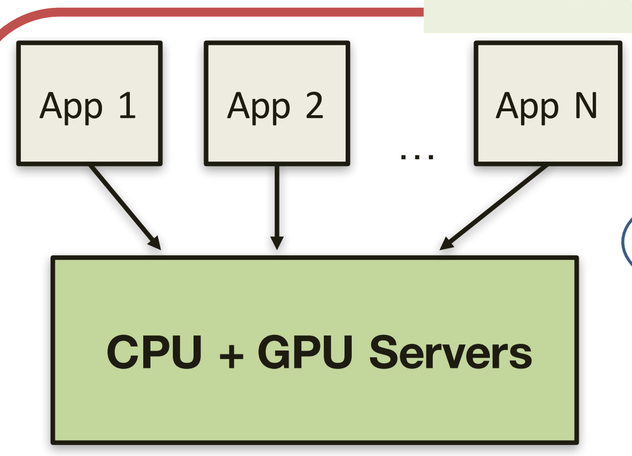
USC Viterbi School of Engineering
Ming Hsieh Department of Electrical Engineering

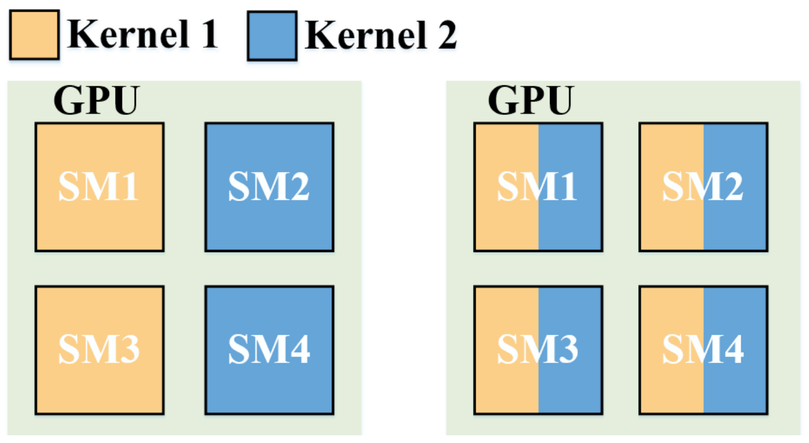
Efficient Intra-SM Slicing for GPU Multiprogramming

