QoS-Aware Dynamic Resource Allocation for Spatial-Multitasking GPUs

Paula Aguilera
Katherine Morrow
Nam Sung Kim

University of Wisconsin-Madison

Outline

- QoS Applications on Multitasking GPUs
- Methodology, Platform & Applications
- GPU Resource Allocation
  - Static Resource Allocation
  - Dynamic Resource Allocation
- Conclusions

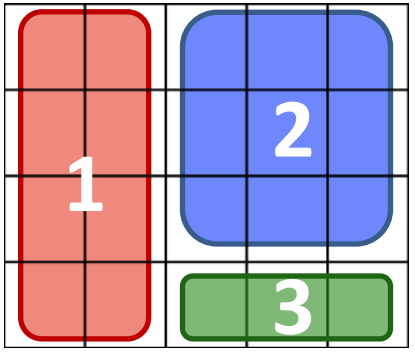
GPGPU Computuation for QoS

- Many applications have QoS performance requirements that they need to satisfy
  - Frames-per-second, transmission rate, etc.
- These types of applications are also often highly parallel
  - GPUs' large number of cores can make them an effective compute platform for meeting QoS
- However, these applications may not run in isolation—multitasking may be required!
  - How can we best multitask a GPU to meet QoS requirements?

GPU Multitasking

- Each generation of GPUs has more parallel computation capability than the previous one
  - But… many GPGPU applications fail to fully utilize the GPU resources

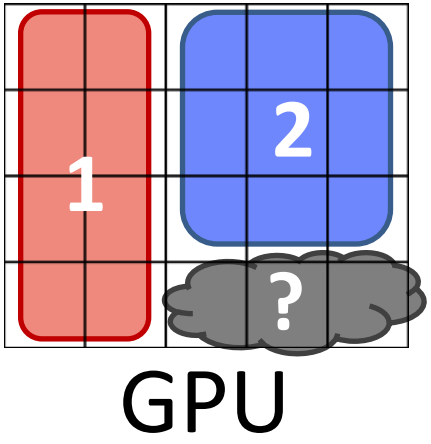
- **SPATIAL MULTITASKING:** Divide GPU resources **spatially** rather than **temporally**
  - Allocate a subset of the GPU's Streaming Multiprocessors (SMs) to each co-executing application
- How do we choose this allocation?



GPU

Resource Allocation for QoS

- Spatially-multitask QoS applications with other applications on the GPU
  - Need to ensure each QoS application is allocated enough resources to meet their requirements
- Goal: Allocate minimum SMs to QoS applications to meet their requirements
- What can we do with GPU resources "unneeded" for QoS?
  - Accelerate co-executing best-effort applications to improve performance
  - Leave them idle for power savings



Methodology

- Measure, for different execution scenarios, how many SMs are not needed to meet QoS
  - Determine effects of using these resources to accelerate other applications vs. leaving them idle
  - Examine how these effects scale with increasing (or decreasing) QoS requirements

- Determine how to allocate the minimum number of SMs to QoS applications to meet requirements
  - Does the required allocation depend on co-executing applications (due to memory/interconnect contention)?
  - Do we need to allocate at runtime based on workload?

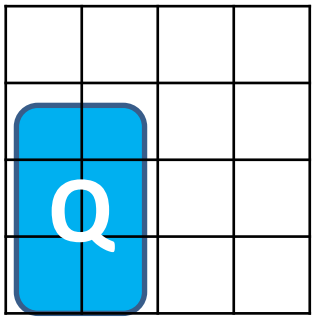
Platform & Applications

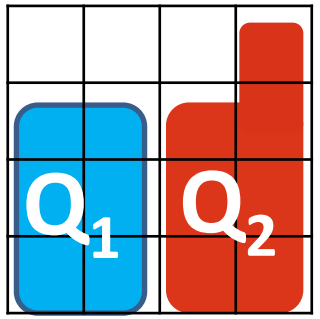
- Modified GPGPU-Sim:
  - Support for spatial multitasking
  - NVIDIA Quadro FX 5800 (GT200 architecture)
  - 30 SMs and 8 memory controllers

