Karlsruhe Institute of Technology

GPUswap: Enabling Oversubscription of GPU Memory through Transparent Swapping

Jens Kehne, Jonathan Metter, Martin Merkel, Frank Bellosa
GI Fachgruppe Betriebssysteme – Frühjahrstreffen 2017

Operating Systems Group, Karlsruhe Institute of Technology (KIT)

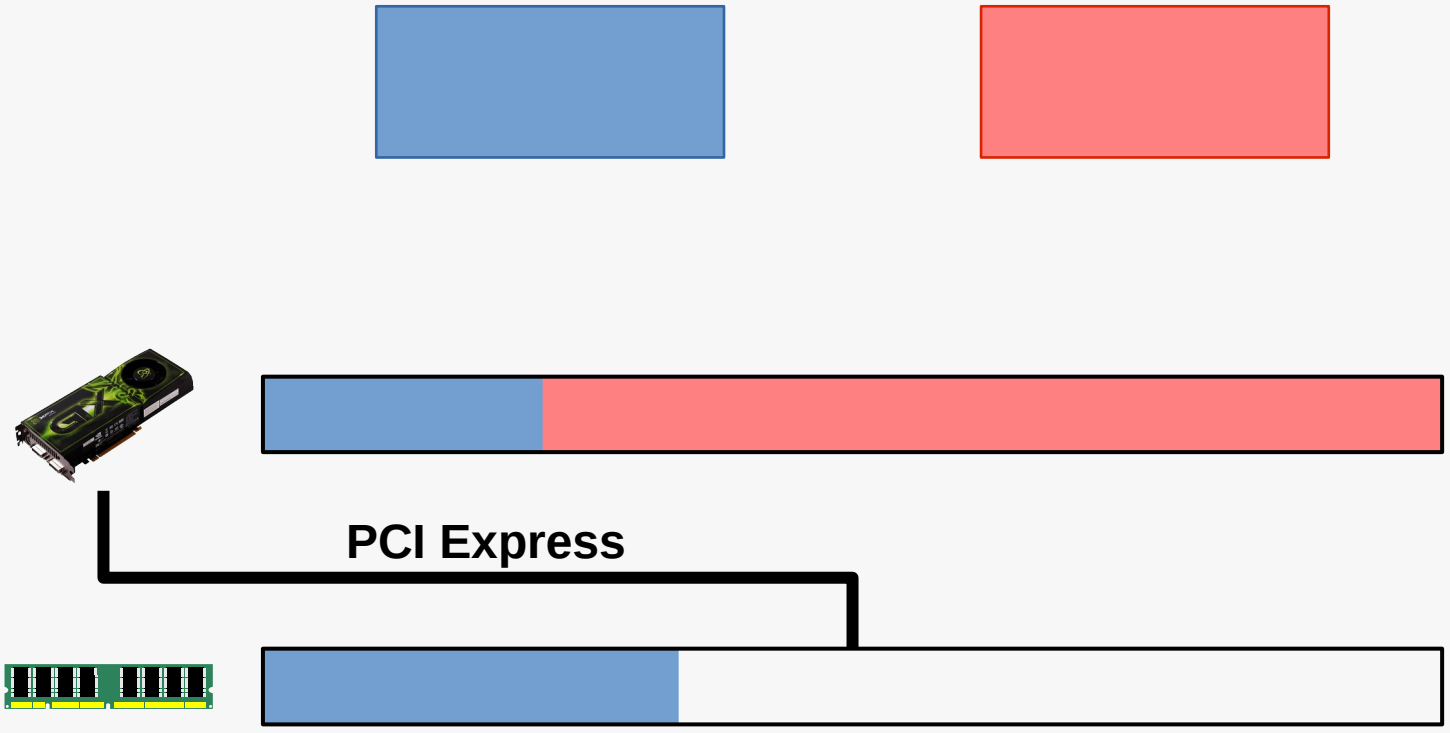
Motivation

- GPUs are widespread in computing
  - Unprecedented performance for some applications
  - Very energy efficient
- GPUs are moving to the cloud
  - Cost effective through oversubscription
- Can safely share computational power
  - Even have fairness to some degree
- But what about memory?

