Register and Thread Structure Optimization for GPUs

Yun (Eric) Liang, Zheng Cui, Kyle Rupnow, Deming Chen

Peking University, China

Advanced Digital Science Center, Singapore

University of Illinois at Urbana Champaign, USA

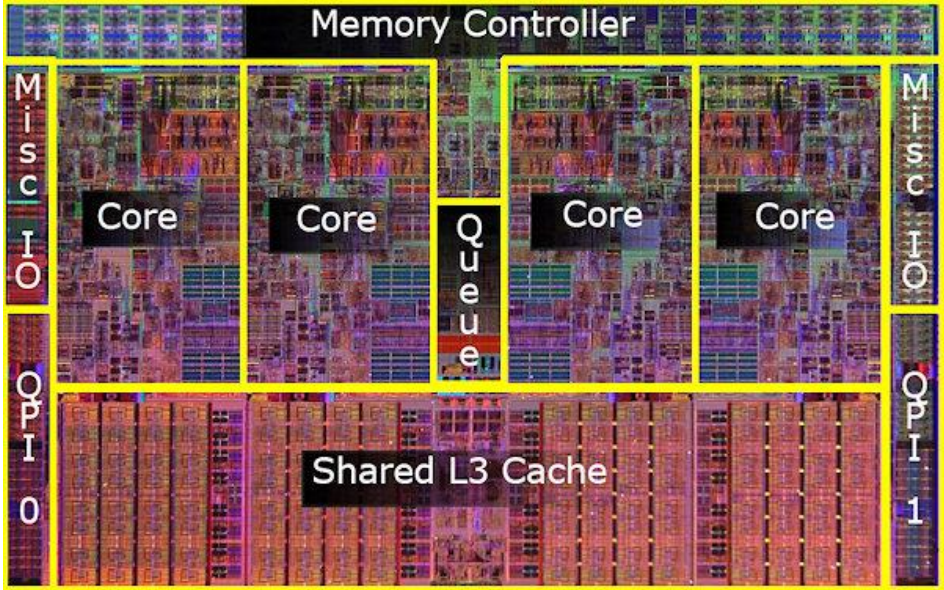
Modern Computing Systems

- GPUs
  - Computation and bandwidth
  - Three of top-five high performance machines on the June 2011 Top 500 list
  - Top two machines on the Green 500 list of the most energy-efficient supercomputers
- Heterogeneity (CPU, GPU, …) common
- Low-power embedded system
  - Tegra 2/3/4

Modern Computing Systems

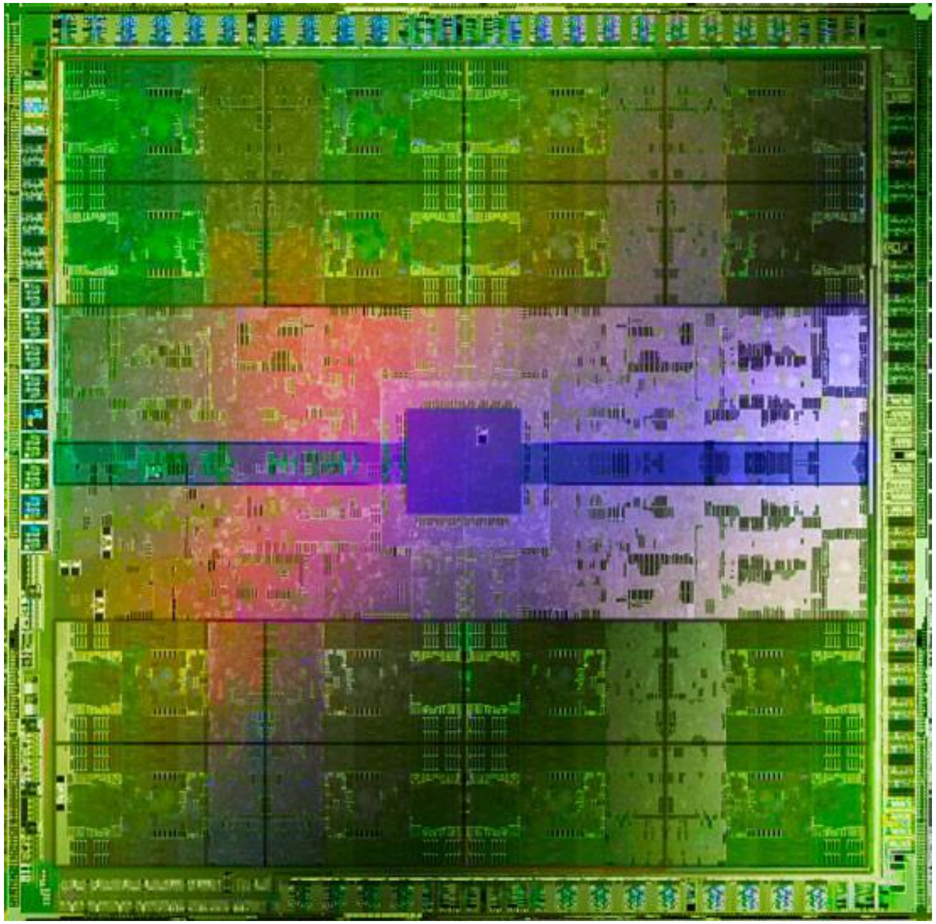
- CPUs ➔ i7 @ 263mm^2
  - Control heavy
  - Complex core
  - Less space devoted to computation resources