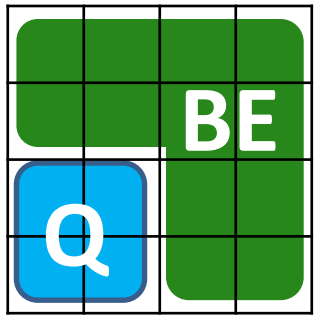
- Benchmarks:
  - QoS: AES Decoding (AES-D), JPEG Decoding (JPEG-D) and SHA1
  - Best-effort: Image Denoising (ID), Ray Tracing (RAY), Dirac Video Codec (DVC), Sum of Absolute Differences (SAD), and Fractals

Execution Scenarios

- Examine three scenarios:
  - One QoS application
    - Leave SMs idle to save power
  - Two QoS applications
    - Leave SMs idle to save power
  - One QoS application + one best-effort application
    - Use extra SMs for acceleration



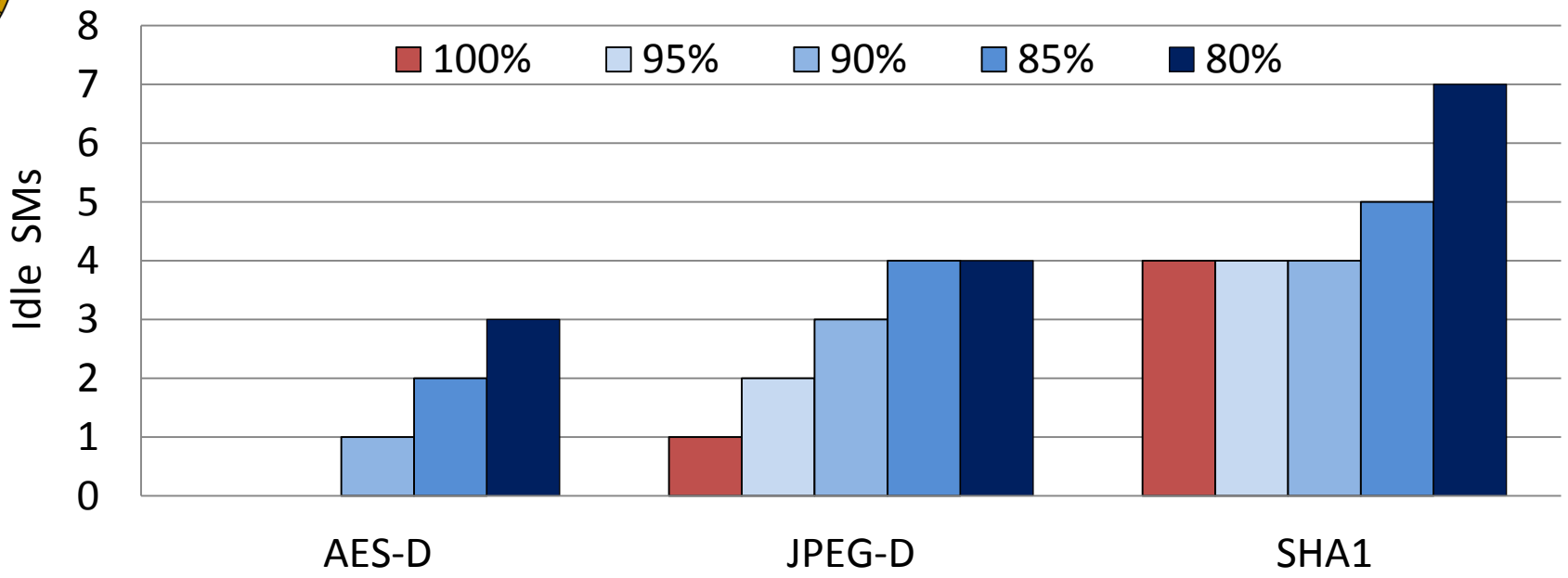




Calculating QoS

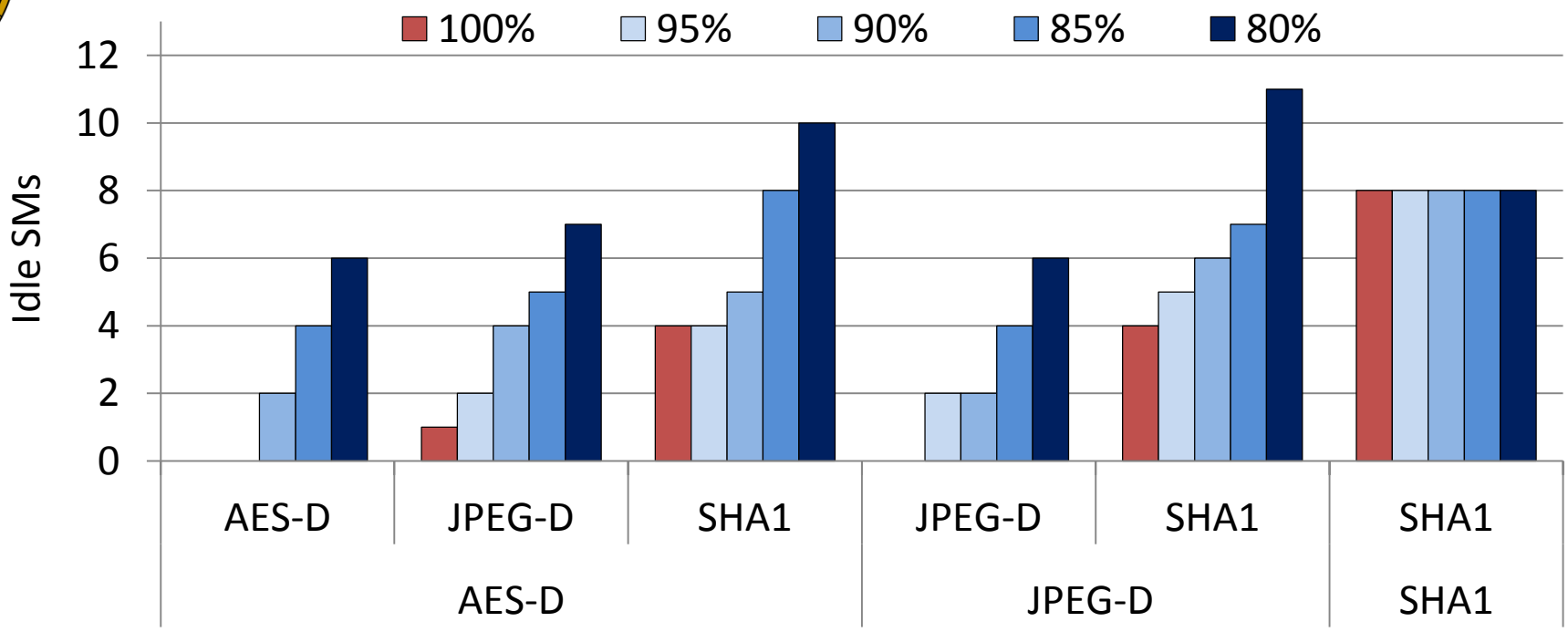
- Assume applications meet their QoS when cooperatively multitasked with one other app
  - Required QoS = work performed using 100% of resources for 50% of simulated time
- Attempt to achieve same work using 50% or fewer SMs for 100% of simulated time
- Also test relaxed QoS levels
  - 100%, 95%, 90%, 85% and 80% of the QoS level calculated above