Sharing GPU Memory – FCFS

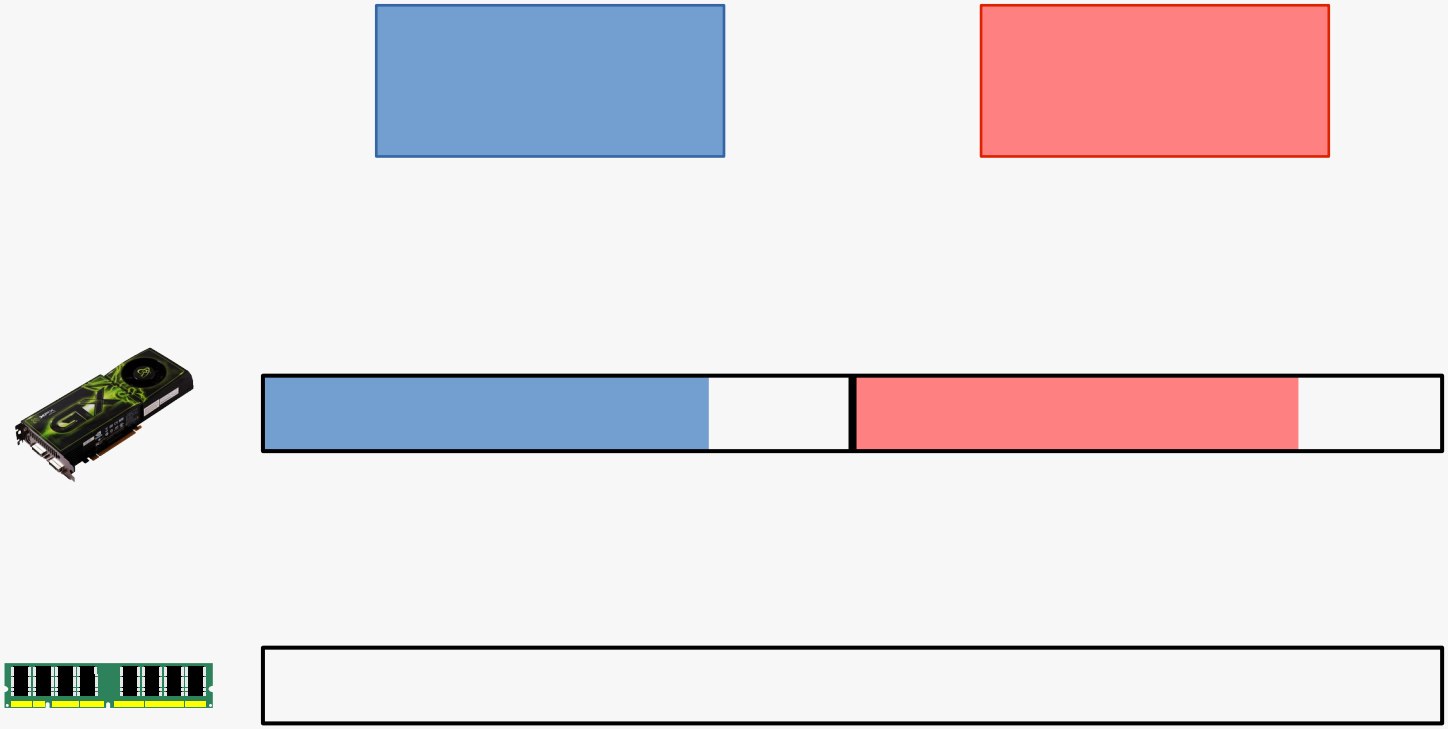
Sharing GPU Memory – FCFS

Sharing GPU Memory – FCFS

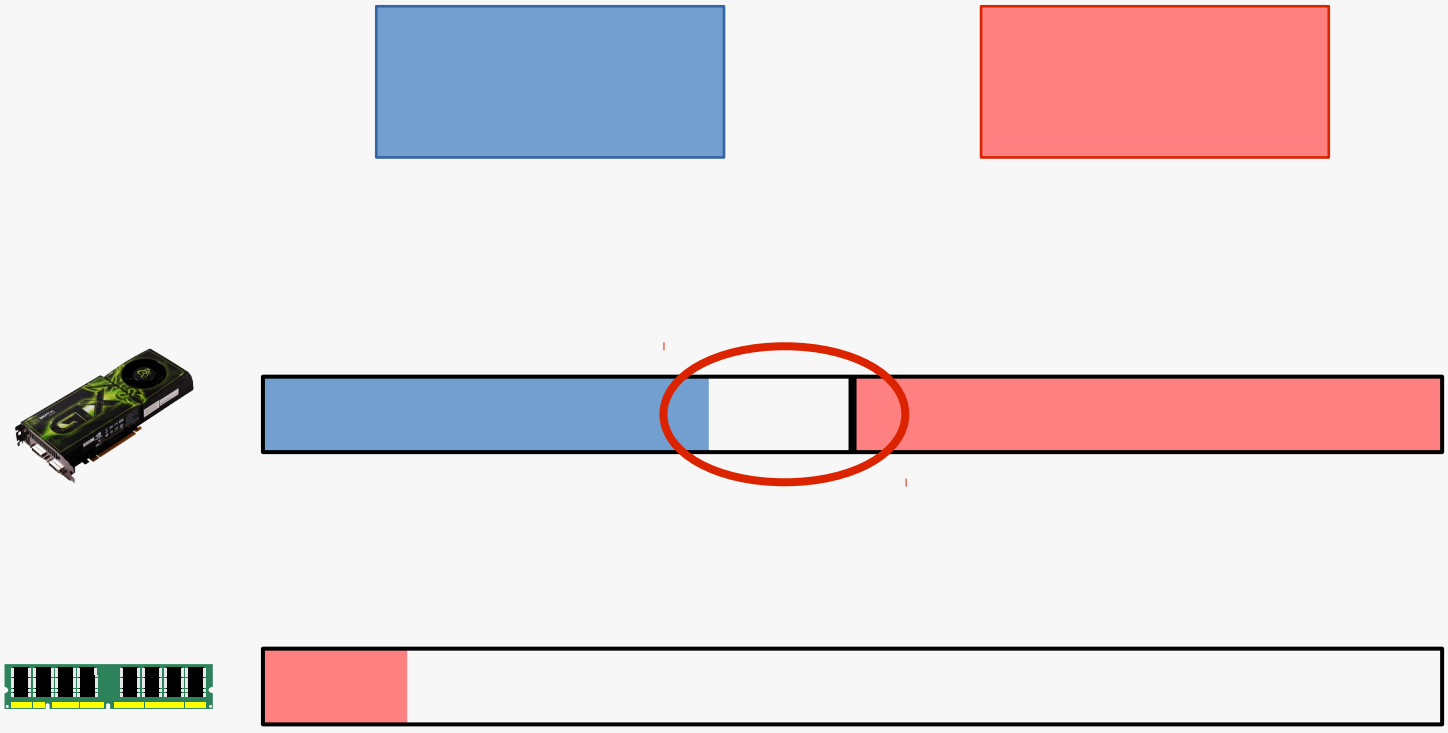


⇒ **Need to ensure fair allocation of fast GPU memory**

Sharing GPU Memory – The Naïve Way

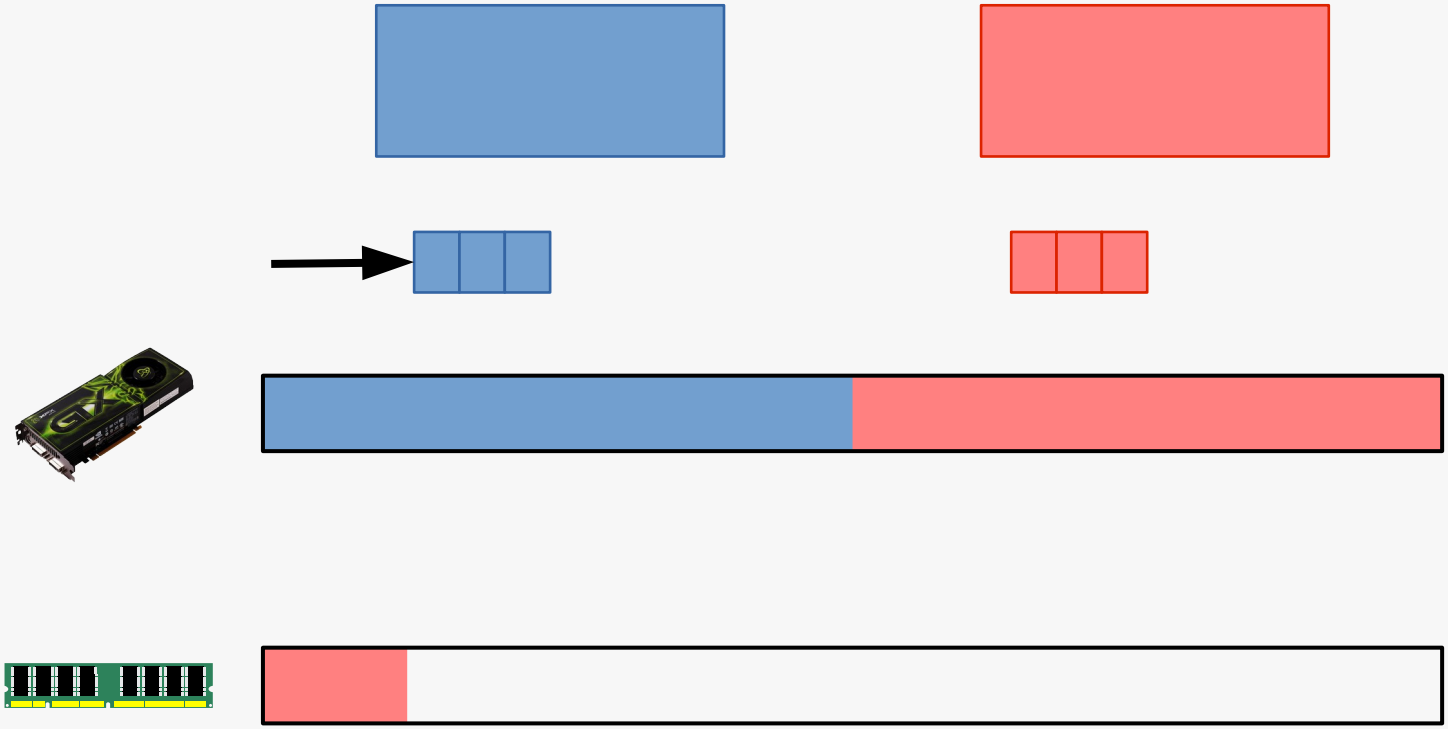


Sharing GPU Memory – The Naïve Way

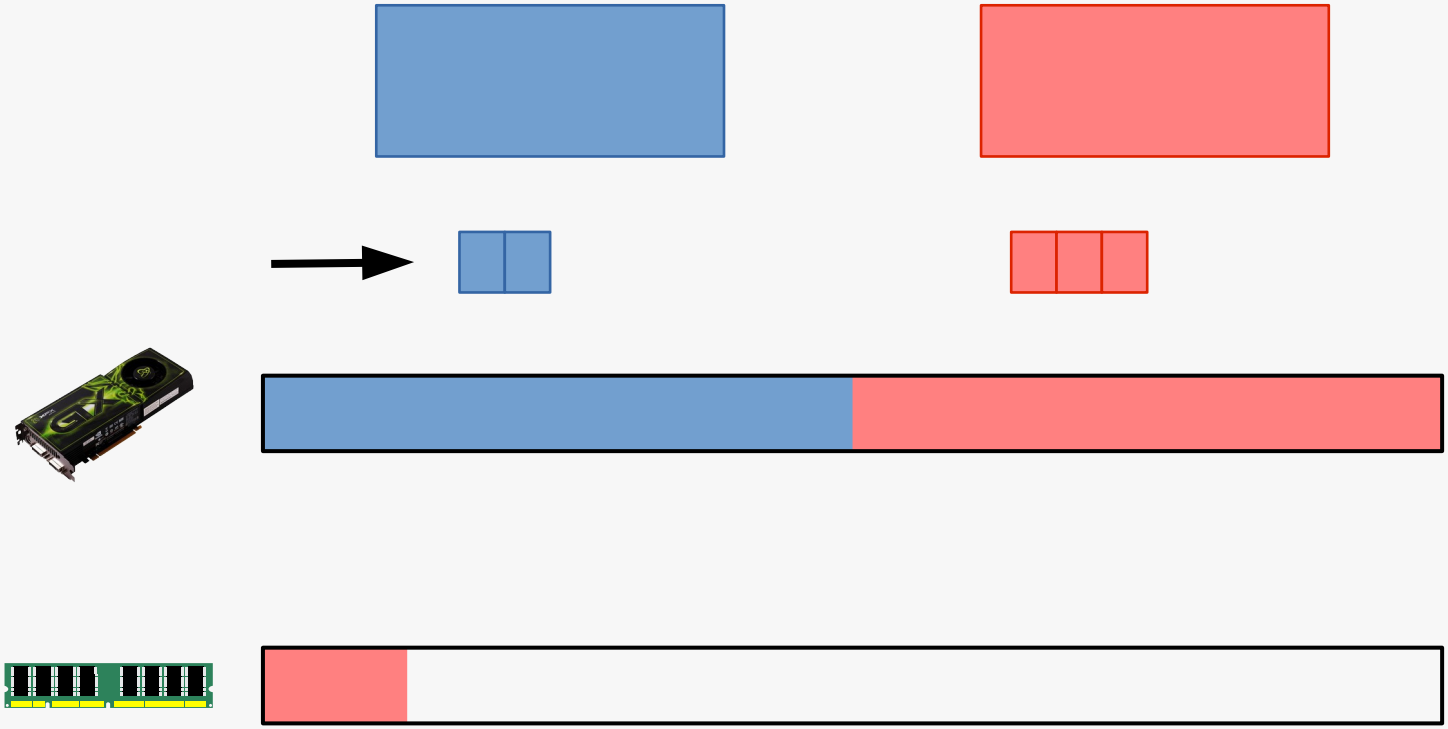


⇒ **Need to maintain good utilization of fast GPU memory**

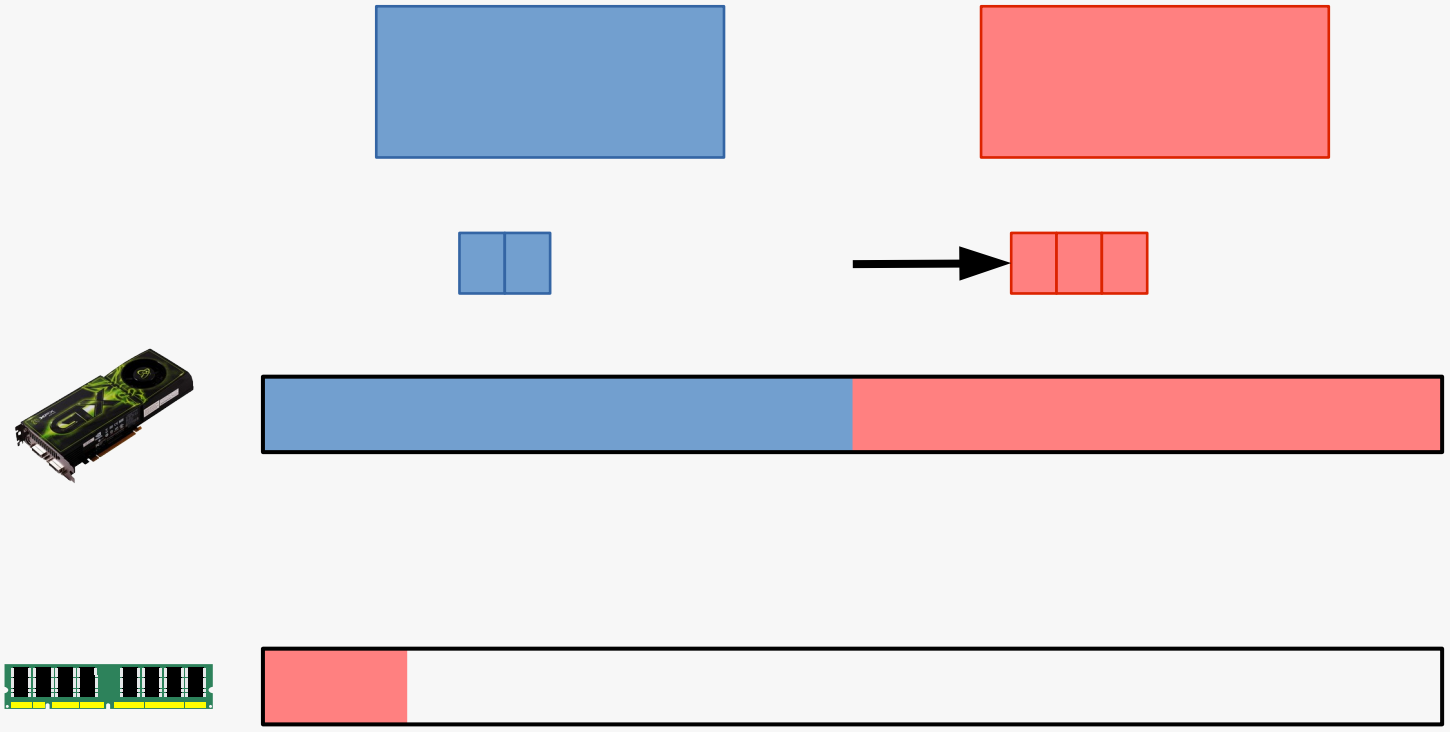
Sharing GPU Memory – Gdev (USENIX ATC '12)