- Good at ILP



Modern Computing Systems

- GPUs➔ GF100 @ 529 mm^2
  - Simple cores
  - Many cores



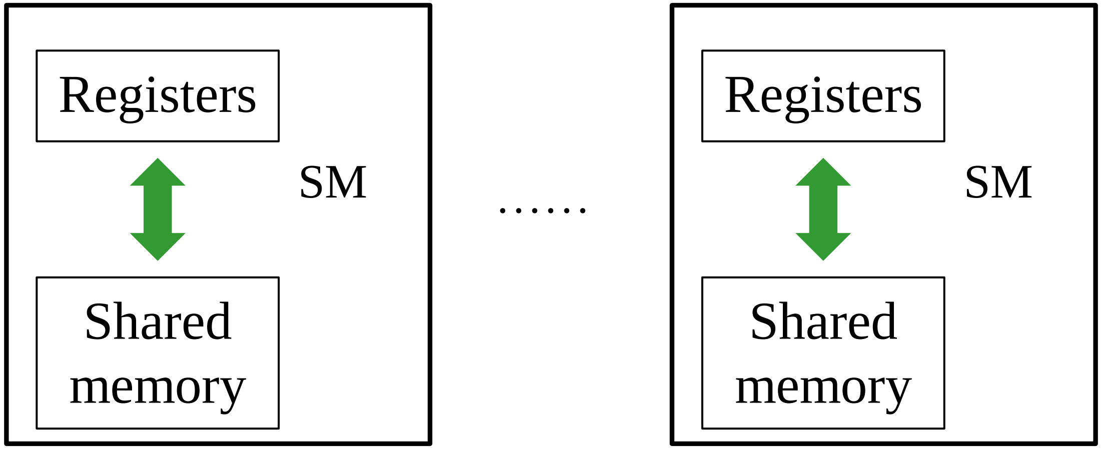
- *Data* parallelism
  - Good at TLP
  - Bad at control

GPU Performance Optimization

- **Performance tuning is difficult**
  - **Many architecture and application parameters**
  - **Kernel development is a heavy lifting task**
- **Automatic analysis and performance optimization**
- **Register and thread structure optimization**
  - **Joint optimization problem**

Register Allocation

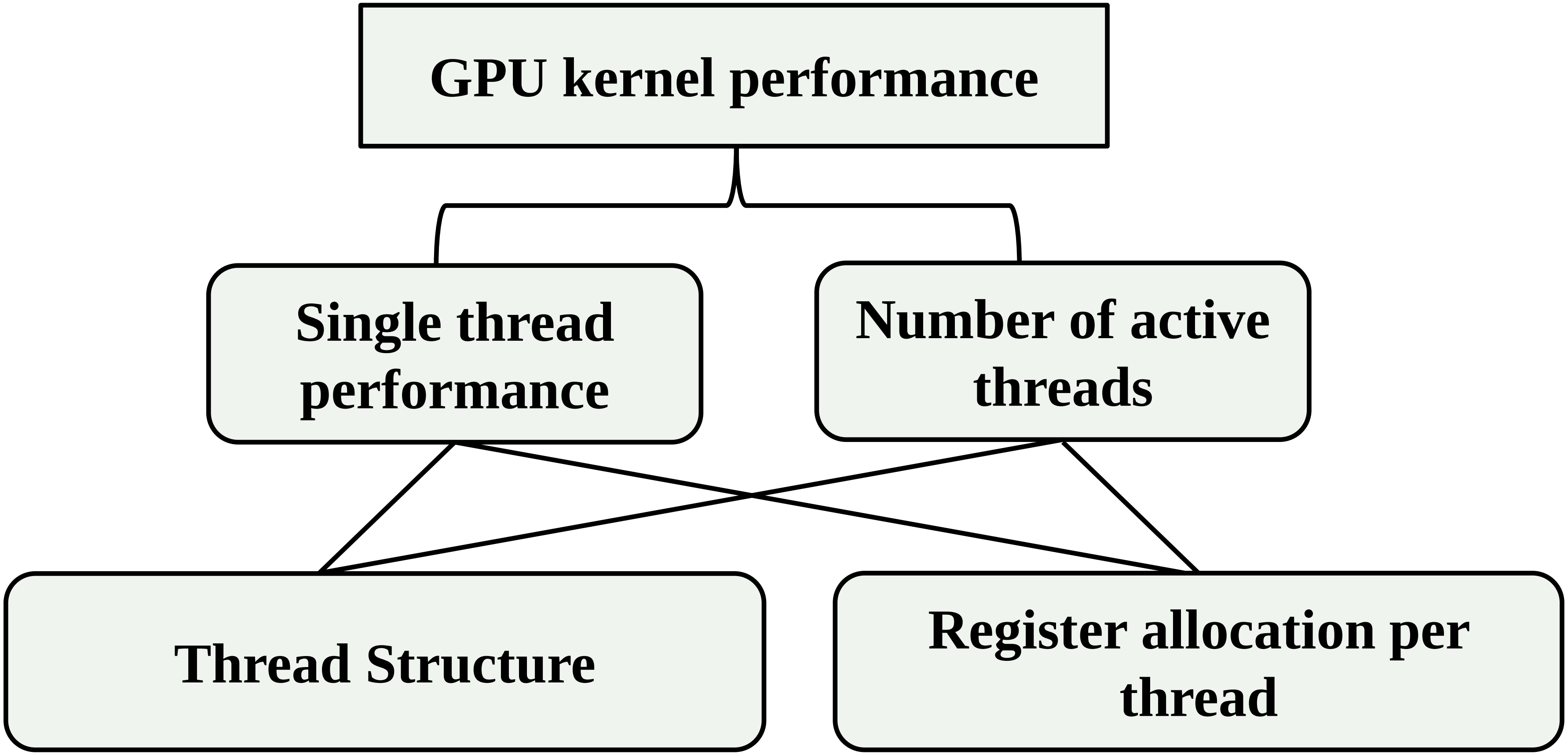
- **Large register file**
  - **GTX480: 49152 bytes; 32768 registers (32 bit) per SM**
  - **nvcc interface: -maxReg: maximum number of registers used per thread.**