One QoS Application



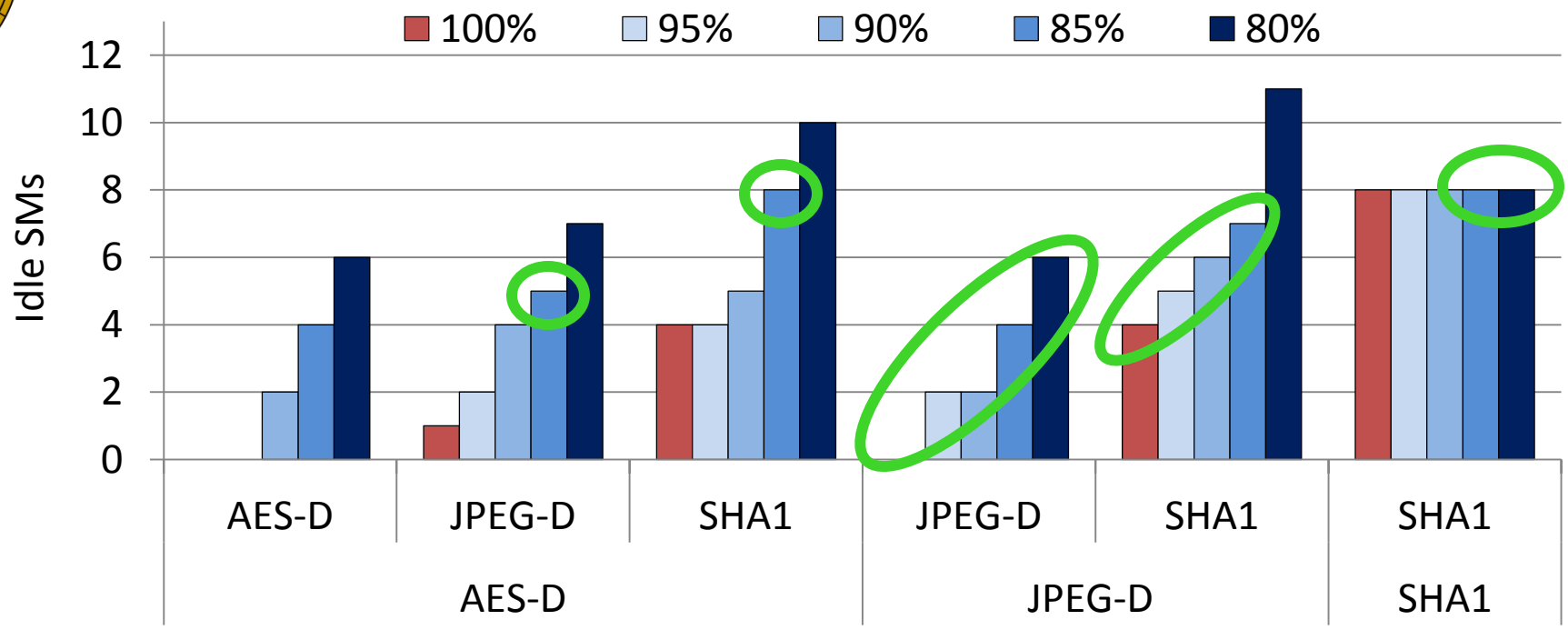
- QoS application in isolation using up to 50% of the SMs for 100% of the time
- Calculate # SMs (out of 15) that are not needed
- Number of SMs that can be left idle increases as QoS requirement is relaxed

Two QoS Applications



- Evaluate max # of SMs that can be disabled and still have applications meet their QoS

Two QoS Applications



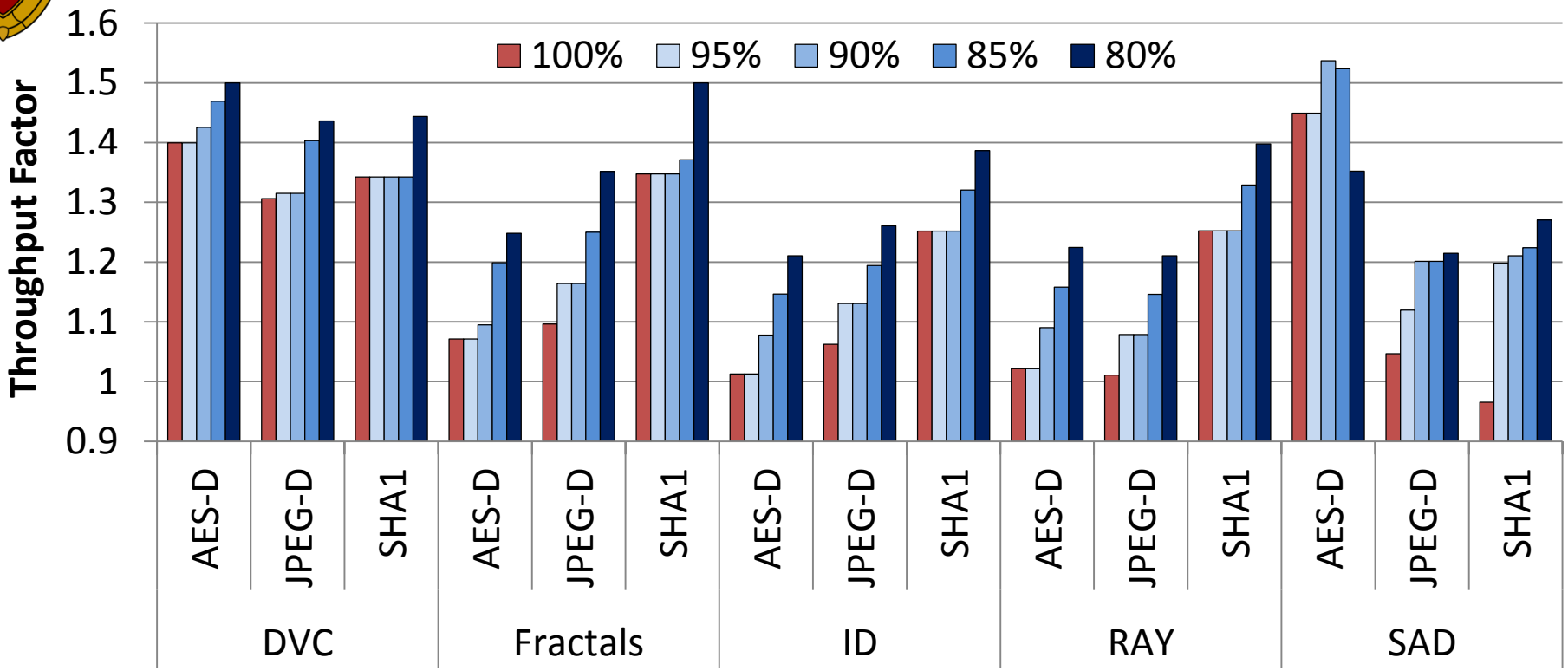
- Evaluate max # of SMs that can be disabled and still have applications meet their QoS
- Number of idle SMs when sharing GPU not always equal to sum of idle SMs in isolation!