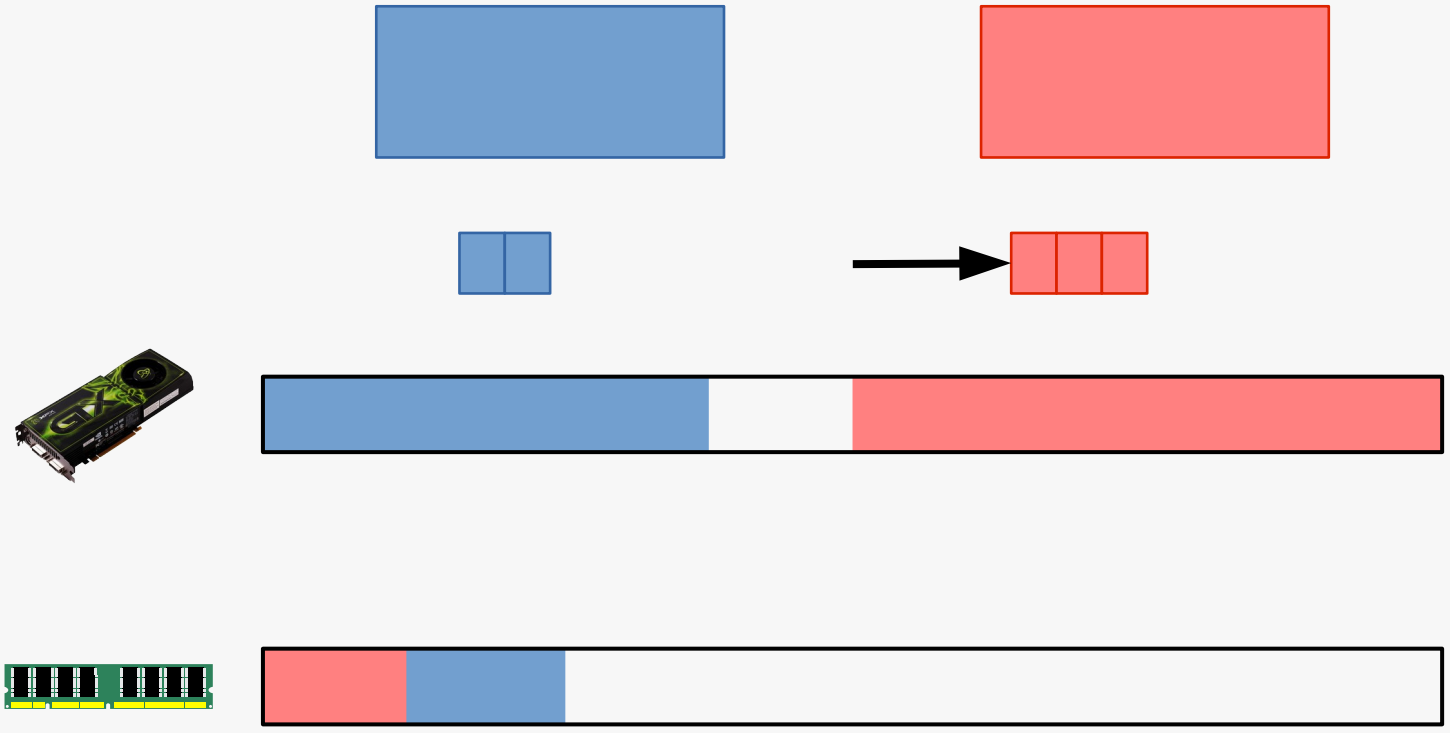
Sharing GPU Memory – Gdev (USENIX ATC '12)



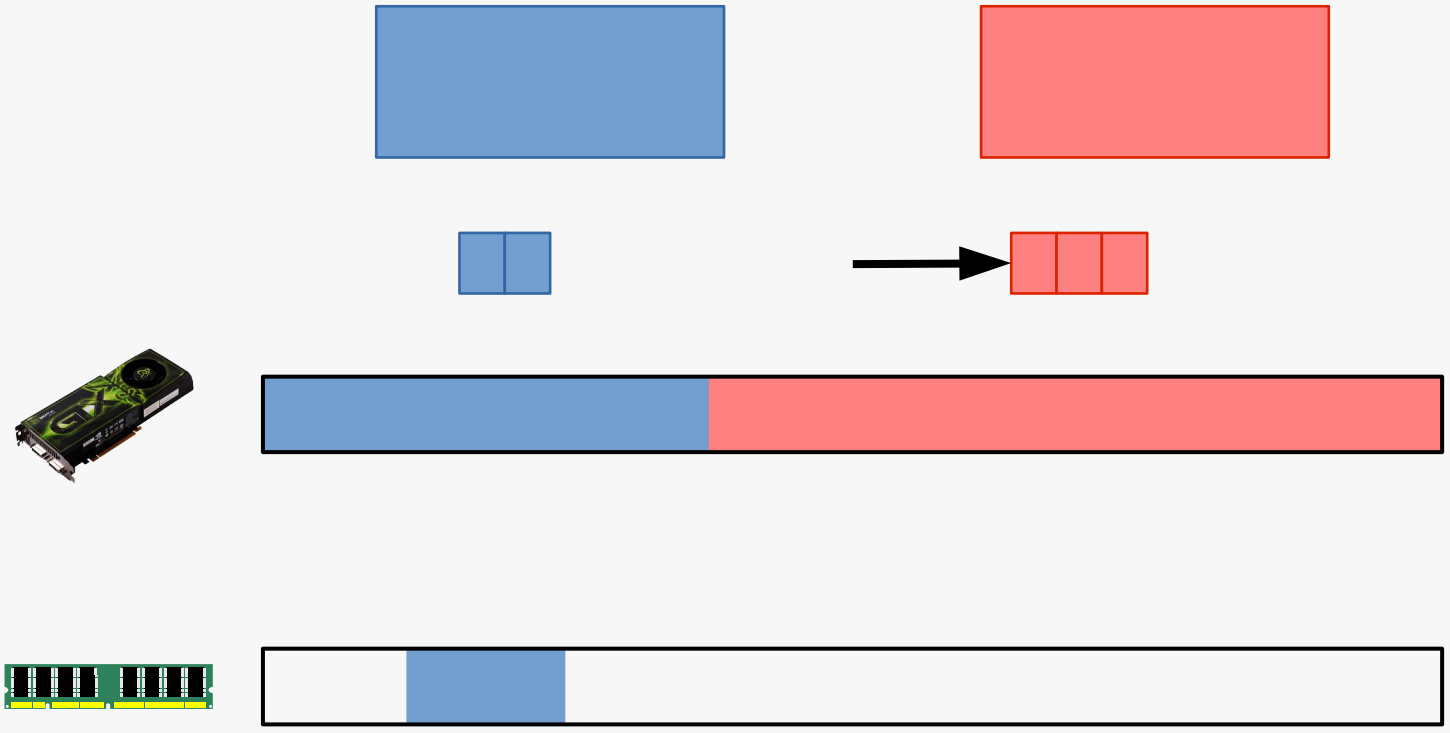
Sharing GPU Memory – Gdev (USENIX ATC '12)



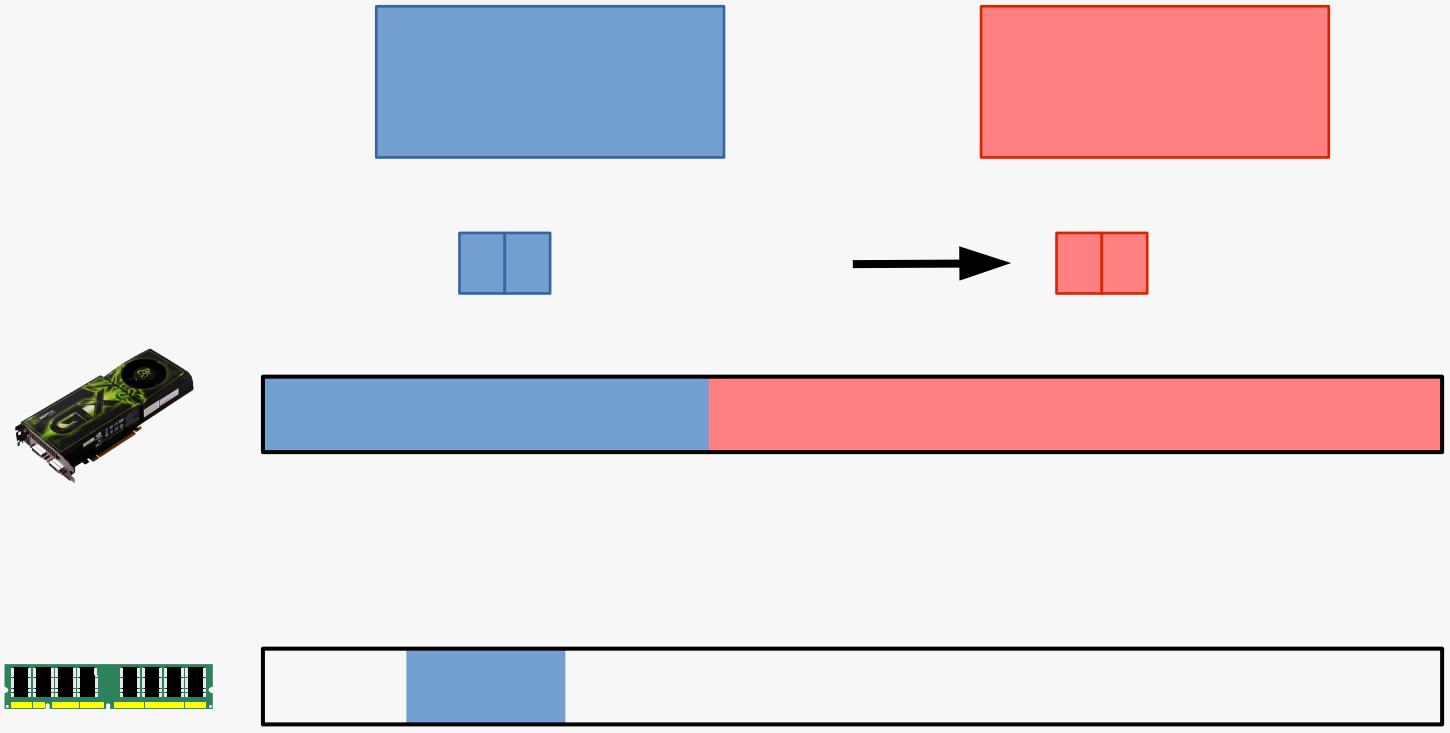
Sharing GPU Memory – Gdev (USENIX ATC '12)



