Supporting I/O and IPC via Fine-Grained OS Isolation for Mixed-Criticality Real-Time Tasks

•••

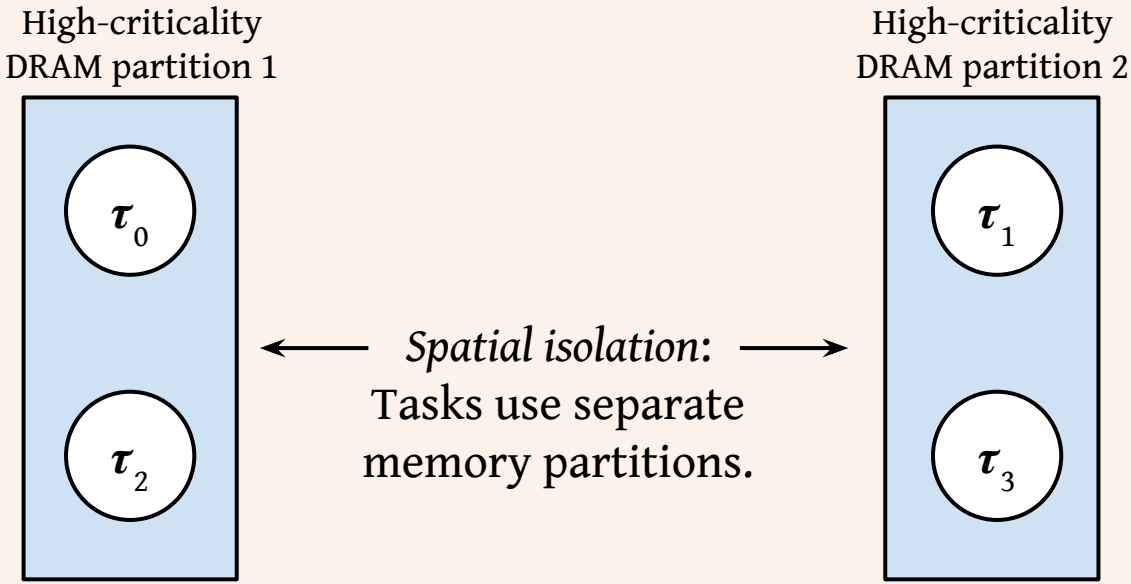
Namhoon Kim, Stephen Tang, Nathan Otterness, James H. Anderson, F. Donelson Smith, and Donald E. Porter

The One-out-of-*m* Problem

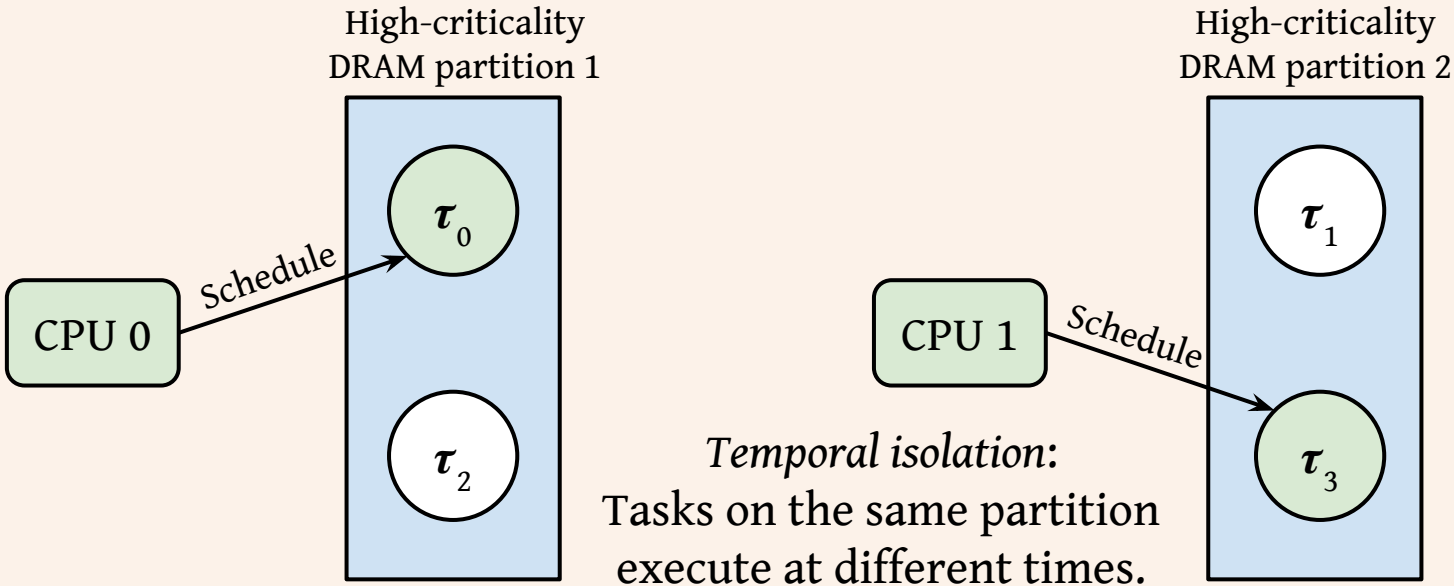
- An *m*-core system may lose the capacity of *m* - 1 cores due to hardware interference and pessimistic worst-case analysis.
- MC^2 (***M***ixed-***C***riticality on ***M***ulti***c***ore) addresses the one-out-of-*m* problem.
  - Reduce interference using *hardware isolation*.
  - Reduce pessimism using *mixed-criticality analysis*.

