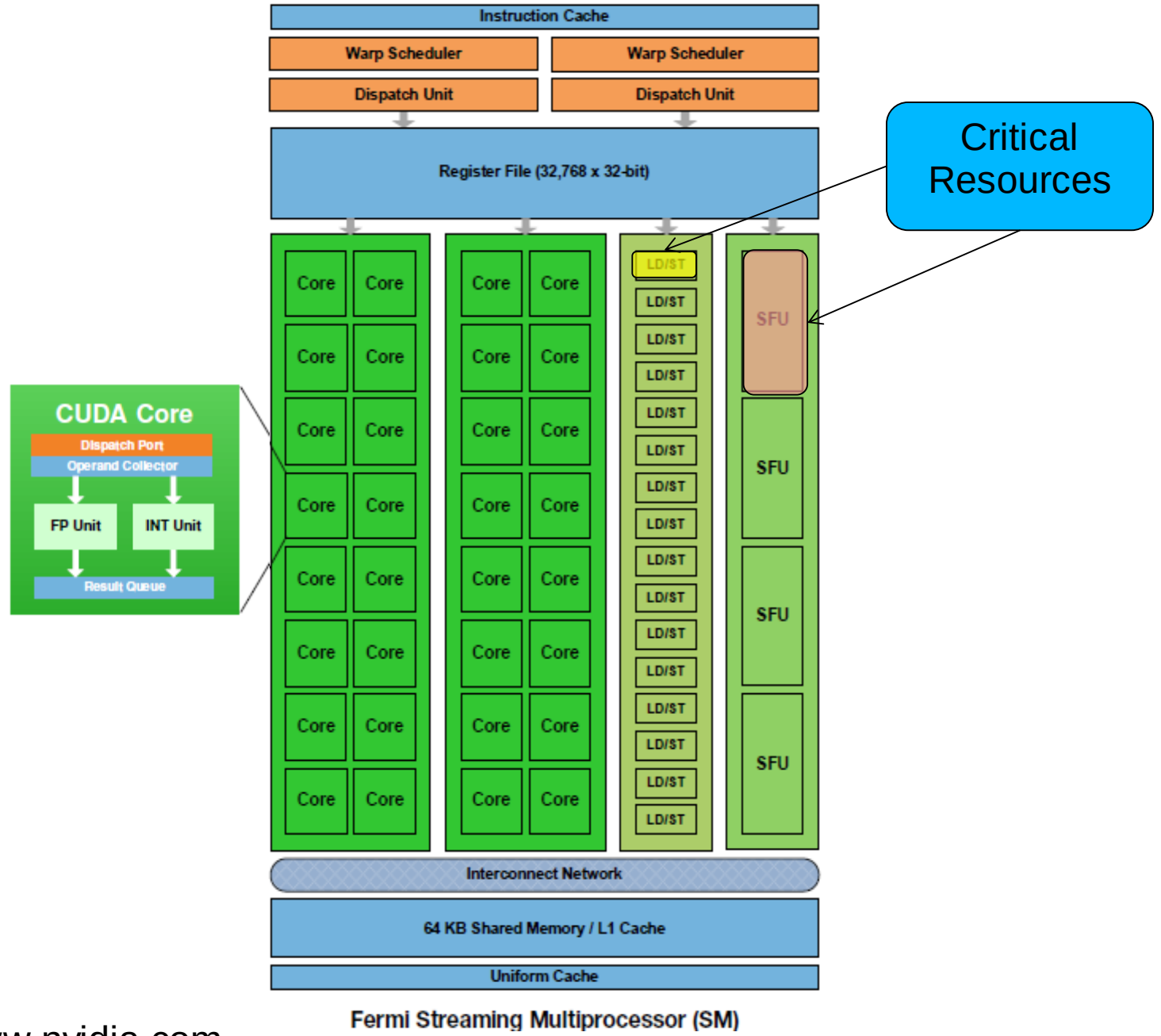


Courtesy: www.nvidia.com

Fermi GPU


GRID
32 33 34 35 36 37 38 39
24 25 26 27 28 29 30 31
16 17 18 19 20 21 22 23
8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7
Thread Block Scheduler
Streaming Multiprocessor
DRAM
Host Interface
GigaThead
L2 Cache
TB0
TB1
TB2
TB3
TB 30
TB 31
Residency of TBs depends on resources
Courtesy: www.nvidia.com