Sharing GPU Memory – Gdev (USENIX ATC '12)



Sharing GPU Memory – Gdev (USENIX ATC '12)

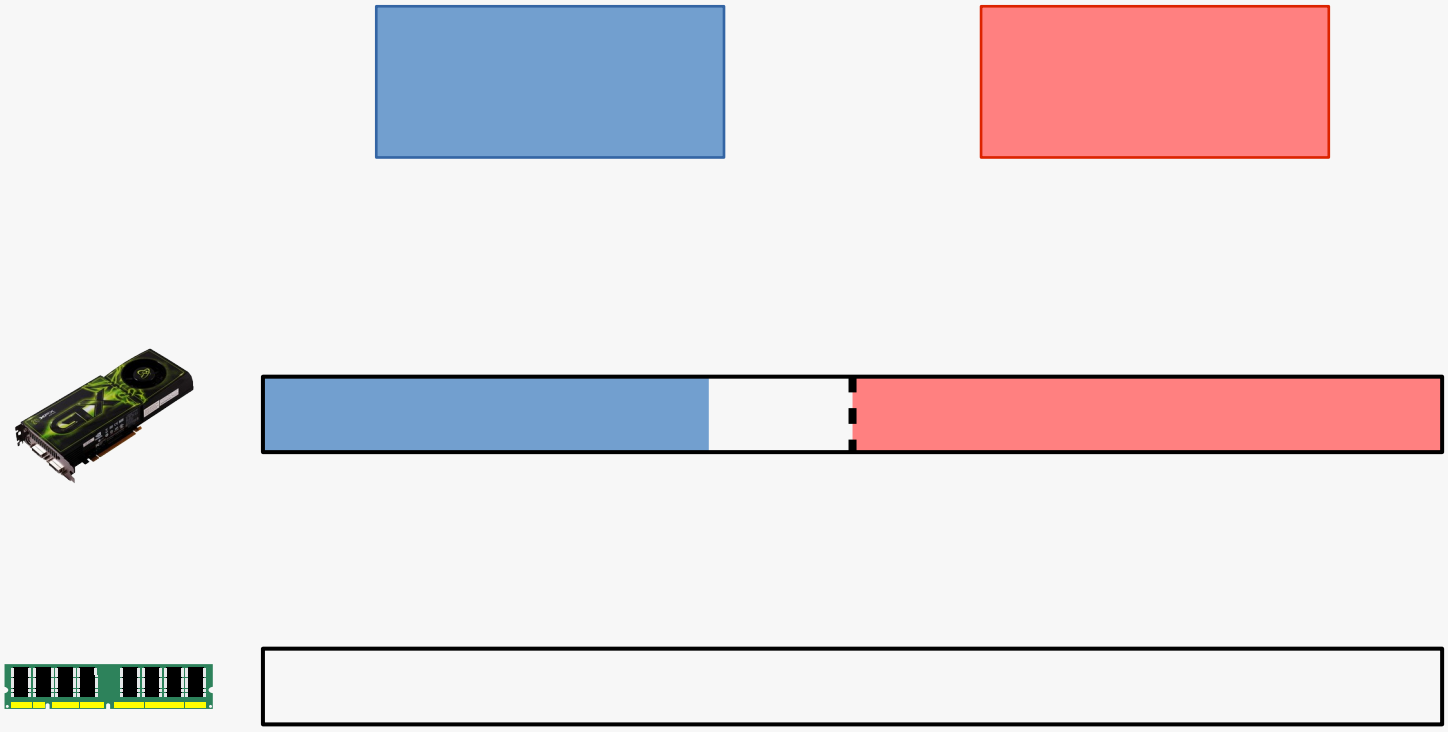


⇒ Need to avoid software scheduling overhead in the common case

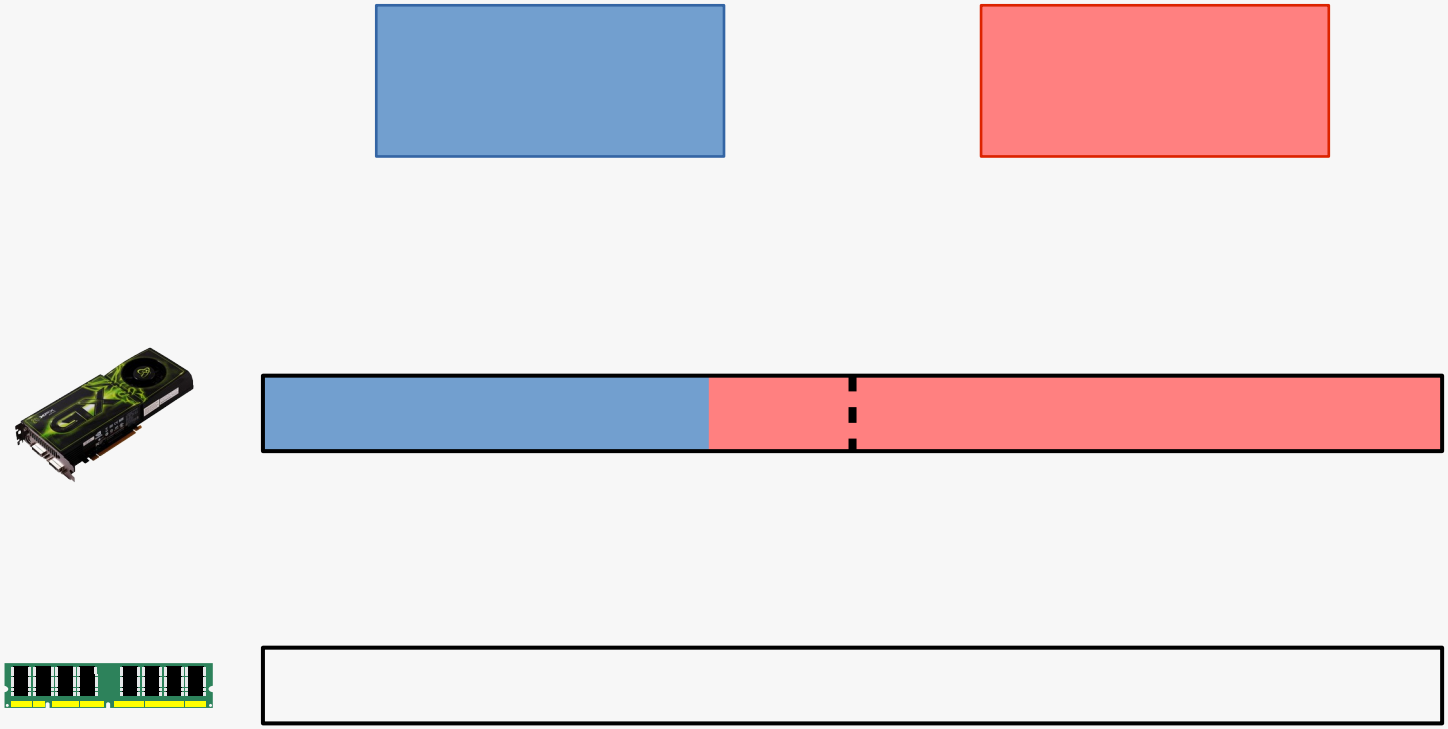
Goals of GPU Memory Sharing

- Fairness
  - Same share of memory for each application
- Performance
  - High utilization of GPU memory
  - Optimize for common case: No overhead if no over-utilization!