Illustration: A Simple 4-task System

Spatial Isolation

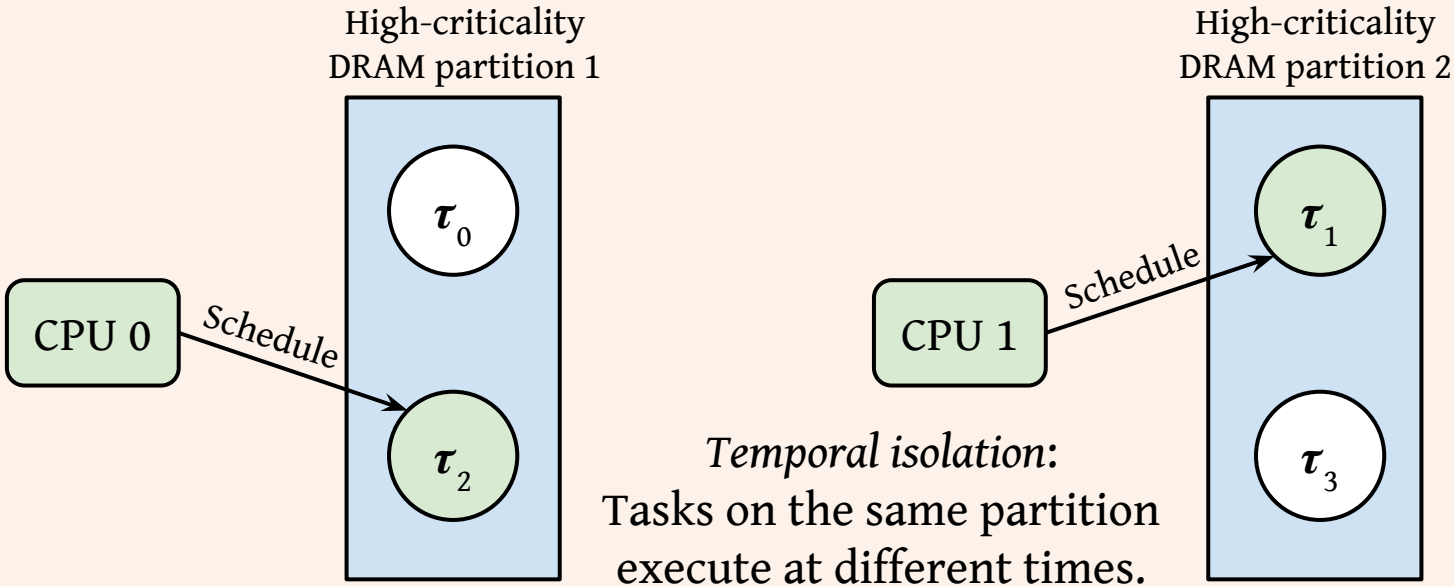


Temporal Isolation



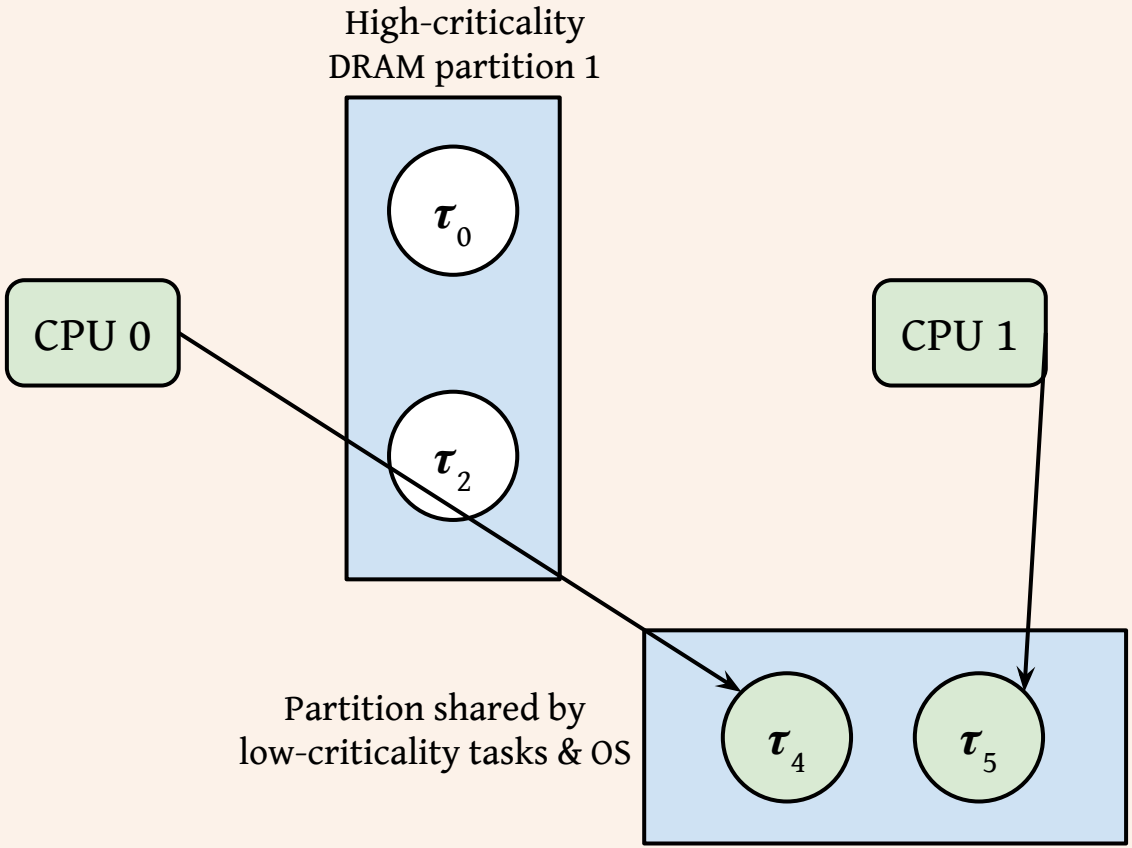
Temporal isolation:
Tasks on the same partition execute at different times.

Temporal Isolation



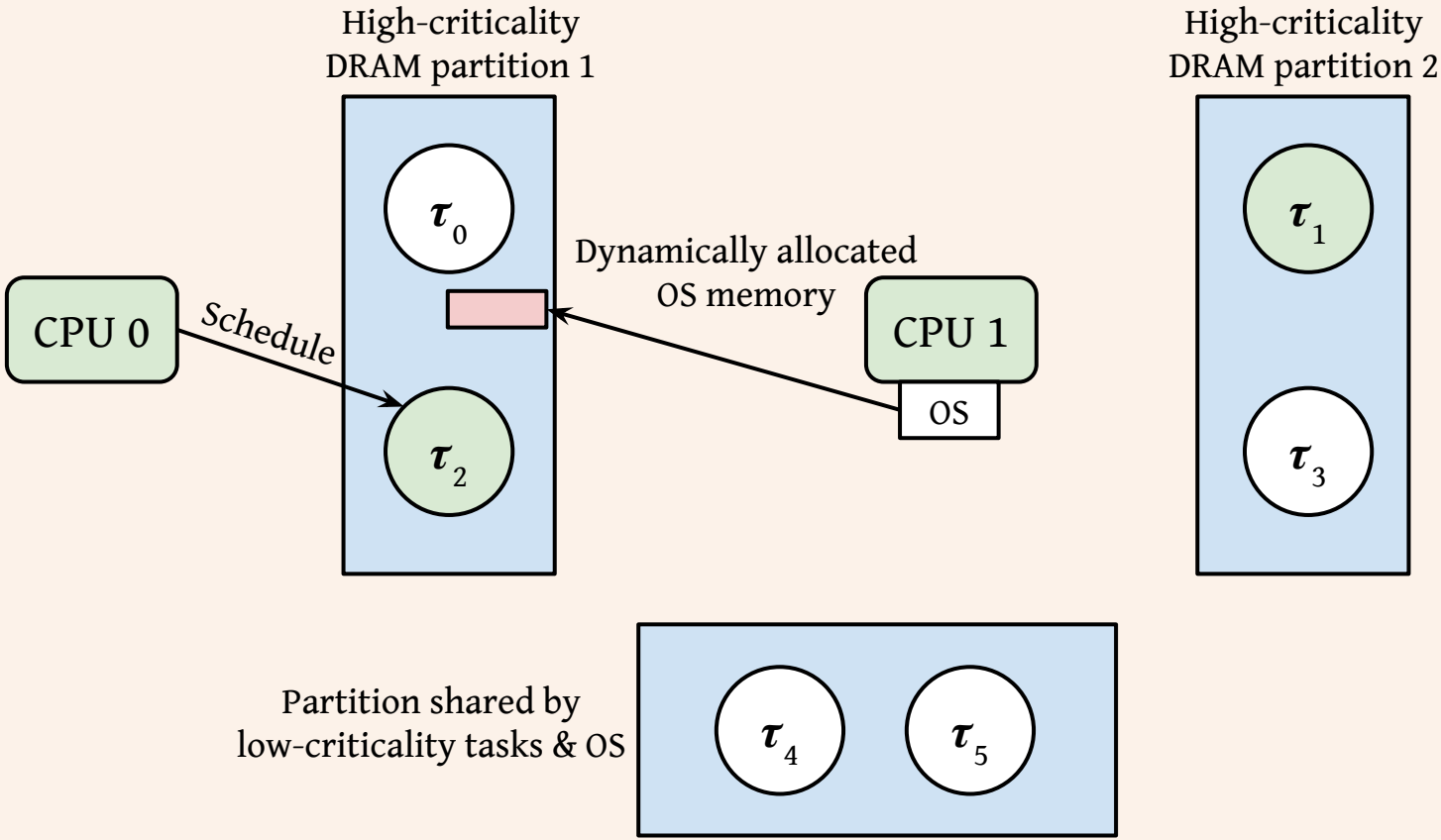
Temporal isolation:
Tasks on the same partition execute at different times.