Idle SM Power Savings

- Use GPUWattch to determine the average power savings for two QoS applications when:
  - Leaving the unneeded SMs idle (reducing dynamic power vs. if those SMs were active), or…
  - Power-gating the unneeded SMs

QoS Level	Idle SMs (W)	Power Gated SMs (W)
100%	2.6	6.9
95%	5.8	11.2
90%	8.8	15.8
85%	13	22.2
80%	18.1	28.6

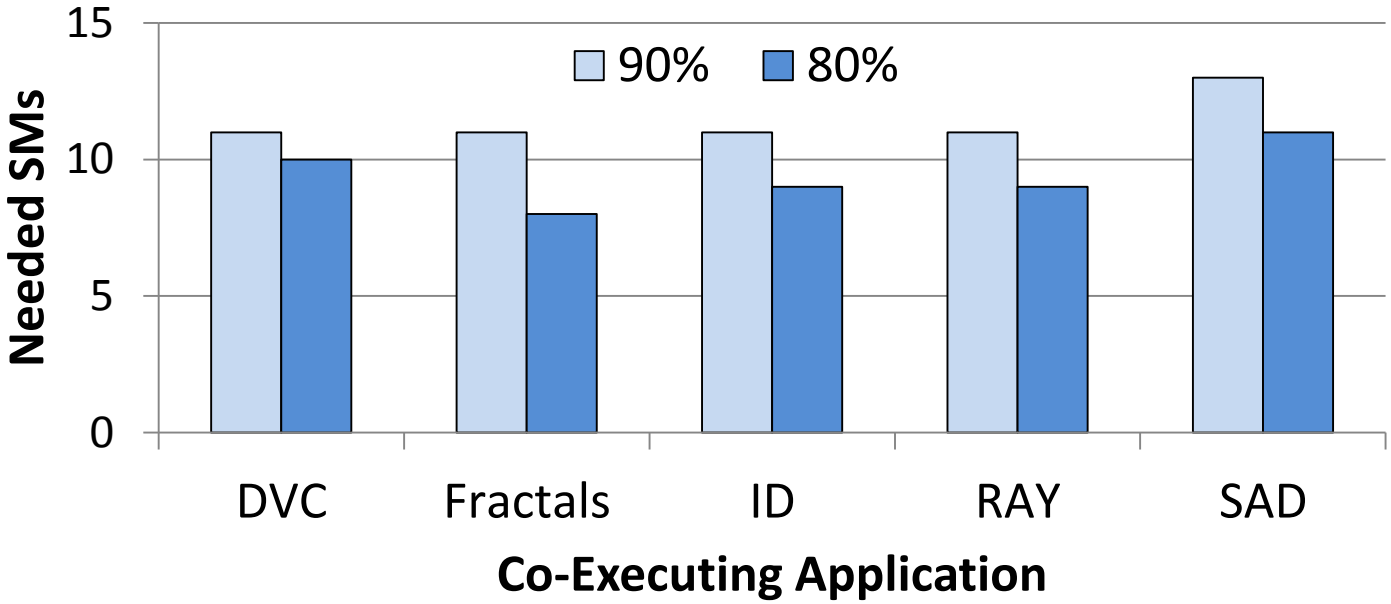
One QoS + One Best-Effort



- Performance increase of allocating "extra" SMs to best-effort application (relative to cooperative multitasking)
  - Average 17.5% performance increase for 100% QoS