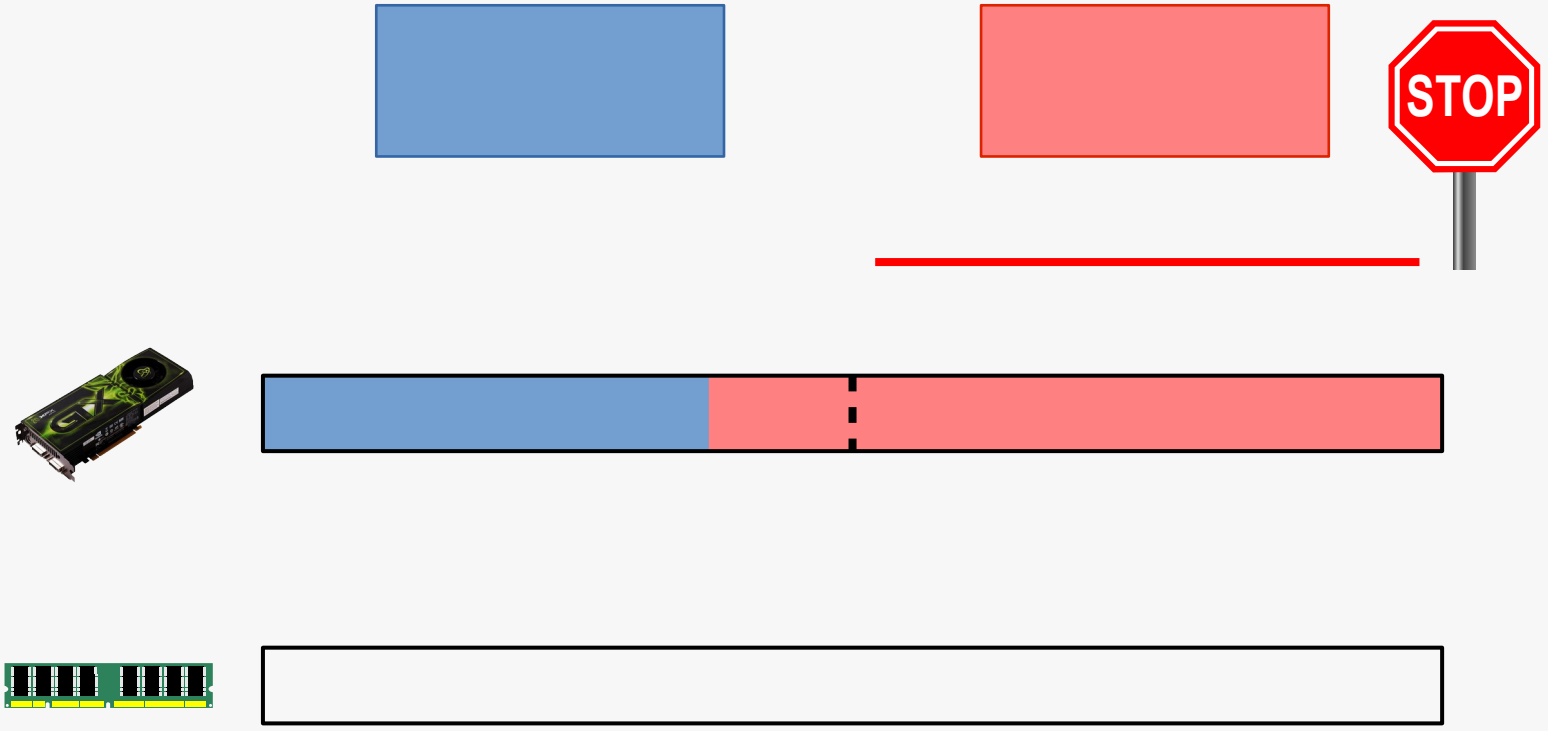
Sharing GPU Memory – Our Approach



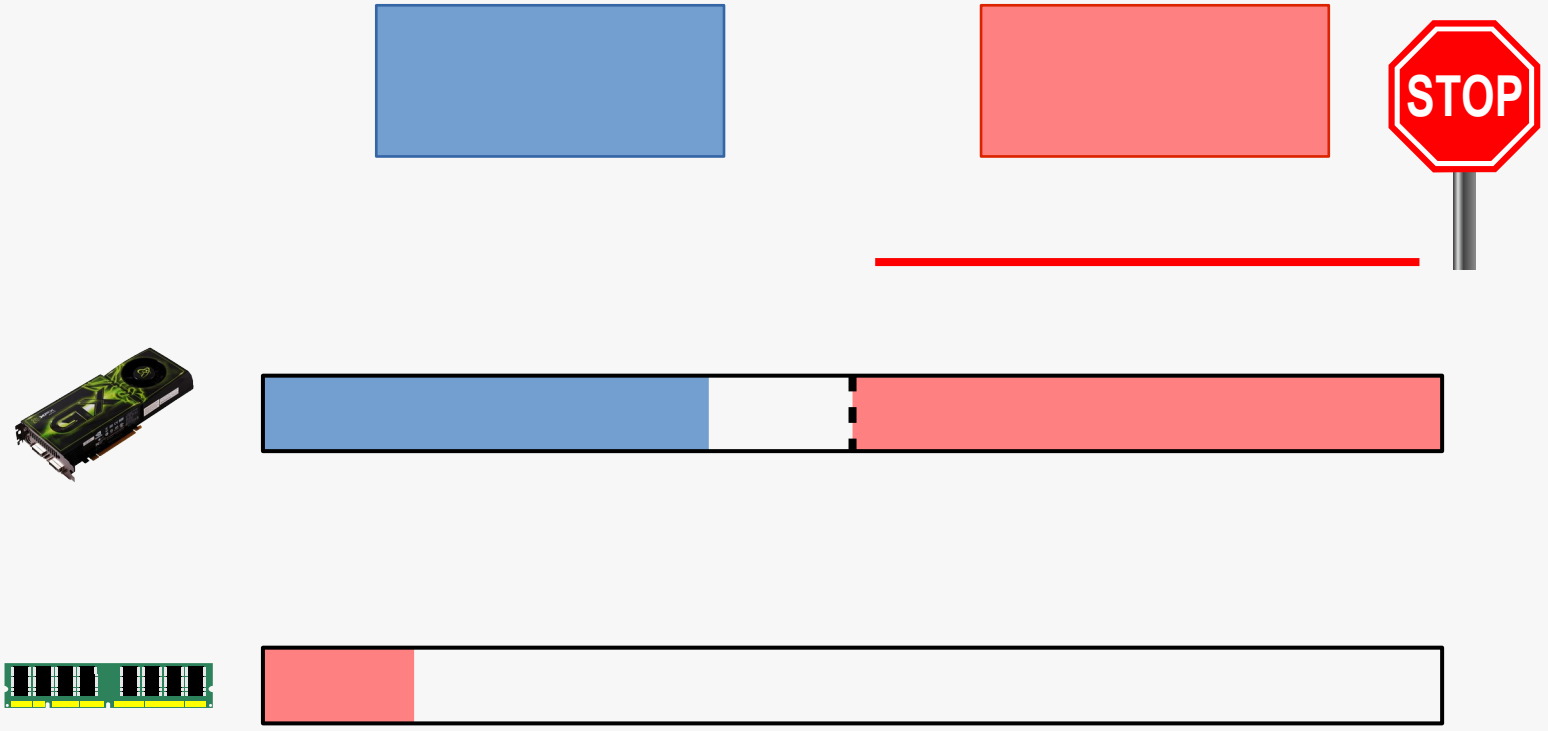
Sharing GPU Memory – Our Approach



Sharing GPU Memory – Our Approach



Sharing GPU Memory – Our Approach

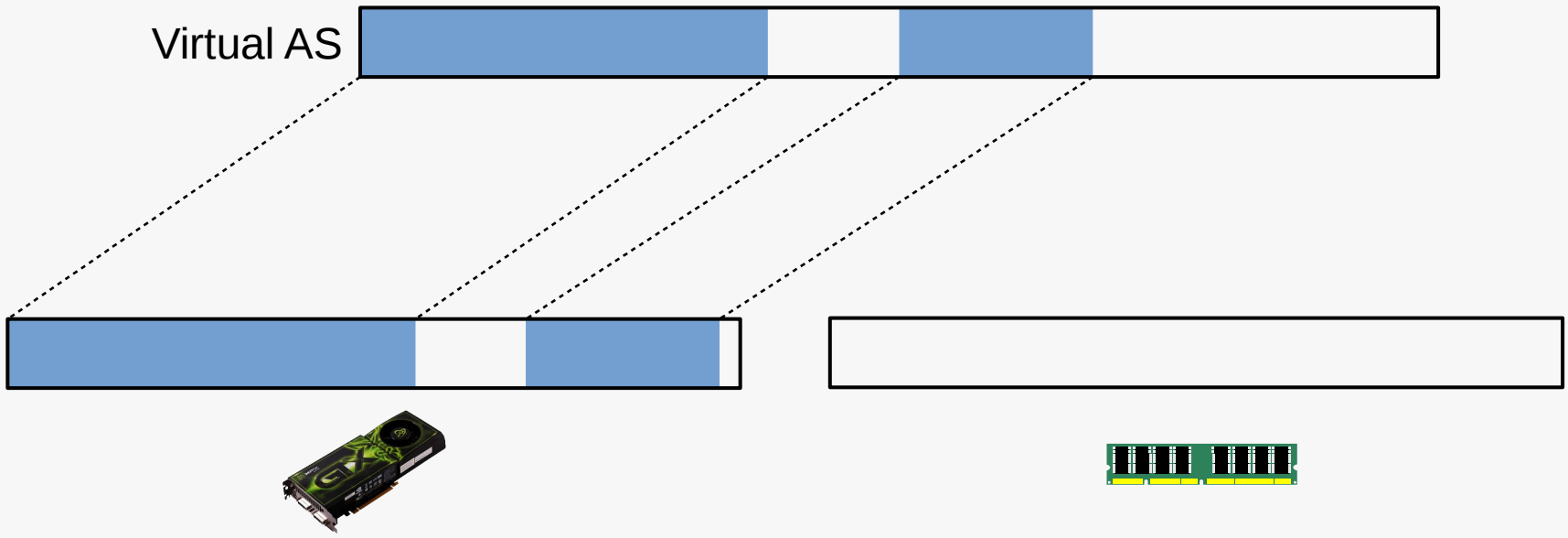


Sharing GPU Memory – Our Approach

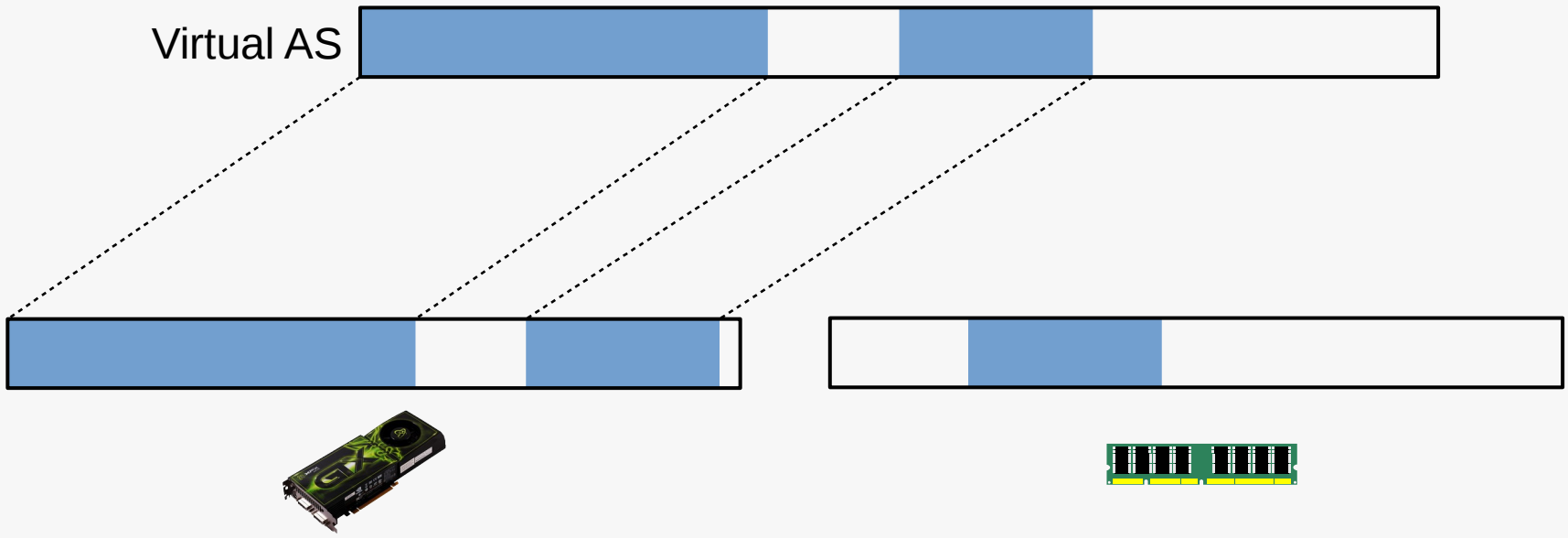