Thread Structure

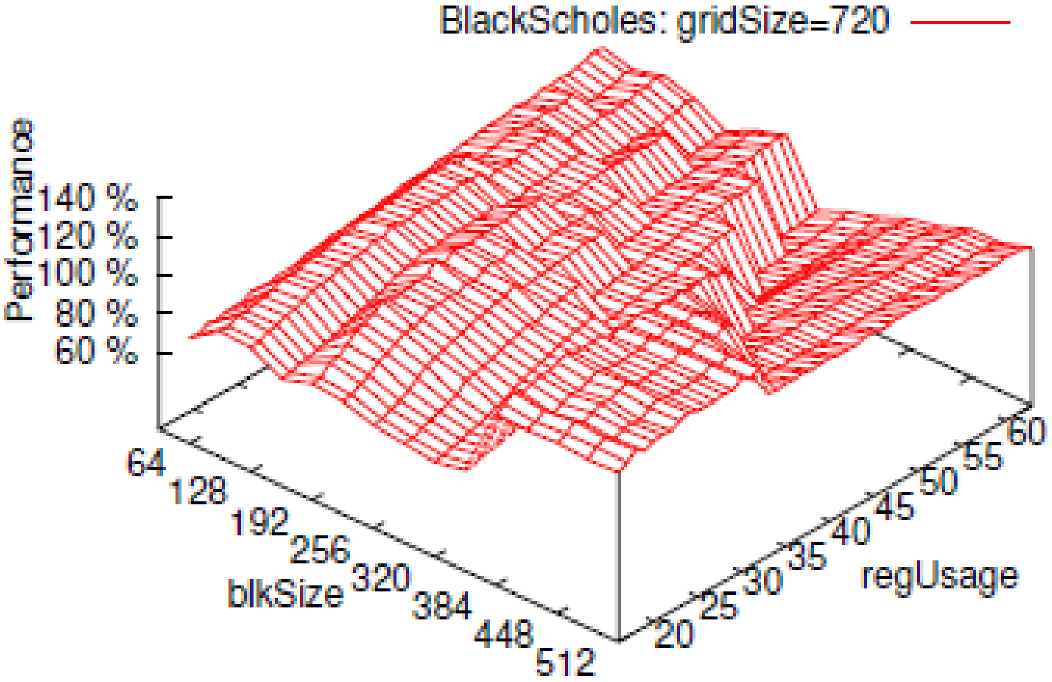
- GridSize: number of thread blocks
- BlkSize: number of threads per thread block
- Total threads: gridSize *x* blkSize
- Thread structure
  - Workload of one thread
  - Number of active threads
  - Thread scheduling

Register and Thread Structure Optimization

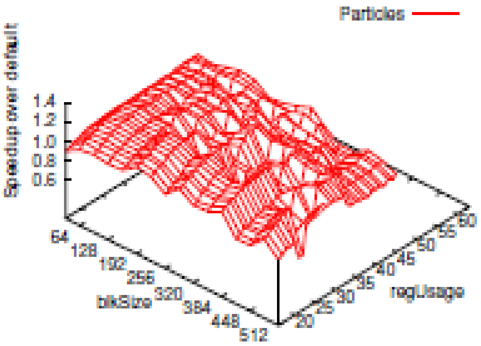


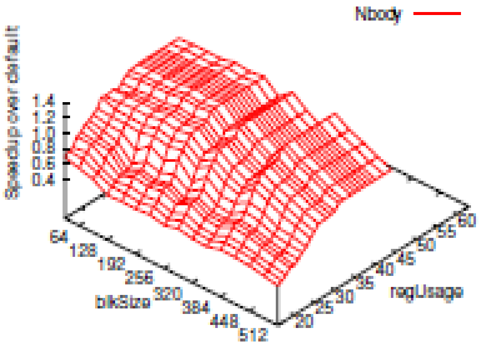
Joint Design Space

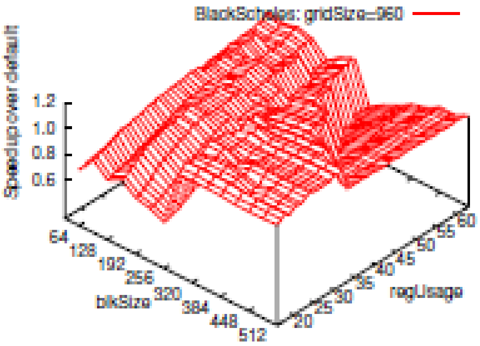
- Large design space
- Counter-intuitive performance tradeoff
- Performance improvement potential