Mixed-Criticality in MC^2

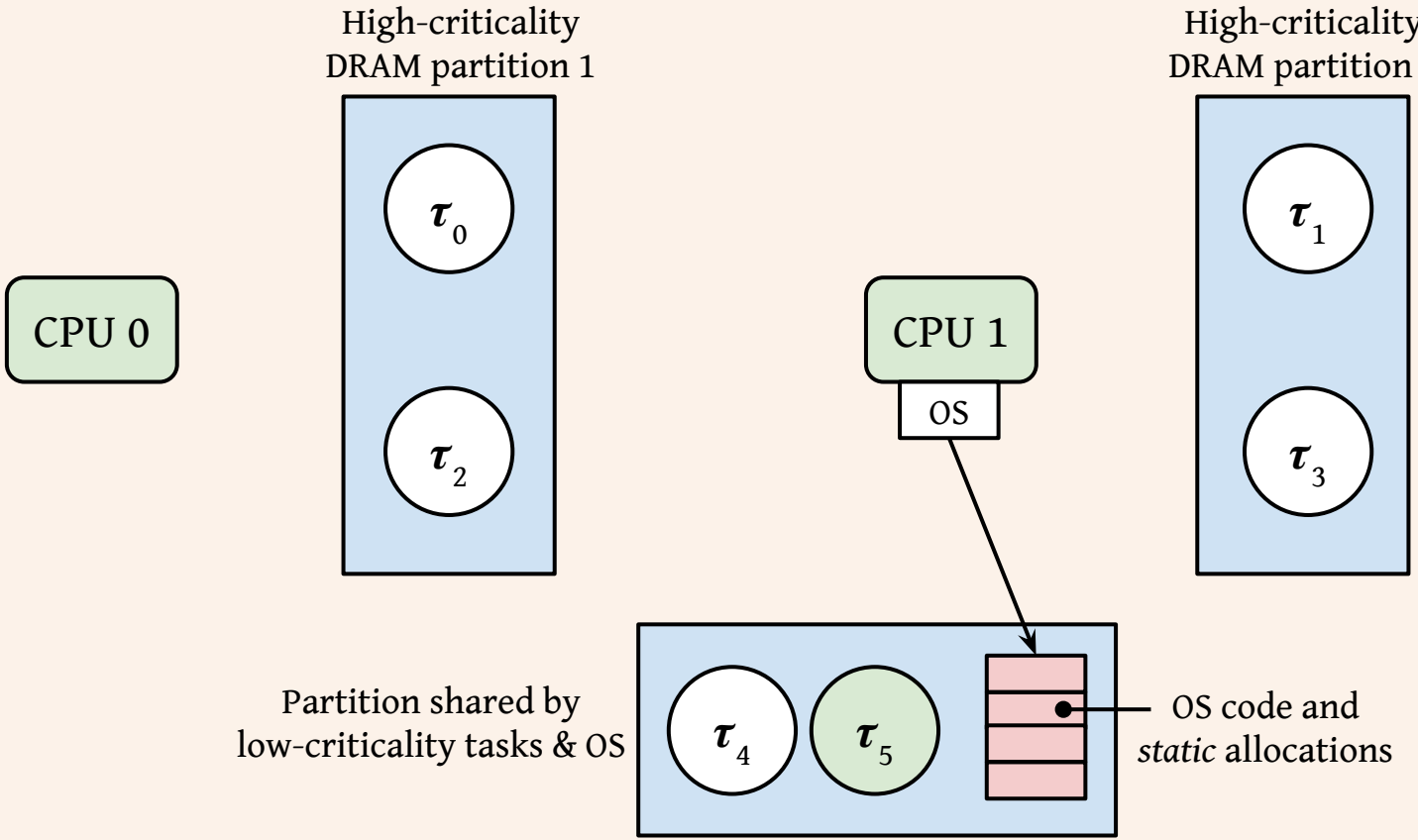


Mixed criticality:
Lower-criticality tasks deal with more interference, but less pessimistically.