⇒ Fairness AND good utilization

⇒ No software scheduling

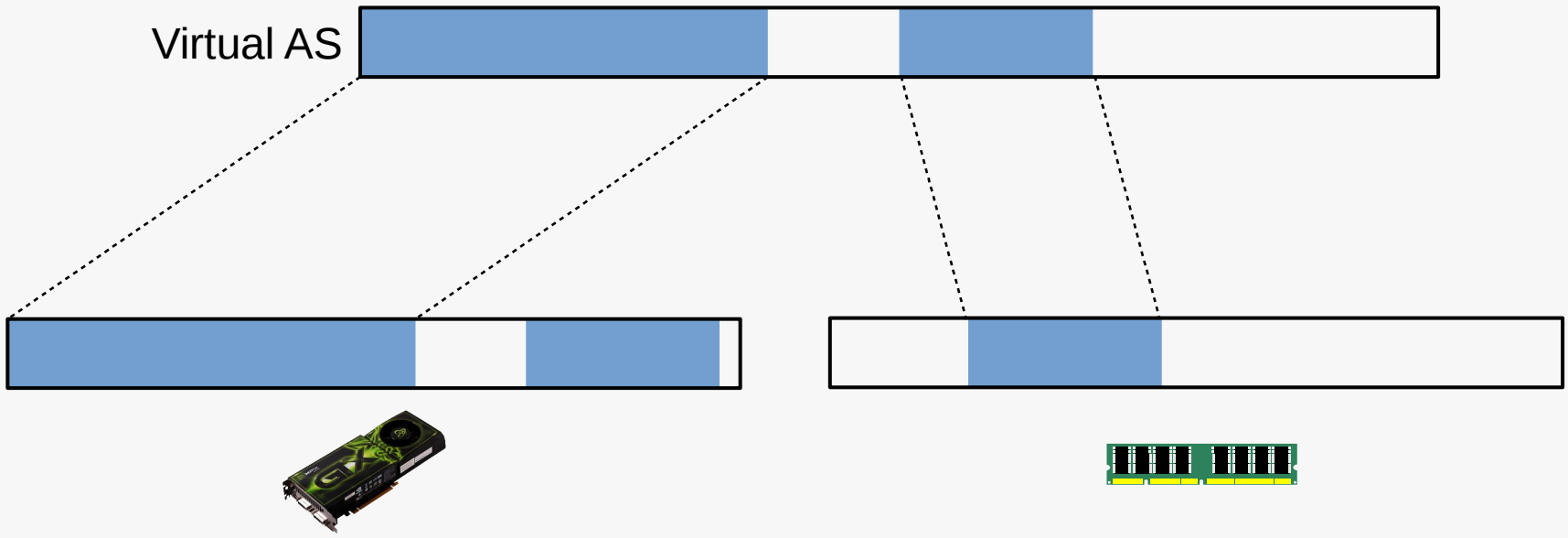
Implementation – Memory Relocation



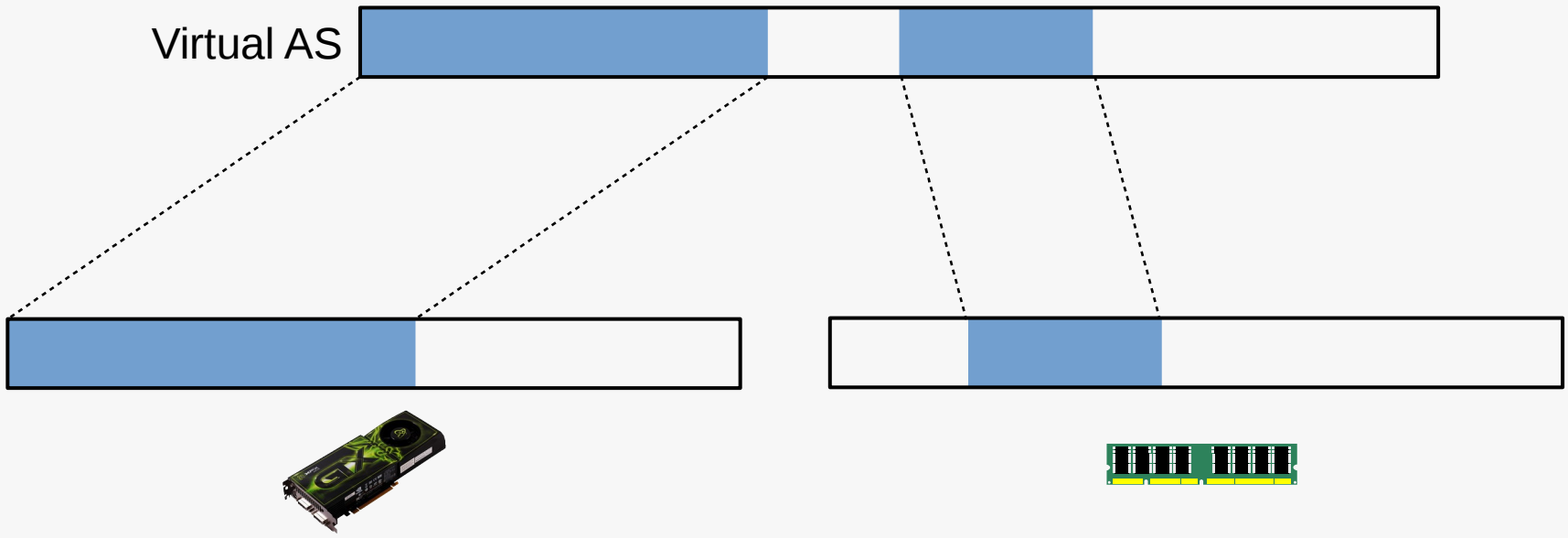
Implementation – Memory Relocation



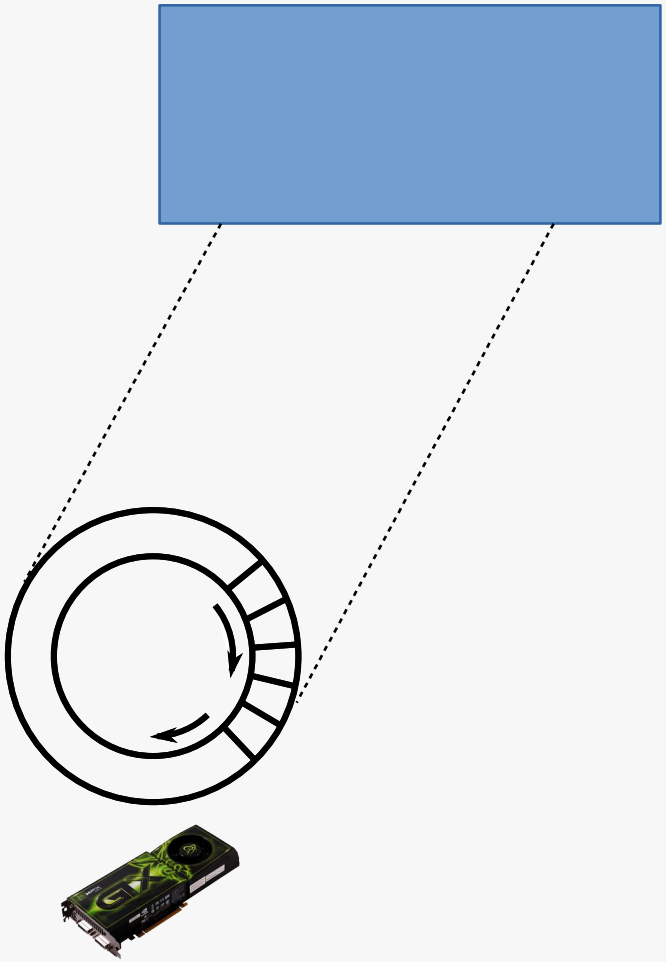
Implementation – Memory Relocation



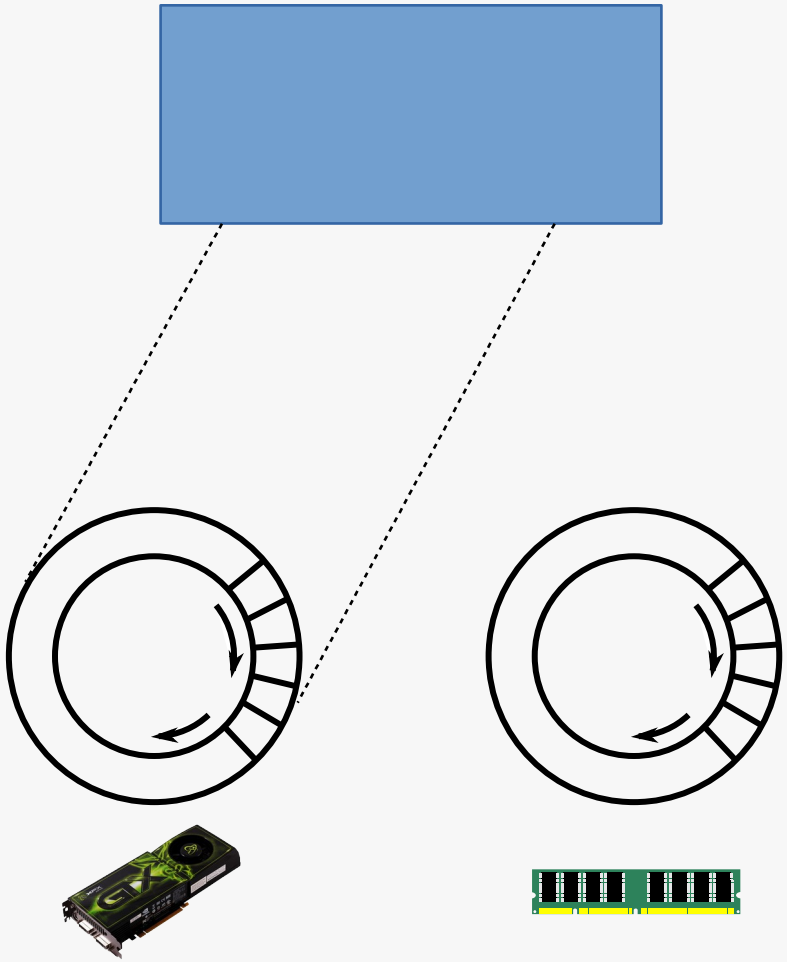
Implementation – Memory Relocation