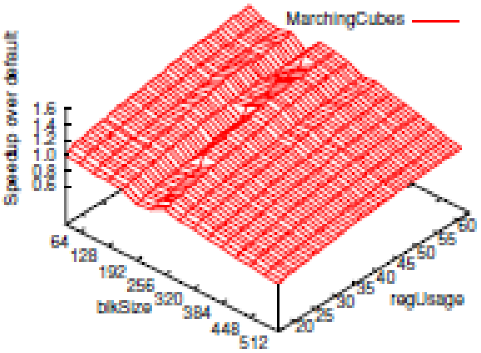


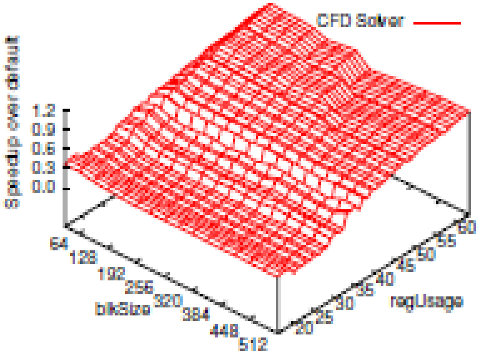
Joint Design Space – More Kernels

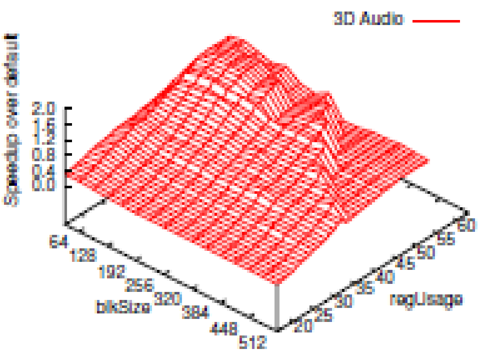




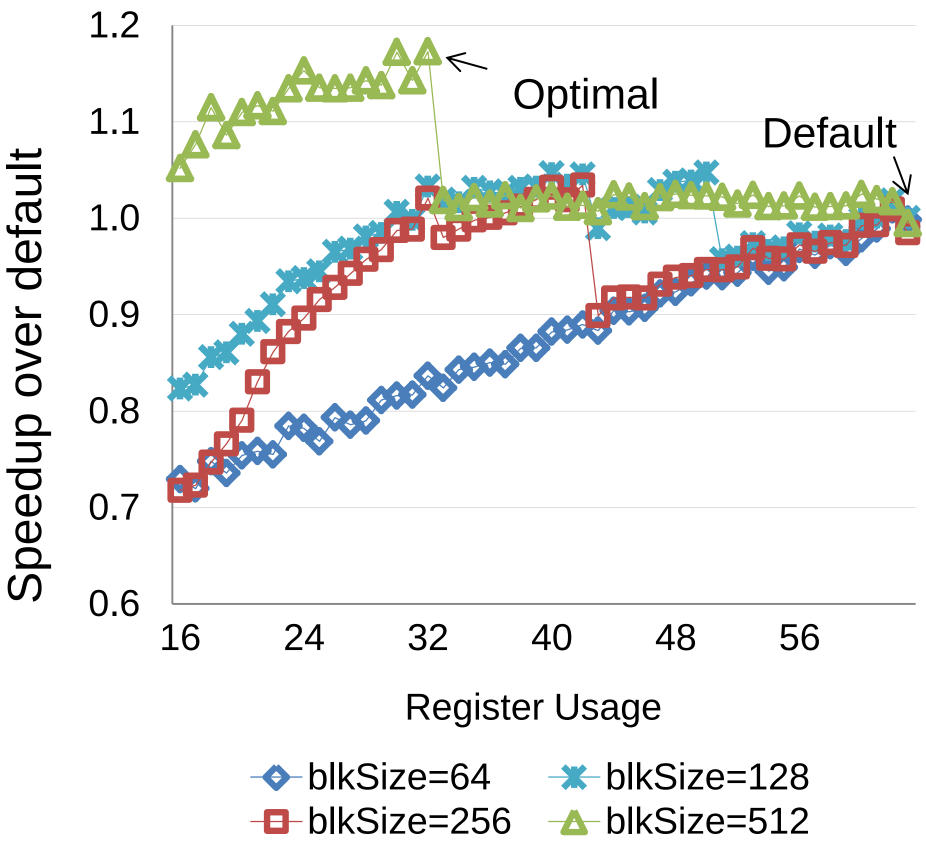




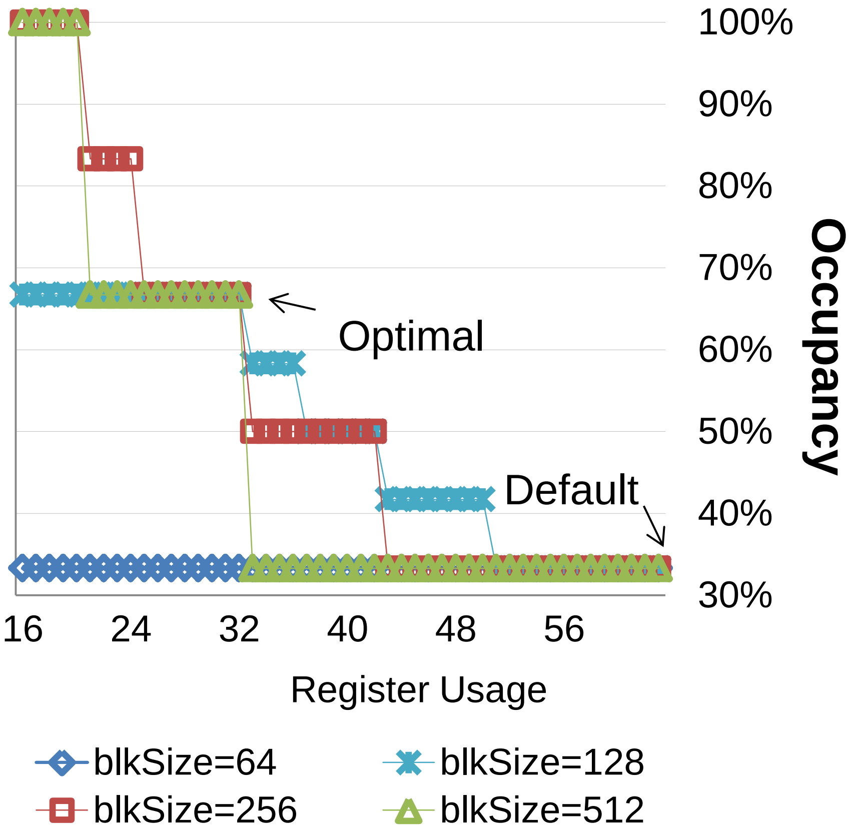




Occupancy



1.2
1.1
1.0
0.9
0.8
0.7
0.6
Speedup over default
16
24
32
40
48
56
Register Usage
Optimal
Default
blkSize=64
blkSize=128
blkSize=256
blkSize=512