Contention When Sharing GPU

of SMs Needed by SHA1 to Meet 90% and 80% QoS



- Performance of an application using spatial multitasking on the GPU depends on any co-executing applications
- Contention in shared resources (memory, interconnect)

Maximum Performance Loss

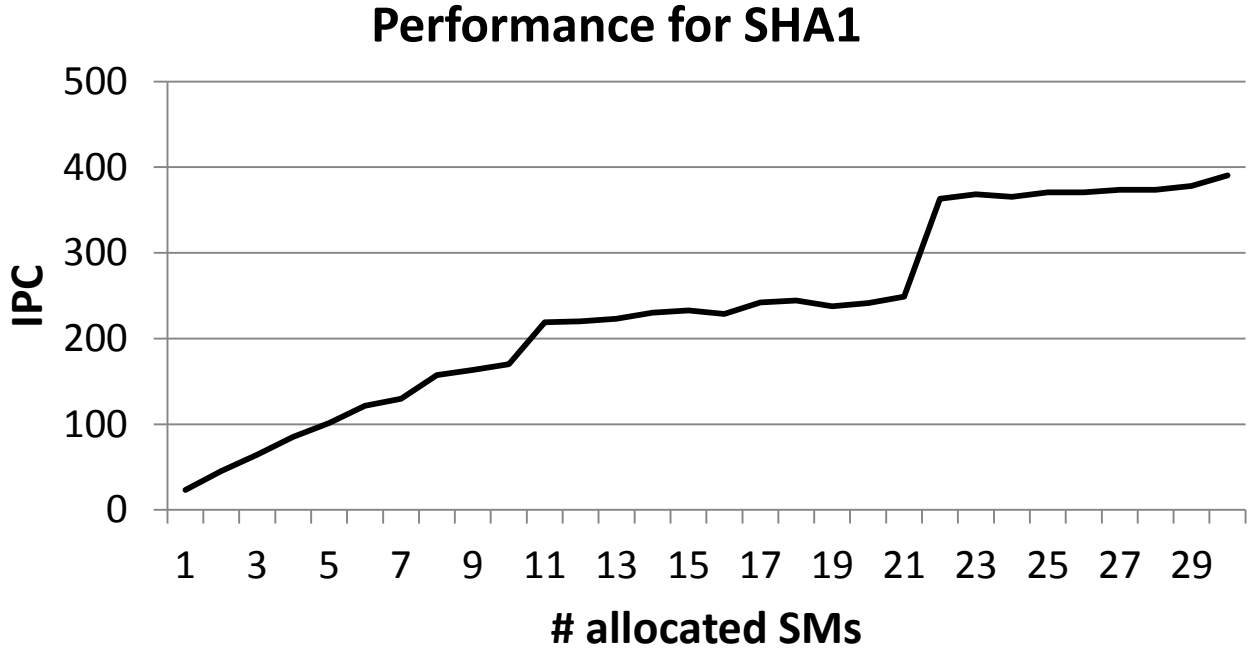
- Compare QoS application performance with “most interfering” best effort application vs. in isolation

APPLICATION	MAX. PERFORMANCE LOSS (15 SMs)	MAX. PERFORMANCE LOSS (10 SMs)
AES-D	0.3%	0.3%
JPEG-D	16%	23.2%
SHA1	18.2%	24%

- For best results, need to allocate at runtime based on the executing workload!

Dynamic Allocation

Performance depends on the characteristics of both QoS and co-executing applications