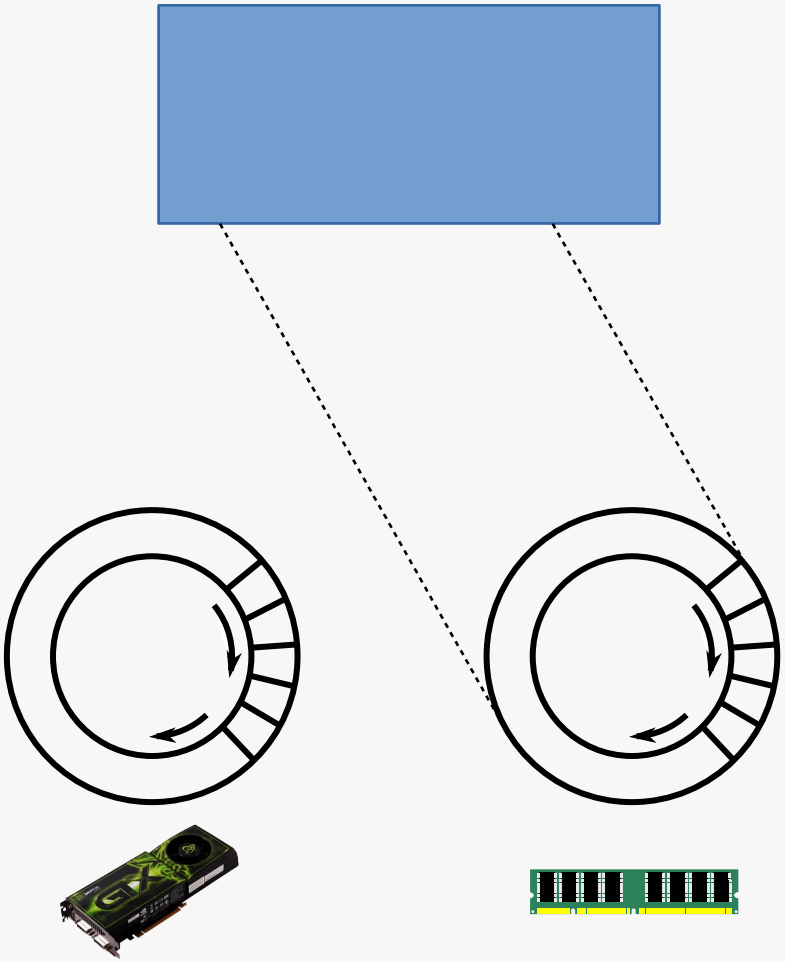
Implementation – Suspending Applications



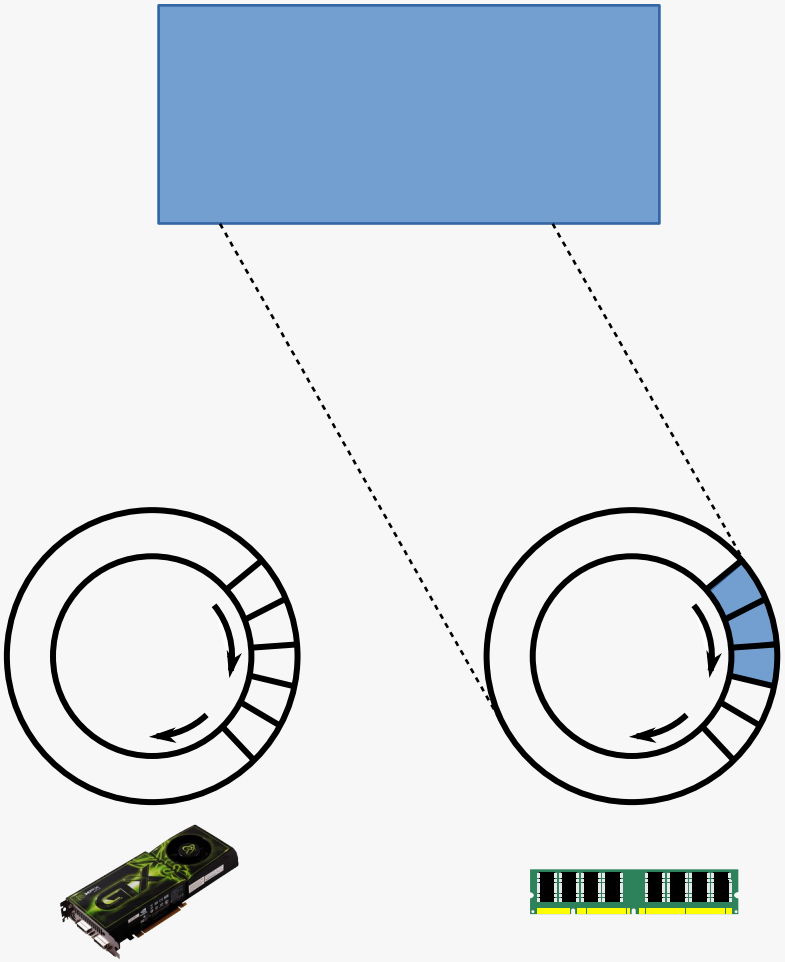
Implementation – Suspending Applications



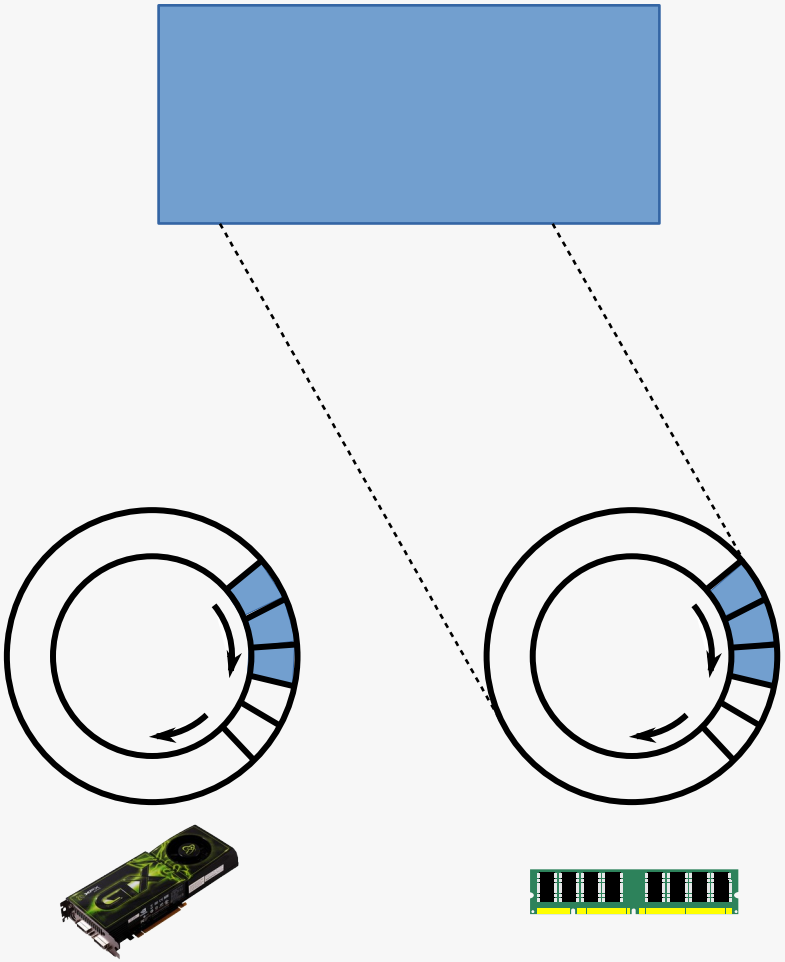
Implementation – Suspending Applications



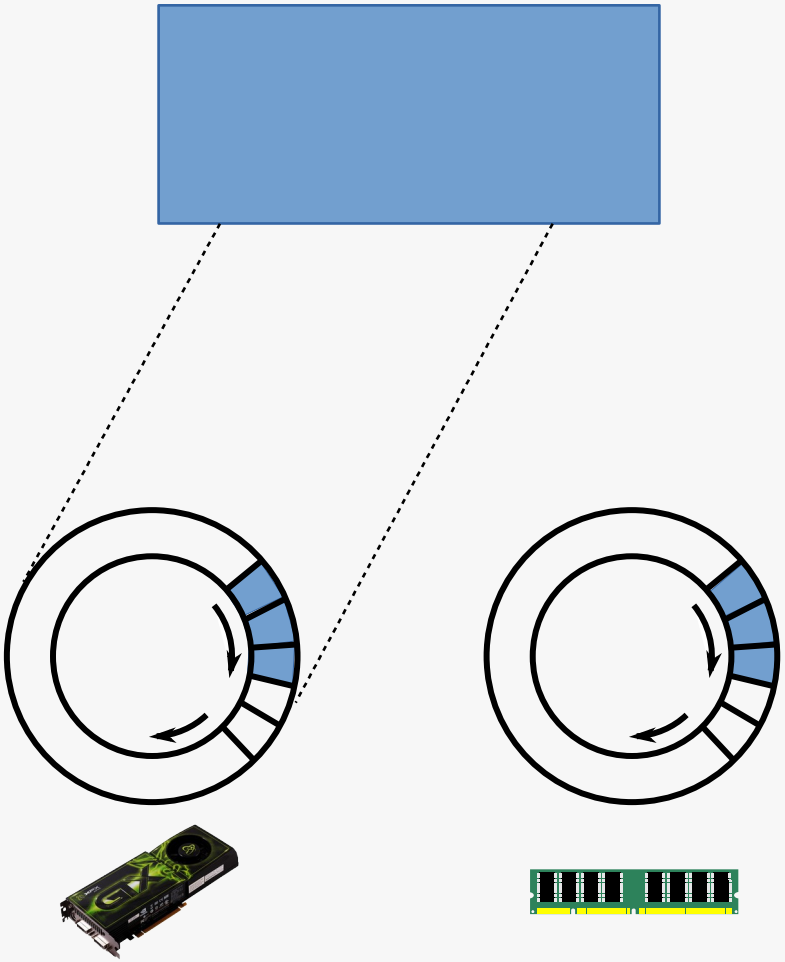
Implementation – Suspending Applications