100%
90%
80%
70%
60%
50%
40%
30%
Occupancy
16
24
32
40
48
56
Register Usage
Optimal
Default
blkSize=64
blkSize=128
blkSize=256
blkSize=512

Challenge… and Opportunity

- Large design space
  - Consistent increase in the shared resource and register limit
- Need to estimate performance accurately
  - Measurement not feasible
- Big speedup opportunity

Performance Estimation

- **Single thread performance**
  - **Latency of instructions**
  - **Dependencies among instructions**
  - **Control flow of the program**
  - **Basic block execution frequency**

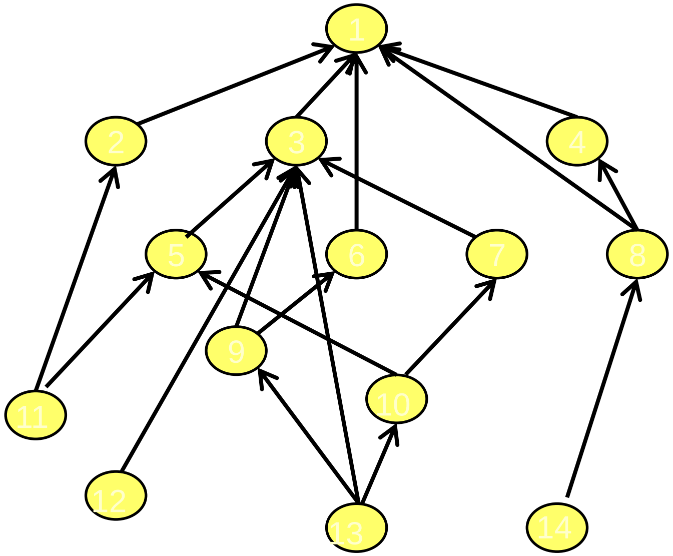
Latency of Instruction

- **Assembly code: *Cuobjdump***
- **Micro-benchmarks approach**
  - *H. Wong et al. Demystifying GPU microarchitecture through microbenchmarking. In ISPASS, 2010.*

```
1. MOV    R5,     R4;
2. F2F     R4,     R4;
3. FFMA  R52,    R55,    c[xxx],  R5;
4. FMUL  R53,    R5,      xxx;
5. MOV    R55,    xxx;
6. FSETP P0,       xxx,    R5,       xxx;
7. F2F      R56,    R52;
8. FMUL  R53,    R5,      R53;
9. FMUL  R5,      R52,    xxx;
10.FMUL R56,     R52,    R5;
11.FMUL R58,     R53,    xxx;
```

Dependencies among Instructions

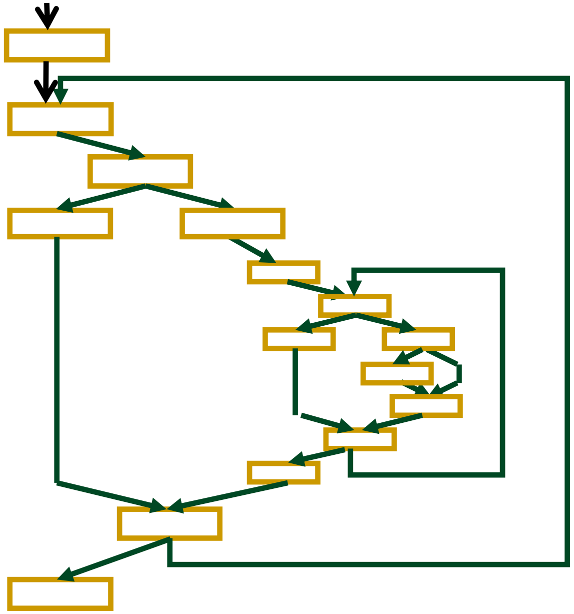
- Instruction dependency graph
  - RAW, WAR, WAW
- Basic block latency estimated as critical path



Instruction dependency graph

Control Flow Graph

- Analysis of cuobjdump code to gather CFG
- GPGPU-Sim to gather execution frequencies
    - *A. Bakhoda et al. Analyzing CUDA worloads using a detailed GPU simulator. In ISPASS, 2009.*



Single Thread Performance Estimation

$$\mathrm{Cycle}(thread) = \sum_{b \in B} cycle[b] \times freq[b]$$

- **Instruction latency**
- **Dependencies among instructions**
- **Control flow graph**
- **Basic block execution frequencies**

Kernel Performance Estimation

- **Overall performance depends on**
  - Single thread performance (Latency estimation)
  - Number of active threads (Occupancy)
- **Register and occupancy**
  - Reg ratio is a linear estimate of thread latency
  - Product of Reg ratio and occupancy
- **Performance and occupancy**
  - 2-tuple < T, C >
  - C = Cycle(thread)
  - T denotes the remaining space for active threads

Design Space Exploration

- Different DSE algorithms with tradeoffs
  - GPU kernel performance
  - DSE runtime

- Design space exploration approaches
  - Exhaustive Search (ES) – Infeasible, but optimal
  - RO Search (ROS)
  - Performance and Occupancy Search (POS)
  - POS with filtering (POSF)

RO Search (ROS)