Qiumin Xu, EE / Dr. Annavaram

Intra-SM Slicing vs. Inter-SM Slicing

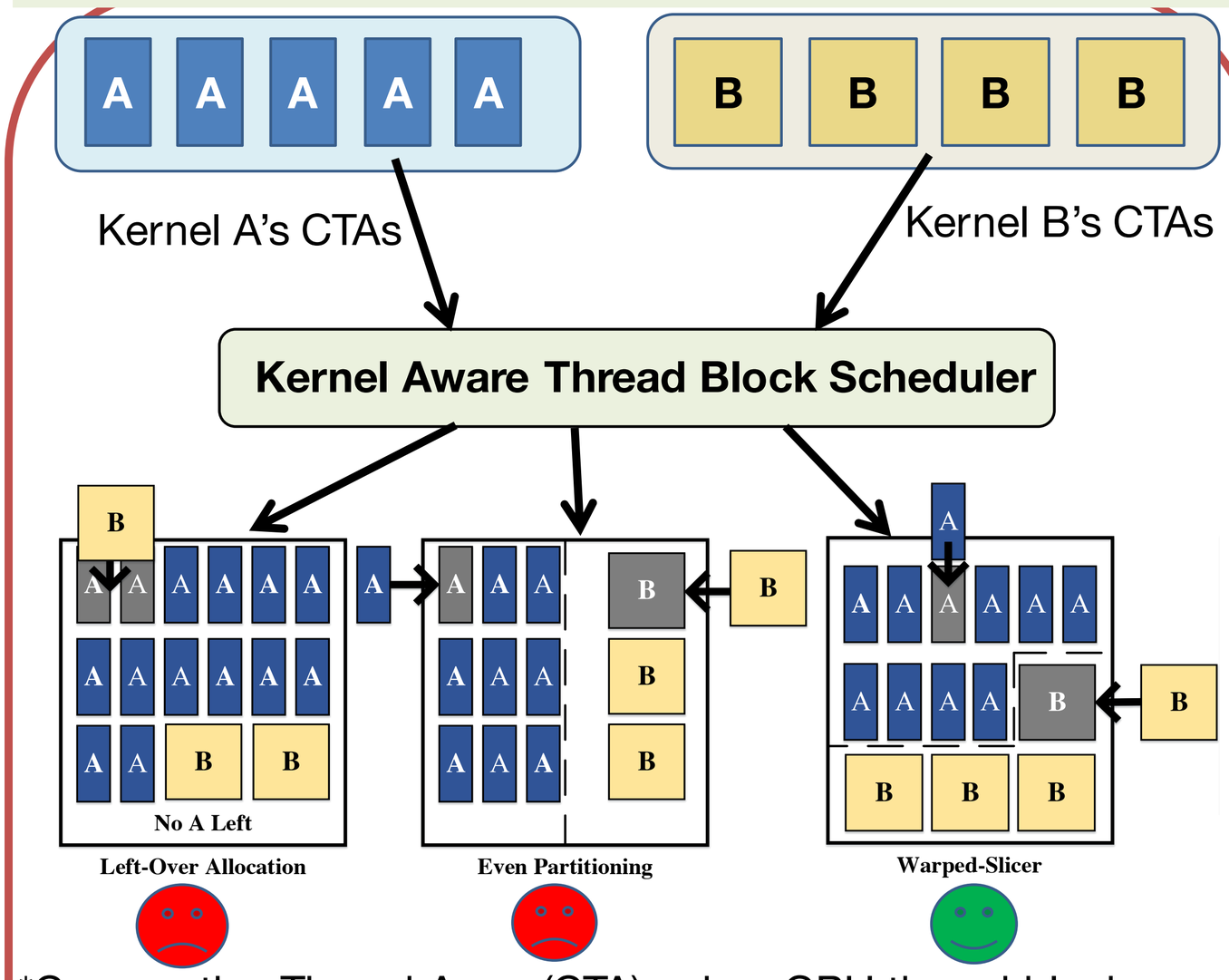


How to assign kernels to different GPU SMs?



*Streaming Multiprocessors (SM), a.k.a. GPU shader cores

CTA Scheduling for Intra-SM Slicing



*Cooperative Thread Array (CTA), a.k.a. GPU thread blocks

Warped-Slicer

$$\text{Max} \ \min_i P(i, T_i) : \sum_{i=1}^{K} R_{T_i} \le R_{tot} \quad (1)$$

P(i, T_i) : normalized IPC of application **i**

K : # of applications sharing the SM.

R_{Ti} : the resource requirement of **T_i** CTAs

R_{tot} : the total resources available in an SM

N: maximum # of CTAs per SM

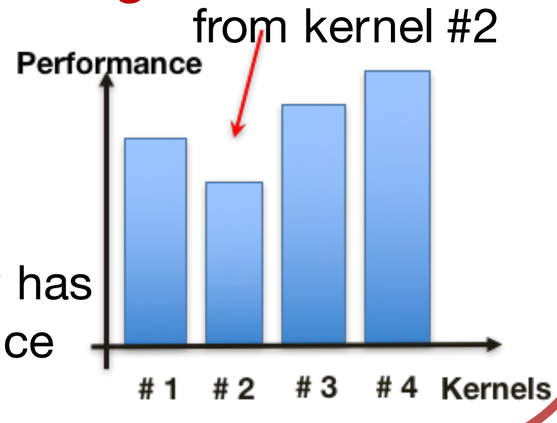
Naïve Brute force algorithm: $O(N^K)$

Proposed waterfilling algorithm:

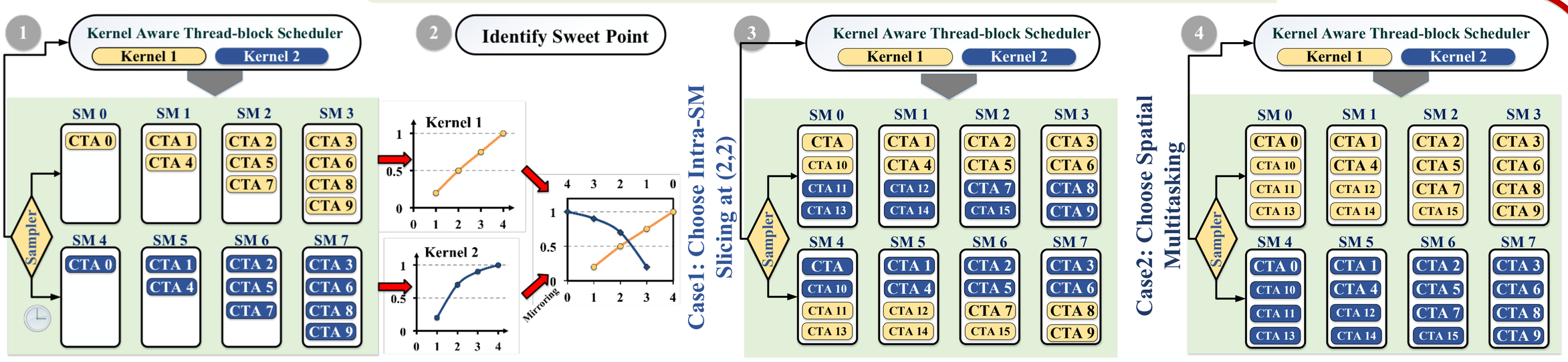
$O(NK)$

i. identify kernel #2 has the MIN performance

ii. assign 1 CTA from kernel #2



Dynamic Resource Partitioning



1. Assign a sequentially increasing # of CTAs from each kernel
2. Employ a sampling phase to measure the IPC of each SM
3. Calculate the best partitioning based on Equ. (1)
4. Continue to assign CTAs to SMs based on the decision

Experimental Results

Application Pairs	Speedup	Increased Util.	Fairness Improved	Hardware Overhead
30	1.23x	15%	14%	~0

qiumin@usc.edu, annavara@usc.edu

DARPA NSF