Implementation – Suspending Applications



Implementation – Suspending Applications



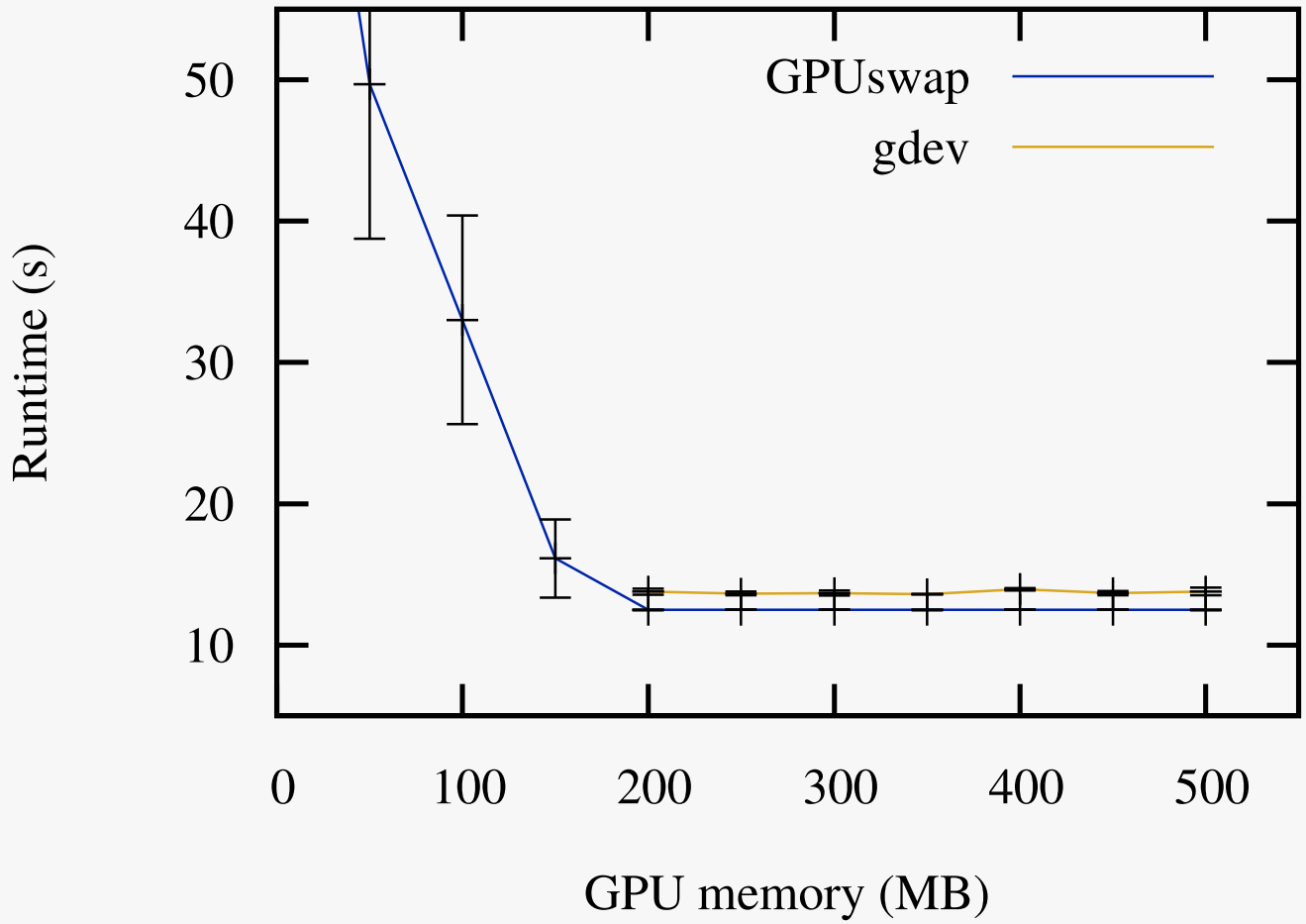
Swapping Policy

- Choose app with most GPU memory („The Victim“)
  - Achieves fairness
- Choose chunk of memory from victim‘s AS

- How do we find the right chunk?

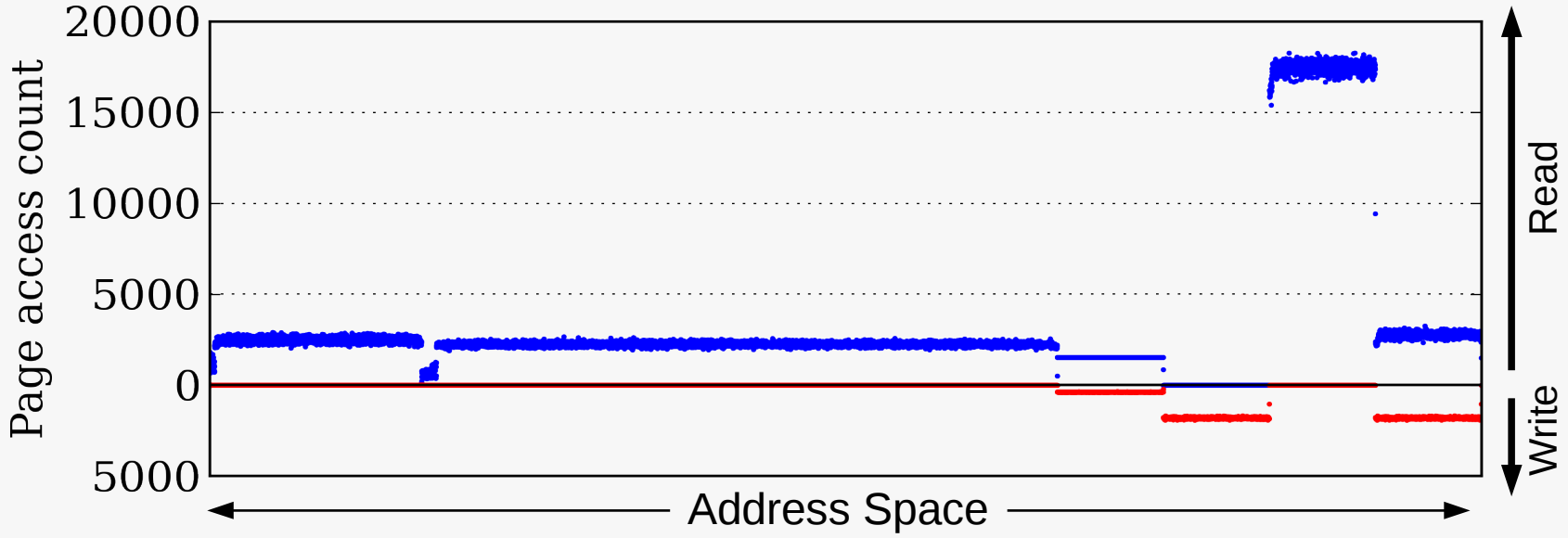
- Original implementation: Random

Results: Runtime Overhead (lud)



Hot and Cold Pages (BFS)

- Use GPU performance counters
- Measure number of accesses per page

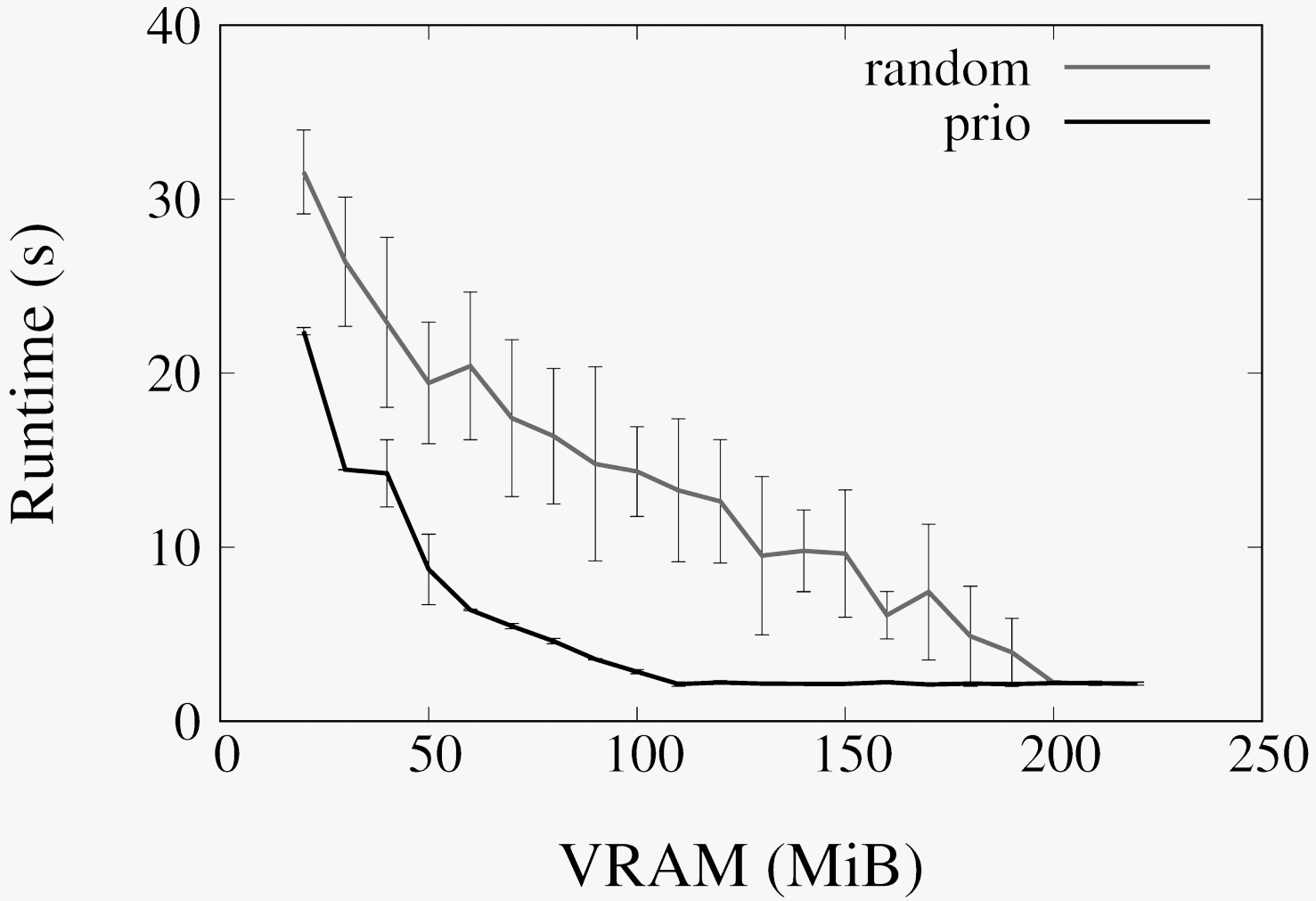


Swapping Policy Mark II

- Idea: Assign a priority to each buffer
  - Profile application
  - Buffers with more accesses per page get higher priority

- Policy:
  - Choose victim (application with most GPU memory)
  - Choose buffer with lowest priority from victim‘s AS
  - Pick chunk from that buffer randomly

Results: Buffer priorities (BFS)



GPUswap

… enables oversubscription of GPU memory

… achieves

- Fairness
- Performance

… optimizes for the common case

- No software scheduling

Credits

- GPU clip art: © Calvin Souto, 2008, GFDL