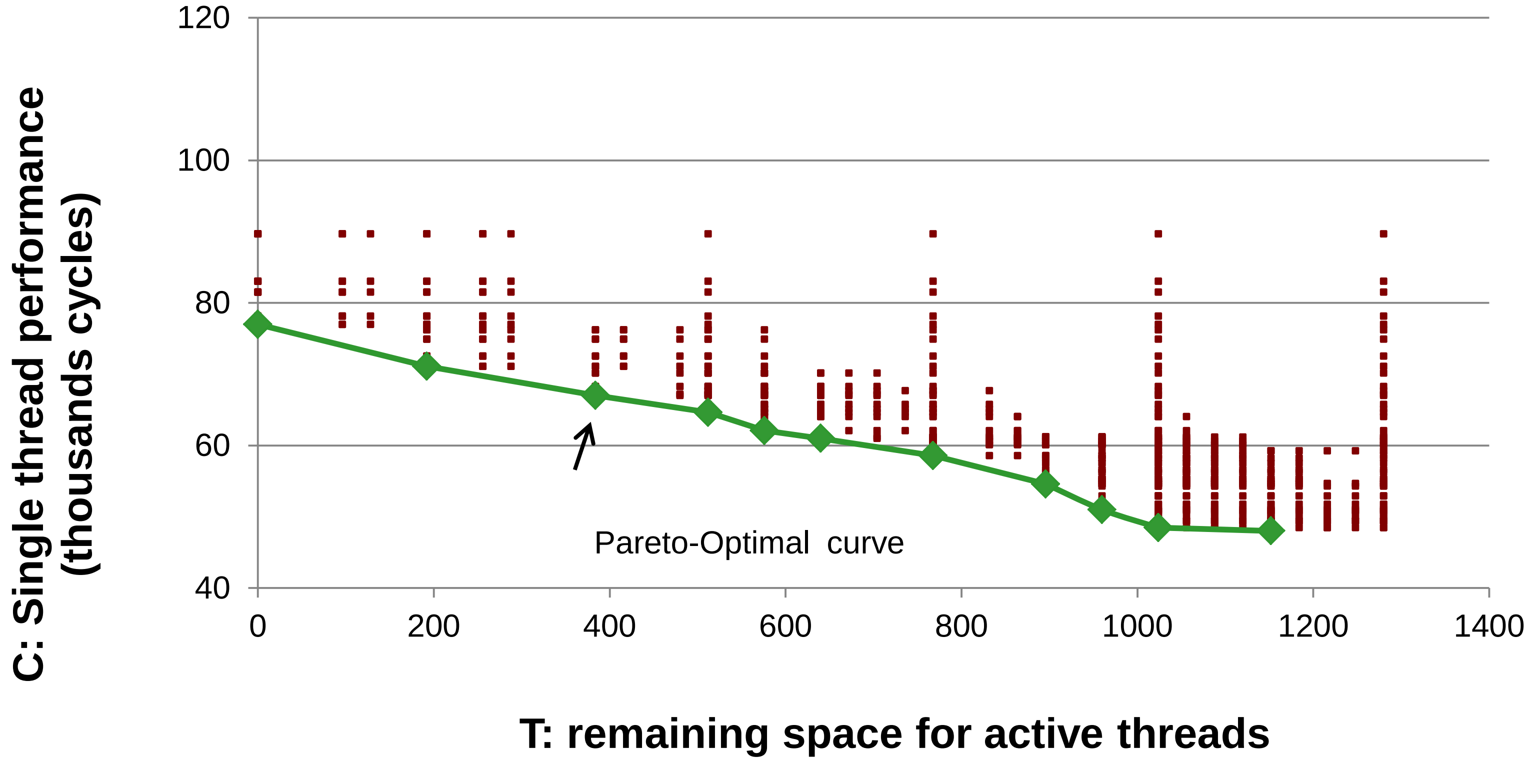
- Use *Reg ratio* **x** *Occupancy* as performance metric
- *0 < Reg ratio <= 1; 0 < Occupancy <= 1*
- Find configurations with maximal RO value
- Break ties through empirical measurement

Performance and Occupancy Search (POS)

- **Design space parameters**
  - **gridSize, blkSize, reg, PO metric (T,C)**
- **Pareto-optimal problem**
  - **Two candidates, A & B: if A is better in both T & C, it dominates B and B can be eliminated**
- **Detailed Algorithm**
  - **Step 1: build the pareto-optimal set of candidates using performance estimation**
  - **Step 2: compare candidates empirically to verify selection**