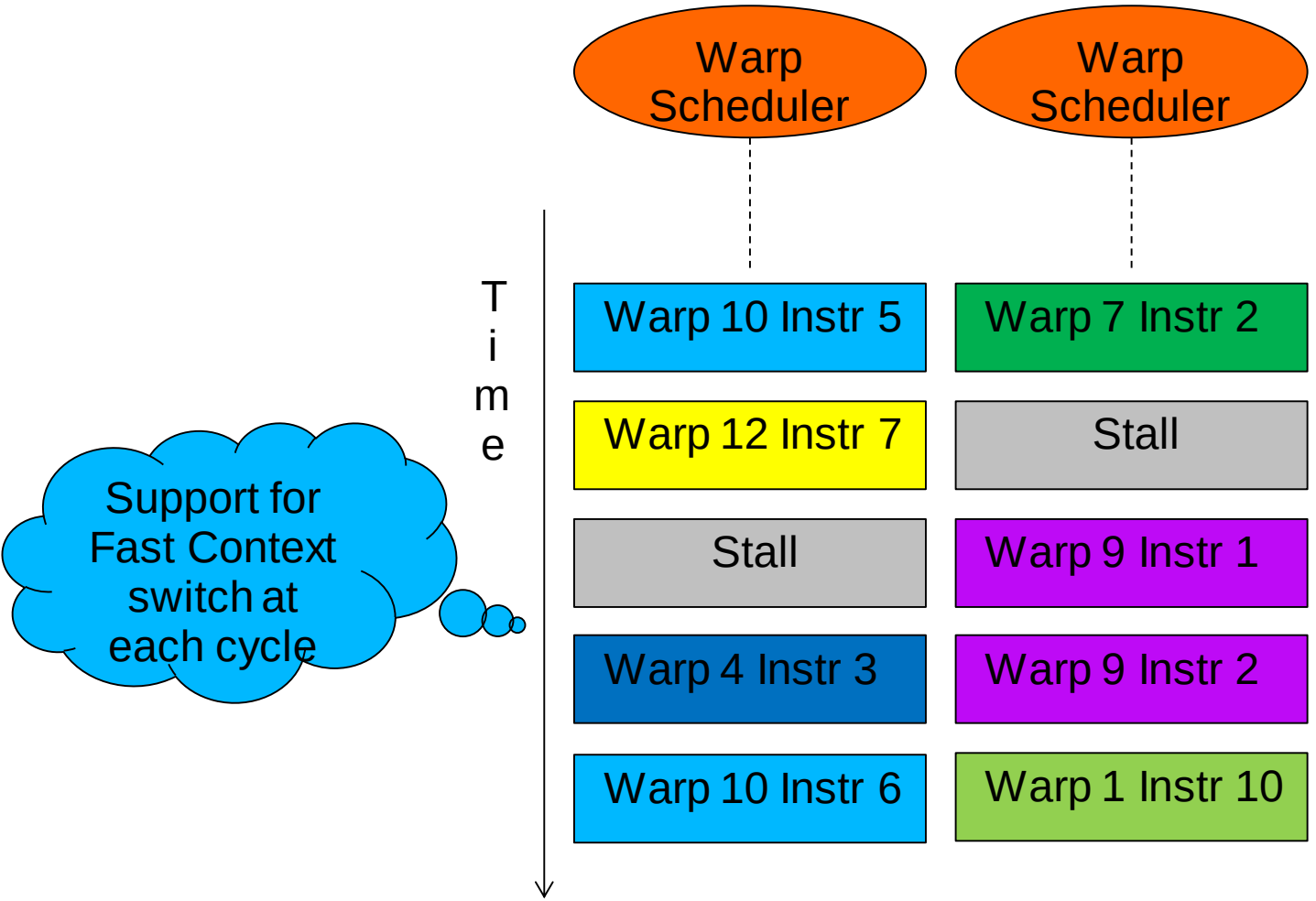
Streaming Multiprocessor



Fermi Streaming Multiprocessor (SM)

Courtesy: www.nvidia.com

Warp Scheduler



- Loose Round Robin (LRR)
- Greedy Then Old (GTO)
- Two Level (TL)

Warp Scheduling

- 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)

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!

RLWS

- **RL Agent** – Warp Scheduler
- **State** – GPU + State of Warps
- **Actions** – Type of warp to schedule
- **Reward** – For scheduling a warp, penalize for stall cycle
- Update function (SARSA)
- Learning rate, Discount factor and Exploration rate

Genetic Algorithm to Select RL Configuration

- Very large design space
- To select state variables and their granularity (number of discrete values)
- To select RL and other parameters

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 our GA
  - Used 15 kernels for learning the above configurations
- Ran 59 kernels and compared the speedup (over existing warp schedulers)
- Best RL onfiguration gives
  - 5 % improvement over LRR
  - 7 % improvement over TL
  - 1 % slowdown wrt GTO
  - Best on 17 and second best on 30 kernels

Conclusion

- RL based GPU warp Scheduler
- Genetic Algorithm to search for the best set of parameter values
- Evaluated on a large set of kernels
- RLWS found to work well “across the board”