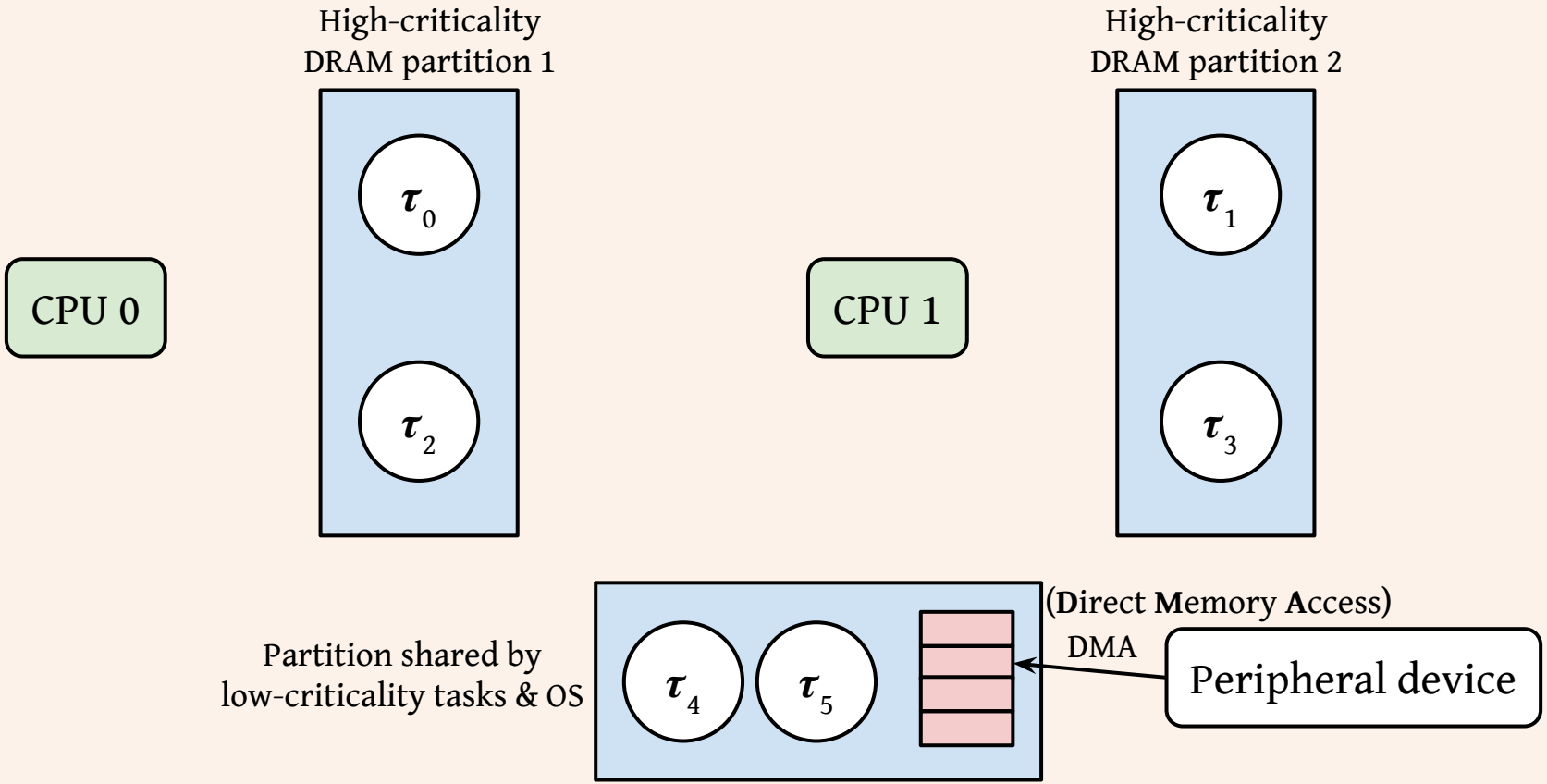
(Not) Handling the Operating System



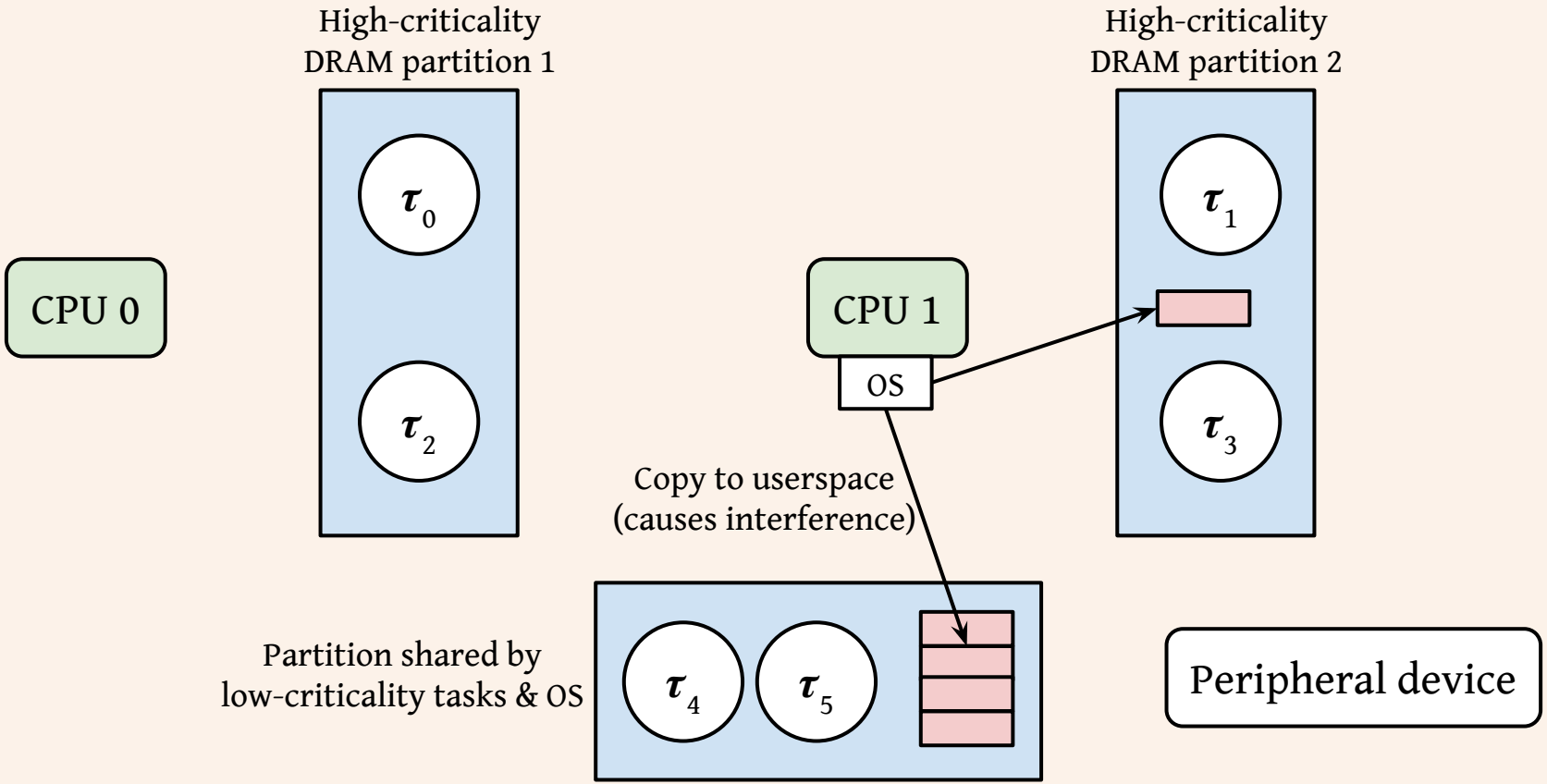
(Not) Handling the Operating System



(Not) Handling Device I/O



(Not) Handling Device I/O



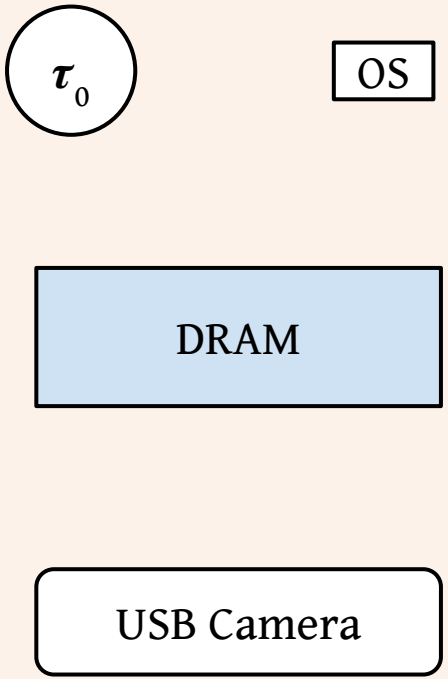
Goal of this Project

Prior versions of MC^2 isolated task memory, but MC^2 must address sharing via the OS and peripheral devices.

Types of Hardware Interference Addressed

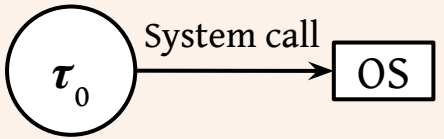
- CPU-sourced interference
  - Unmanaged IPC
  - Copying data from DMA buffers
- DMA-sourced interference
  - Devices directly accessing DRAM

Example: Interference Sources from USB I/O



Example: Interference Sources from USB I/O

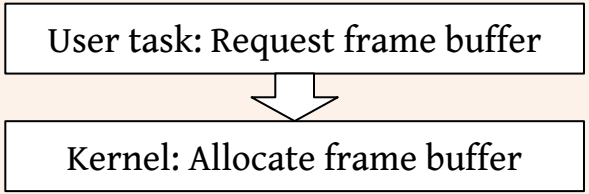
User task: Request frame buffer

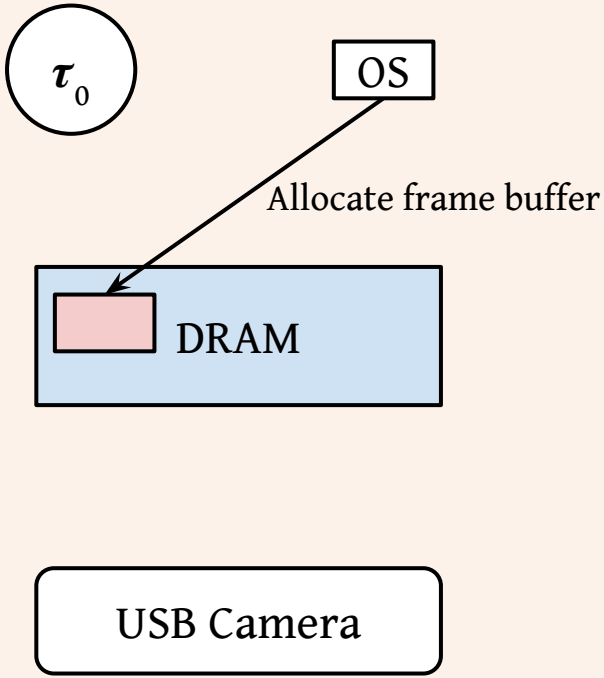


DRAM

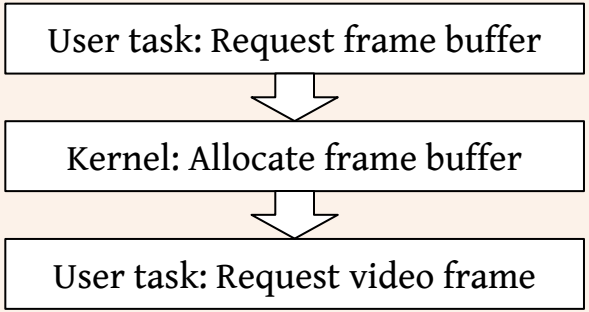
USB Camera

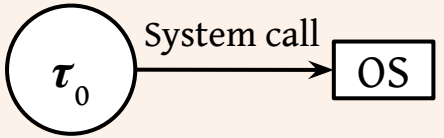
Example: Interference Sources from USB I/O