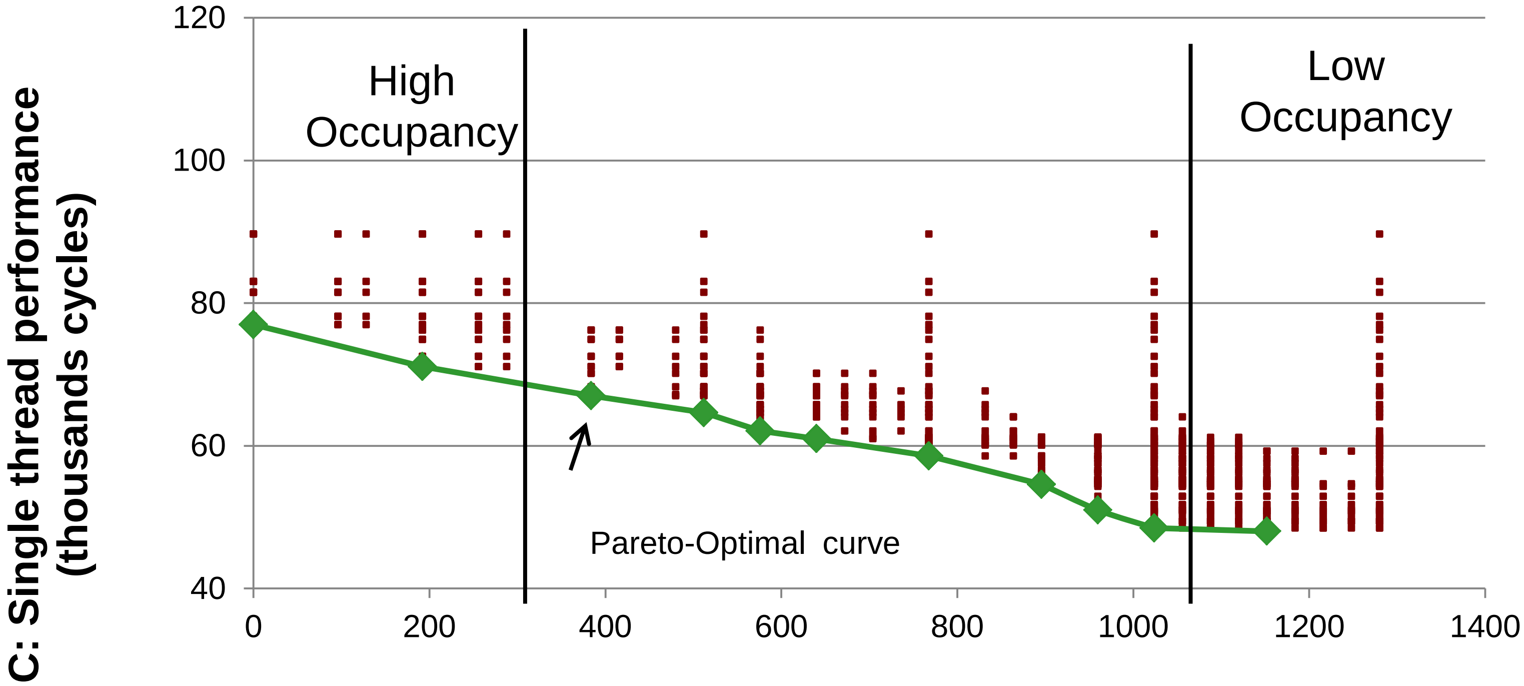
PO Search – Example Pareto-Optimal Set

- Find pareto-optimal points and compare



POS with Filtering (POSF)

- **Prune candidates less likely to be the optimal**



Solution Summary

- **nvcc provides interface for register control**
  - **Maxreg: maximal number of registers allocated per thread**
- **Thread structure**
  - **blkSize and gridSize are kernel call arguments**
- **Algorithms**
  - **ES, ROS, POS, POSF**
- **Suitable for compiler integration and portable to any GPU architecture**