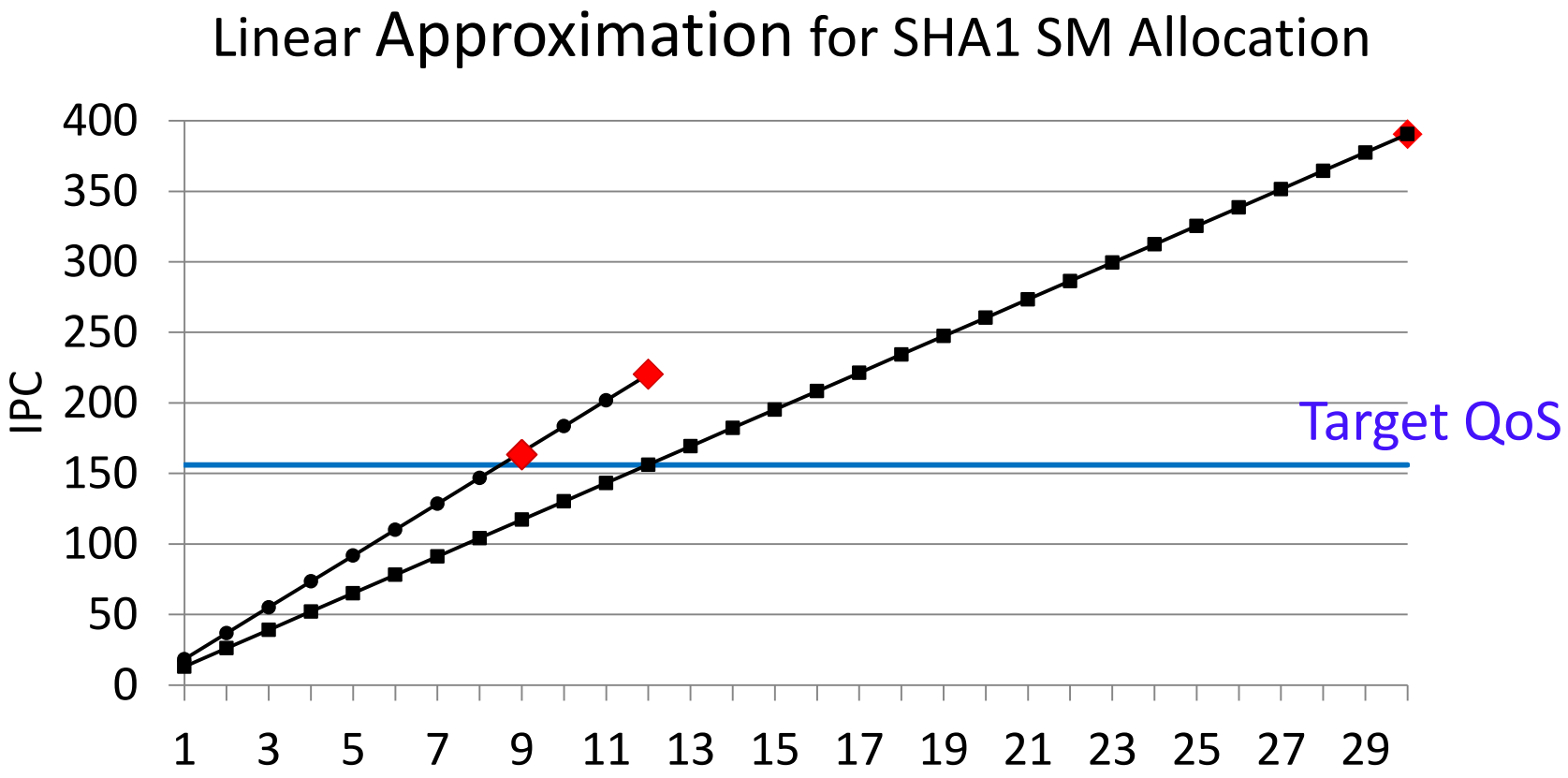
- Need runtime algorithm
  - Take advantage of fact that most applications show sub-linear speedups with # SMs

Dynamic Allocation

- Iterative method:
  1. Allocate more than enough SMs to meet QoS
  2. Measure performance
     - If performance > QoS: estimate if fewer SMs can be used. If yes, choose the middle point of the estimation
     - If performance < QoS: estimate how many more SMs are needed
  3. Change to the new # of SMs, and goto #2.

- When the extra SMs are assigned to other applications we need to first wait for them to finish execution

Dynamic Allocation Example



[Example uses end- instead of mid-point as simplification]

Conclusion

- Spatial multitasking satisfies QoS and:
  - Improves system performance (17.5%) by allocating unused SMs to best-effort apps, or…
  - Saves power (7W) by leaving them idle
- # SMs required to meet QoS depends in part on other applications using the GPU
  - Dynamic allocation is necessary
- Presented a dynamic allocation algorithm based on linear approximation to maximize benefit of spatial multitasking for QoS