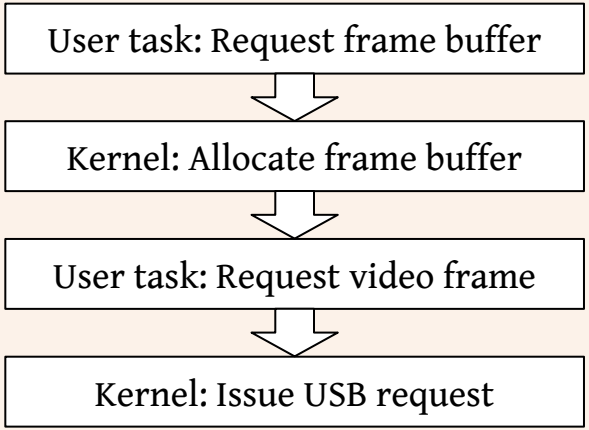


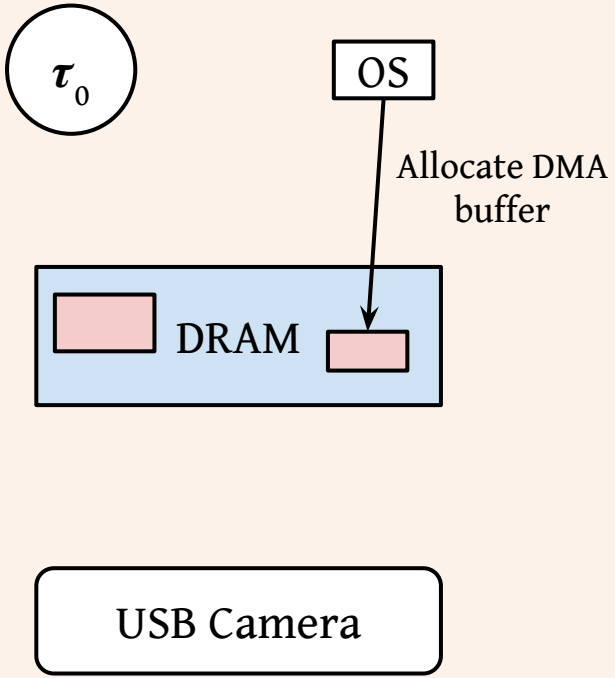
Example: Interference Sources from USB I/O



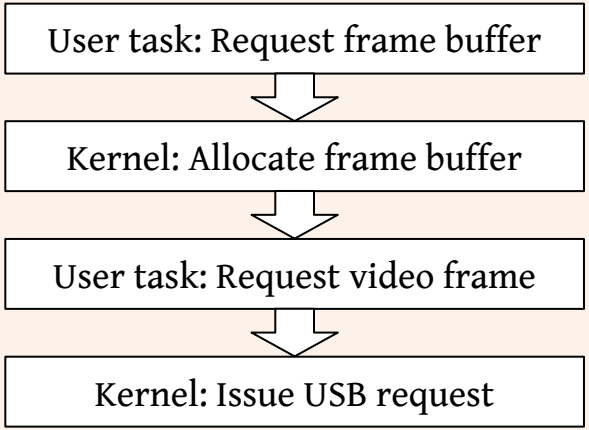


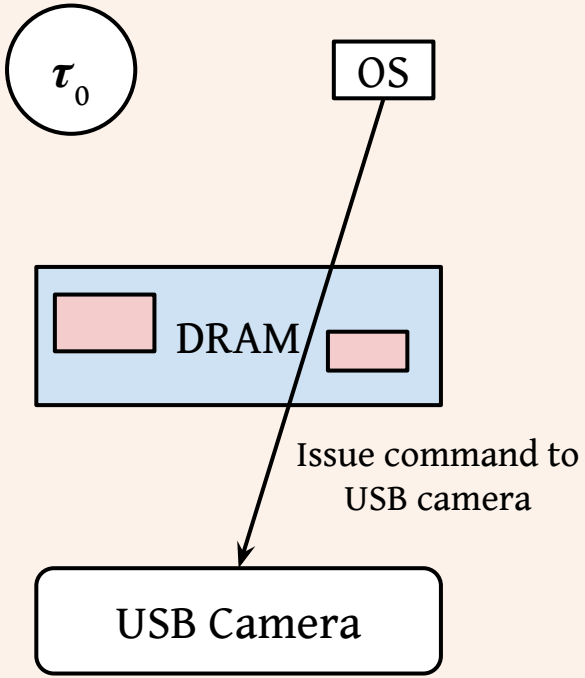
Example: Interference Sources from USB I/O



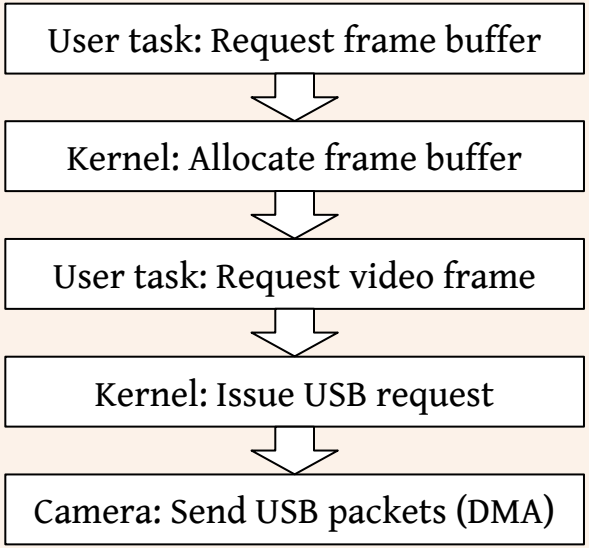


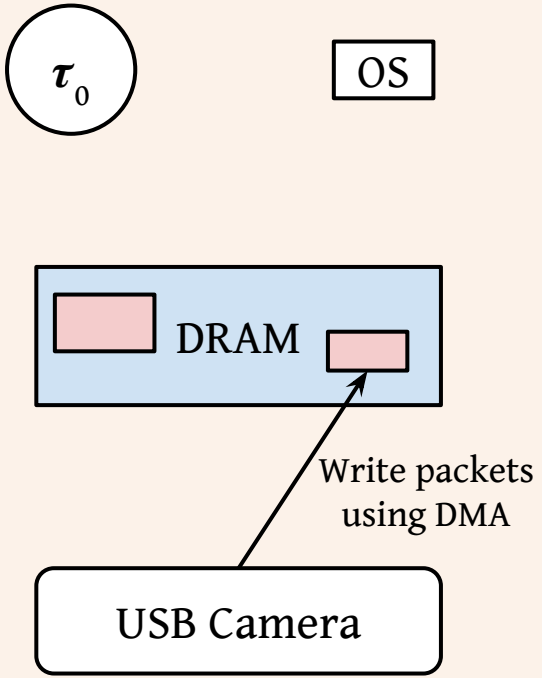
Example: Interference Sources from USB I/O