Experiments

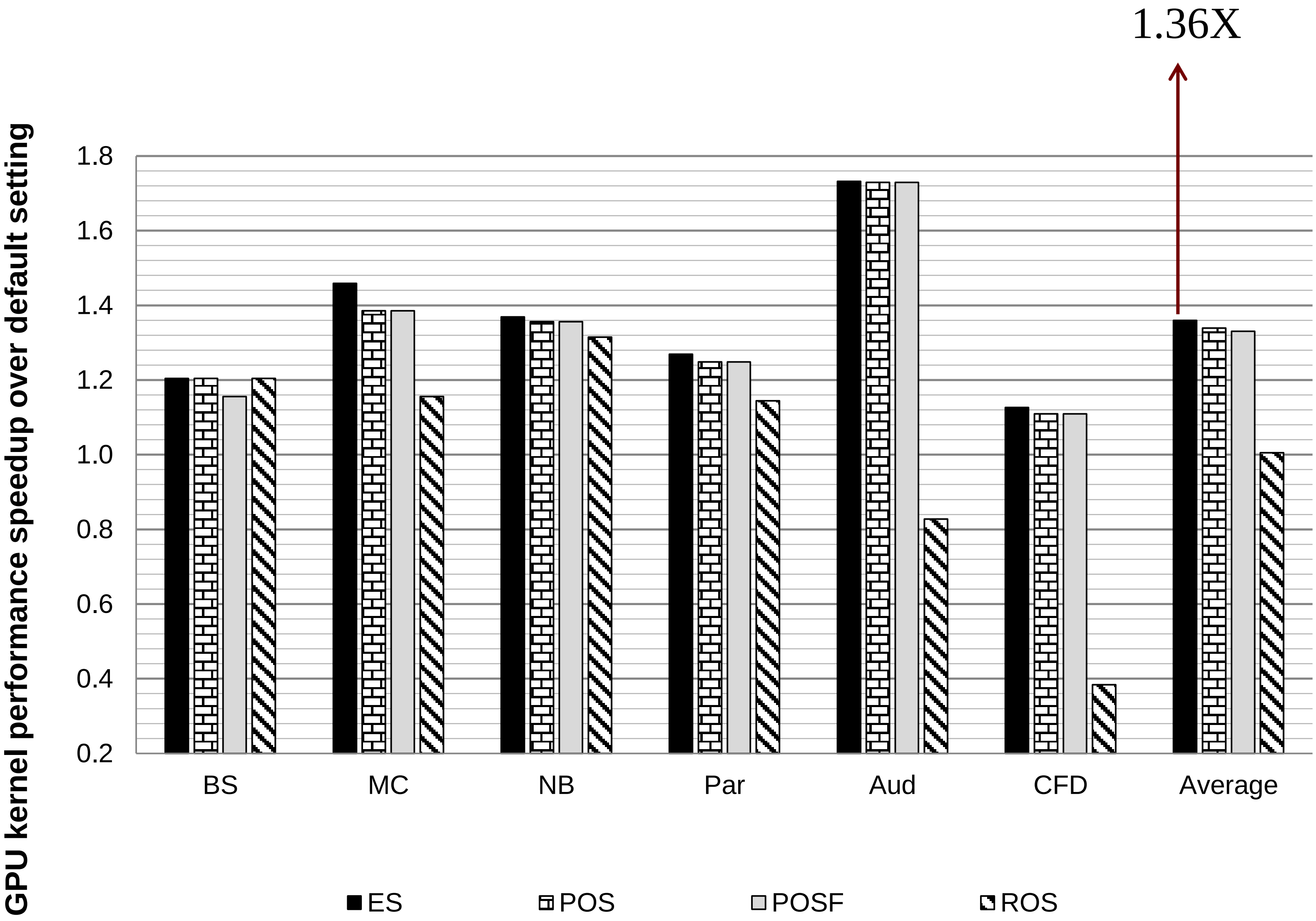
- GTX480

Benchmarks		
Blackscholes (BS)	CUDA SDK	blkSize and gridSize
MarchingCubes (MC)	CUDA SDK	blkSize/gridSize
Nbody (NB)	CUDA SDK	blkSize/gridSize
Particles (Par)	CUDA SDK	blkSize/gridSize
3D Audio (Aud)	Real-applications	blkSize
CFD Solver (CFD)	Rodinia	blkSize/gridSize

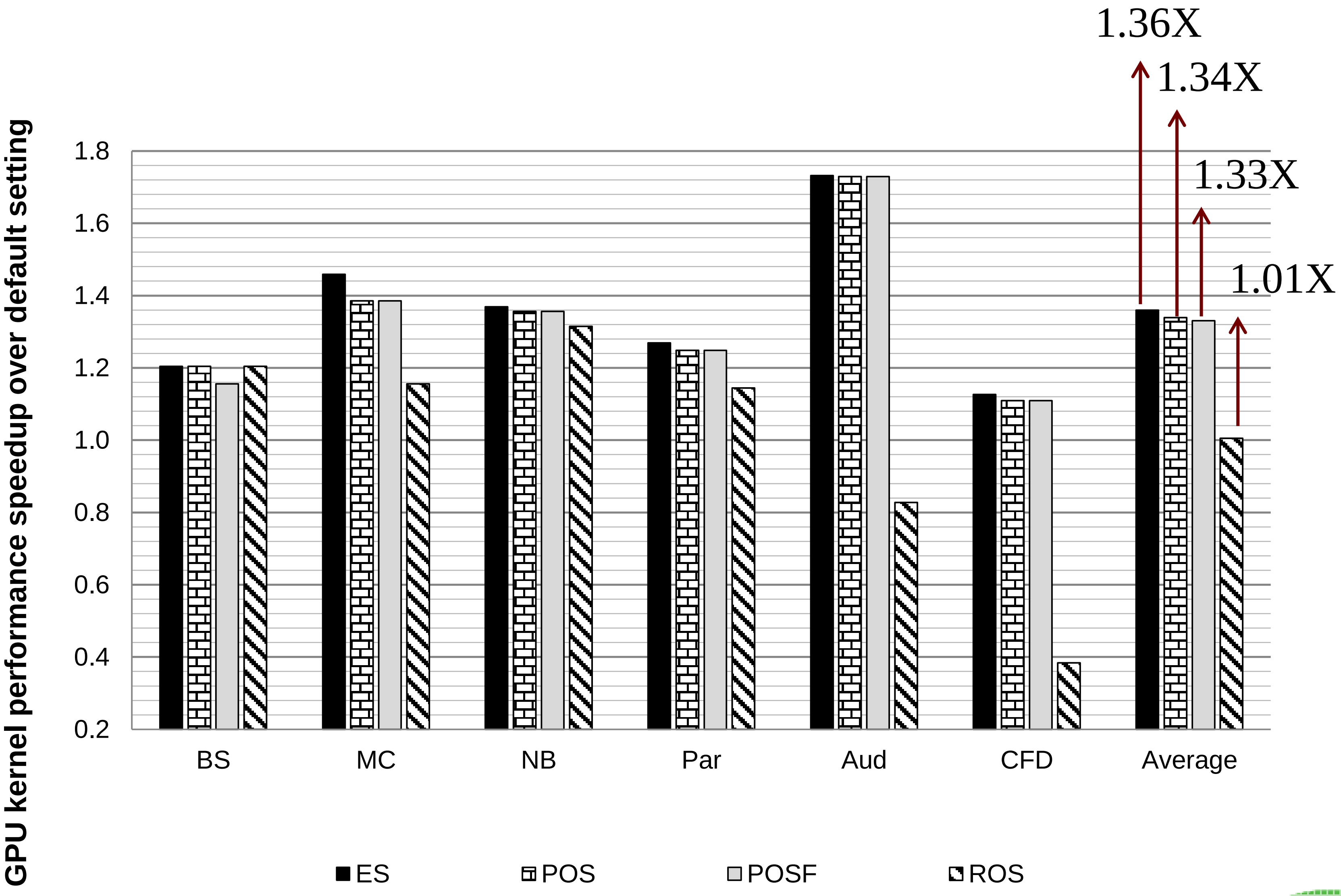
Design Space

- Register per thread
  - 16 – 63
- Threads per block
  - multiple of 32 as warp size is 32
  - 32 to 512
- POSF filter range
  - 0.3 – 0.5

Speedup on GTX480



Speedup on GTX480



Design Space Exploration Runtime

Benchs	Runtime (sec)				Speedup
	ES	RO	POS	POSF	POSF
BS	14472	55	693	244	59X
MC	25746	95	465	169	152X
NB	76490	225	667	64	1199X
Par	40560	183	416	76	531X
Aud	17454	70	1649	274	64X
CFD	4364	21	270	34	128X
	Average				355X

Conclusion

- GPU optimization of register & thread structure
  - Acceleration opportunity, but design space very large
  - Accurate performance estimation
  - Efficient design space exploration

- POS, POSF algorithm
  - High improvement with small runtime overhead
  - Kernel latency speedup 1.33X
  - Design space exploration speedup 355X

Thank you !!!