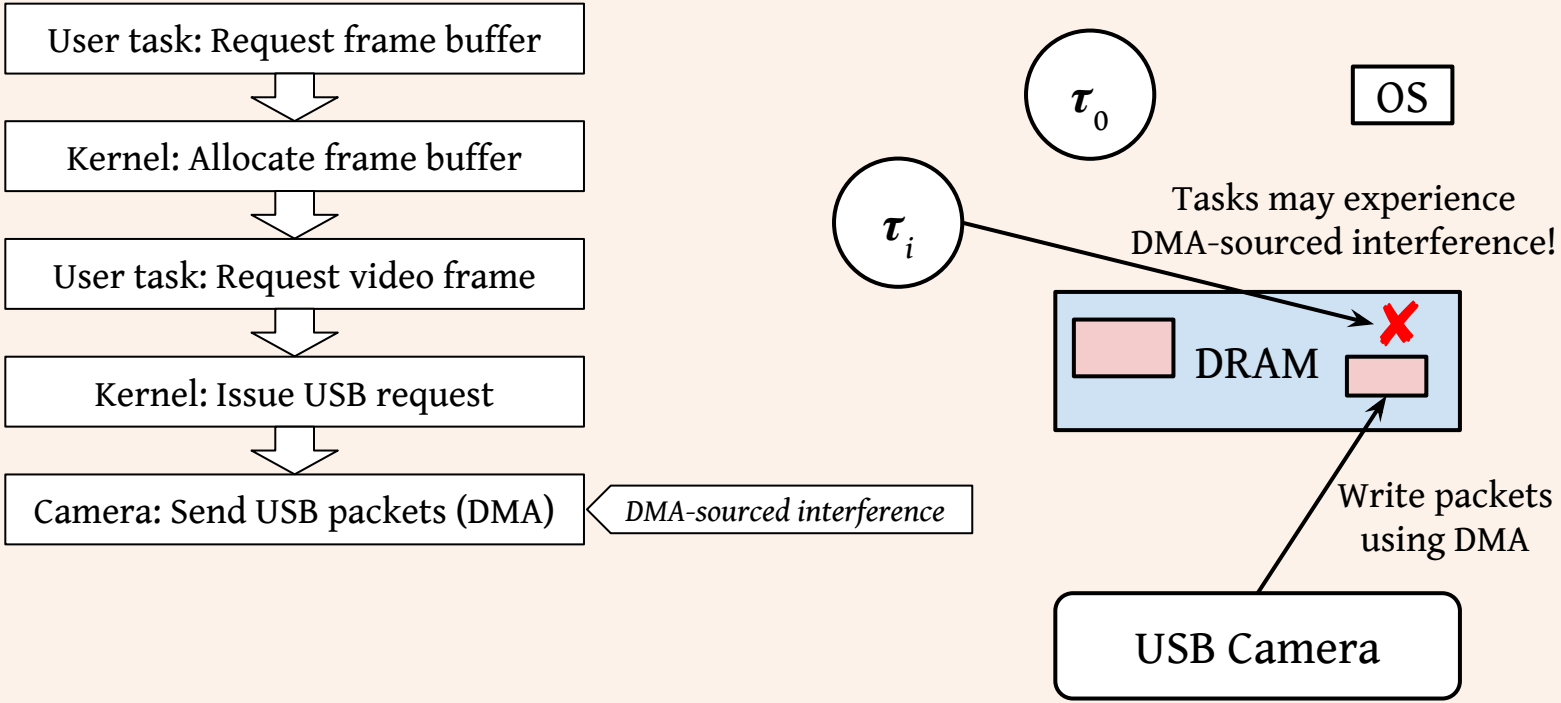


Example: Interference Sources from USB I/O

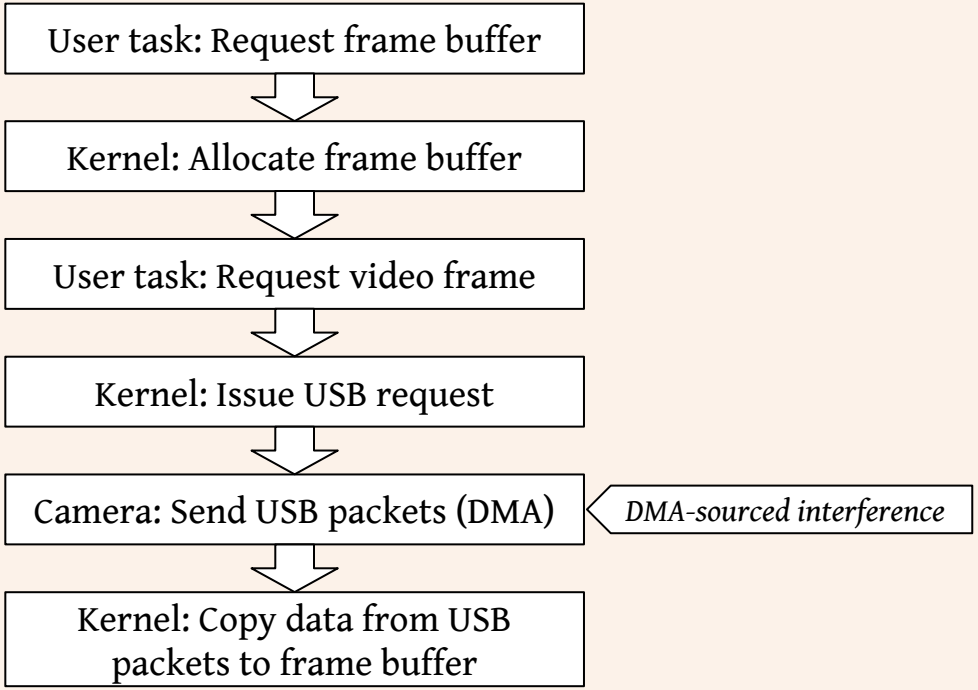


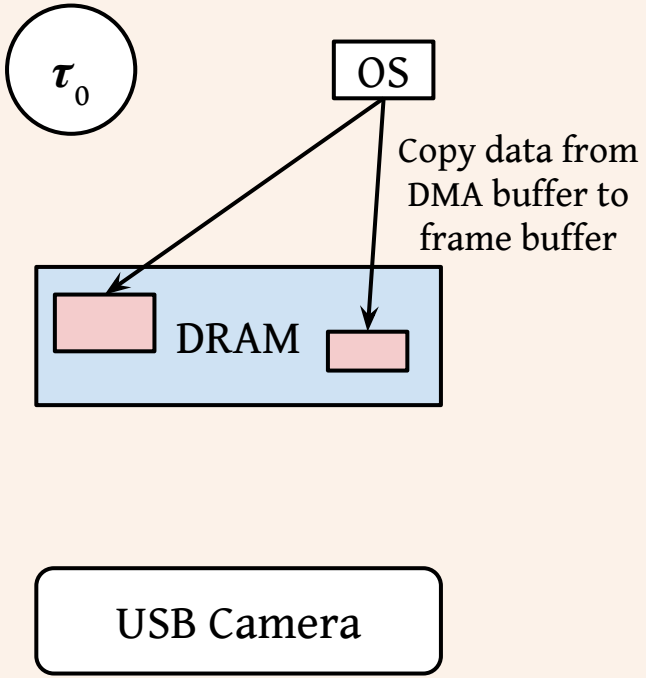


Example: Interference Sources from USB I/O

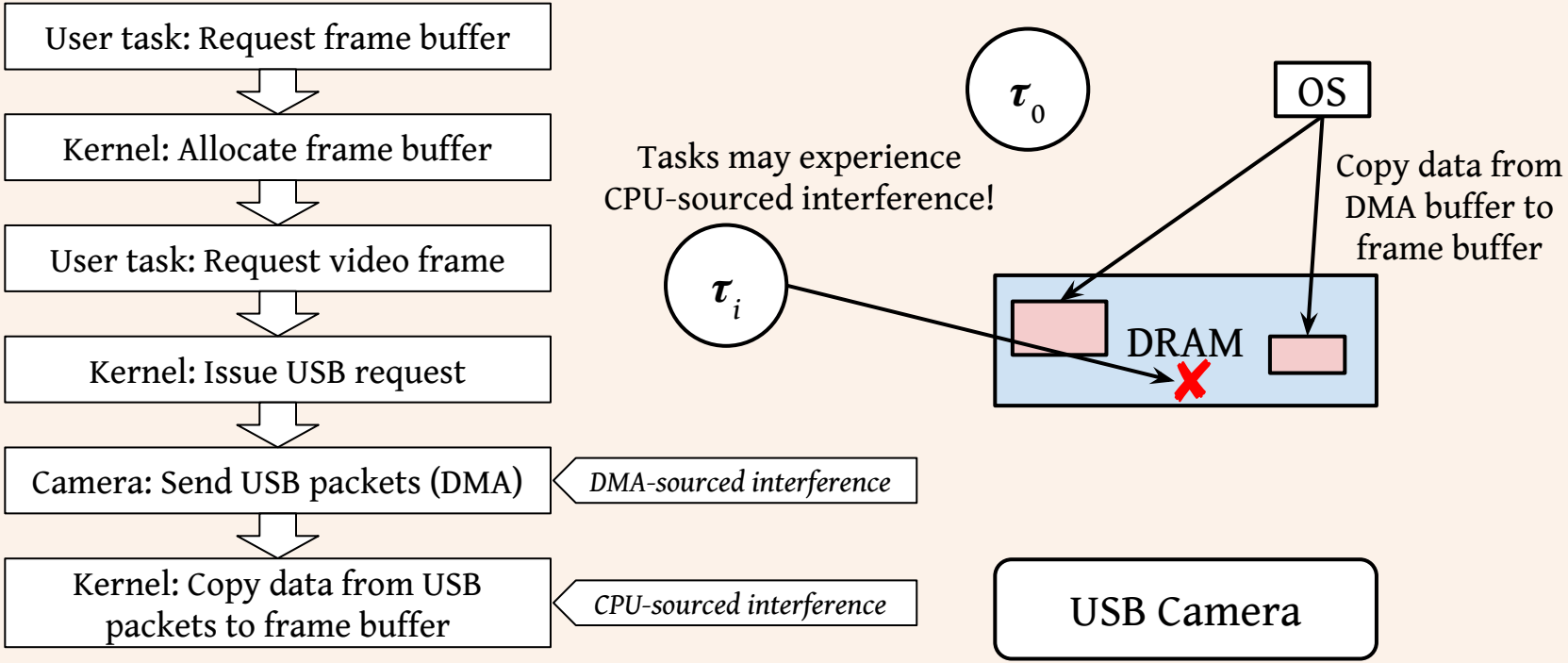


Example: Interference Sources from USB I/O

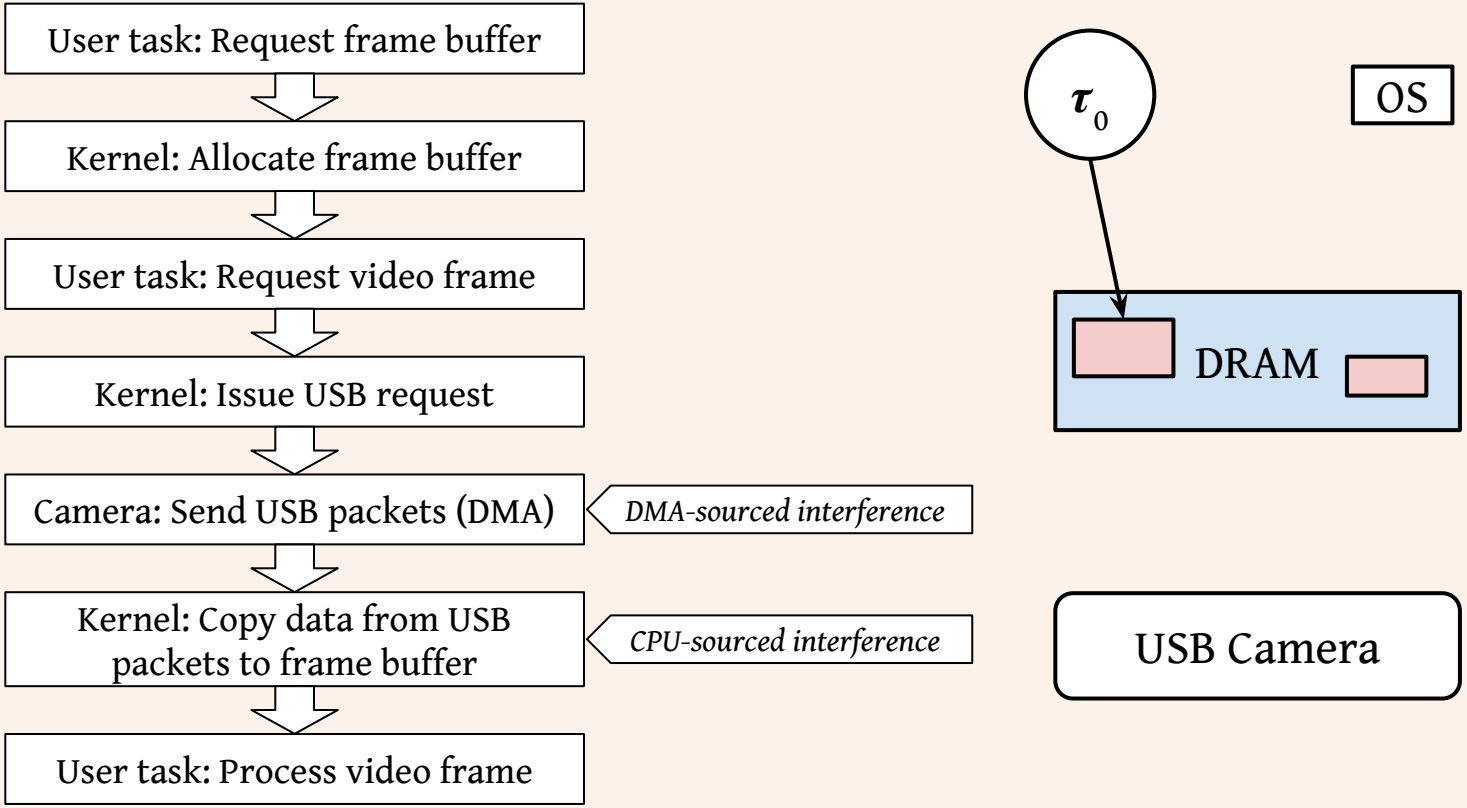




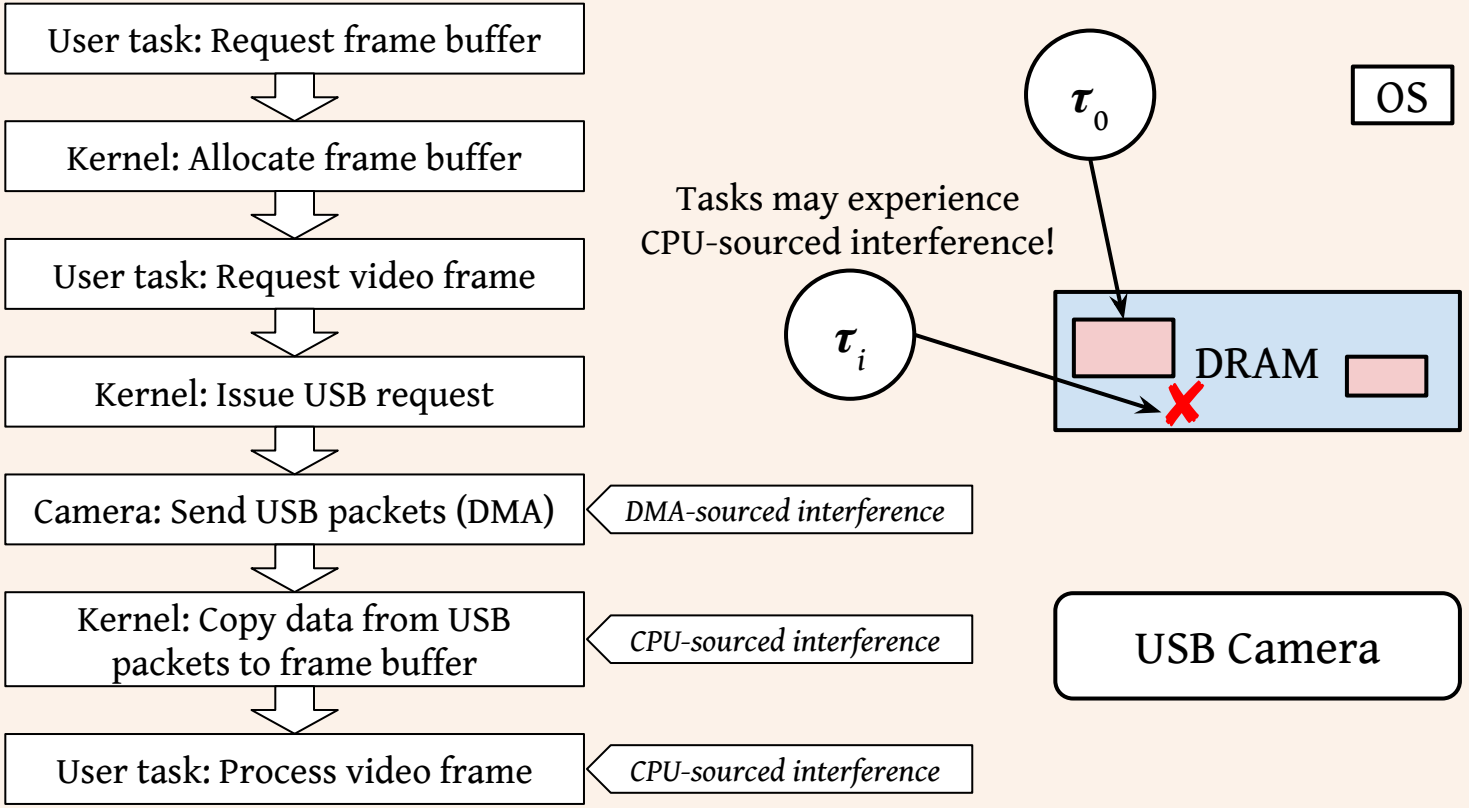
Example: Interference Sources from USB I/O



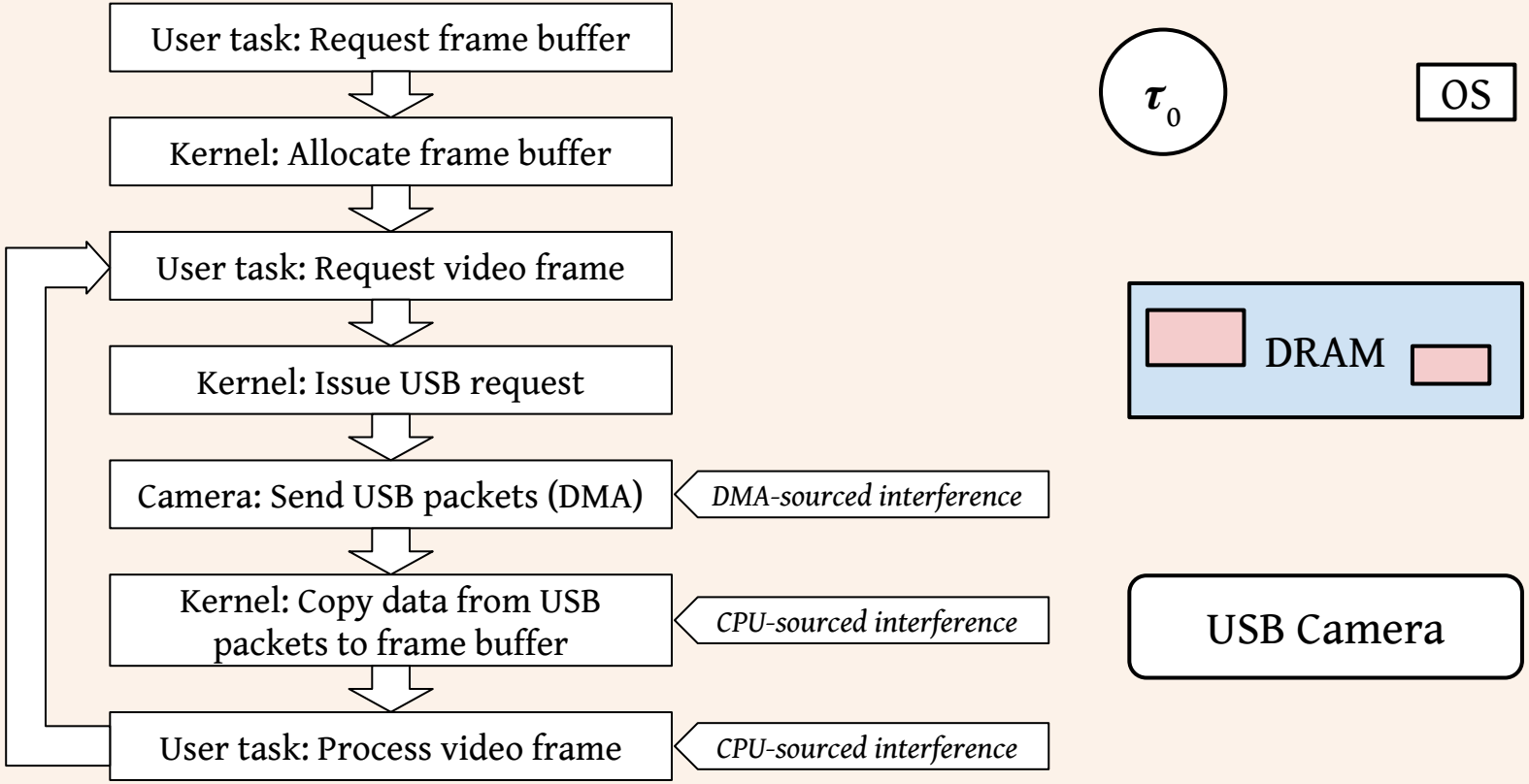
Example: Interference Sources from USB I/O



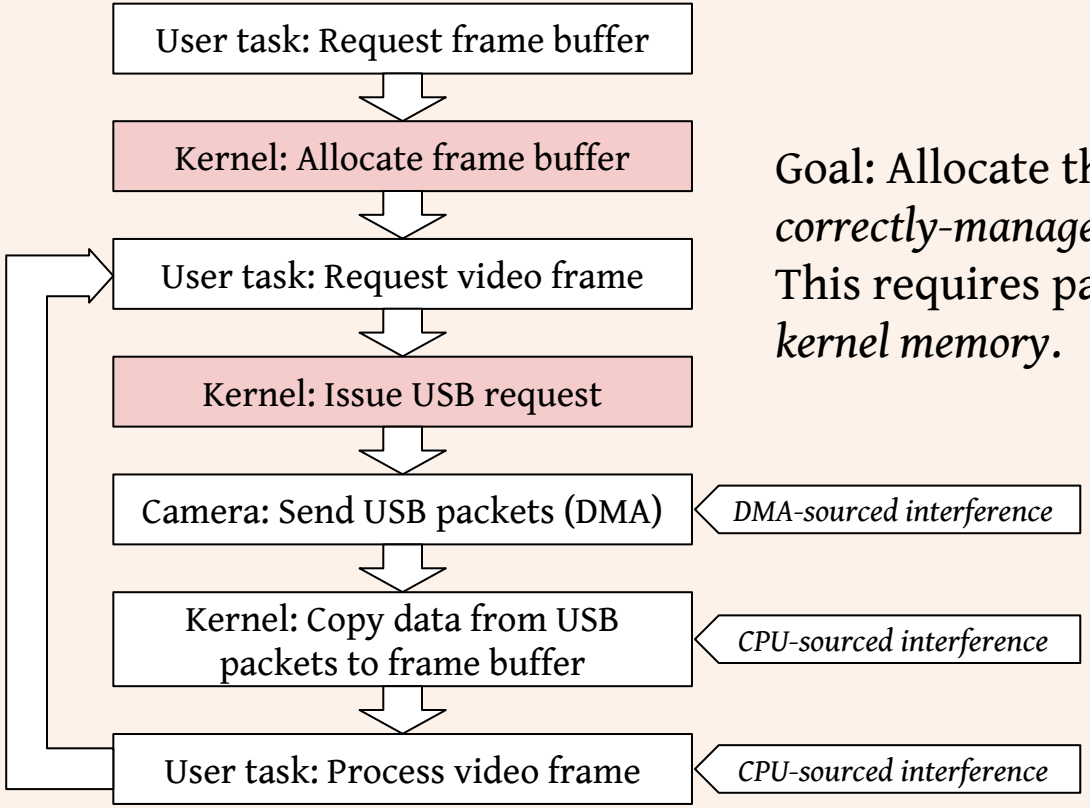
Example: Interference Sources from USB I/O



Example: Interference Sources from USB I/O

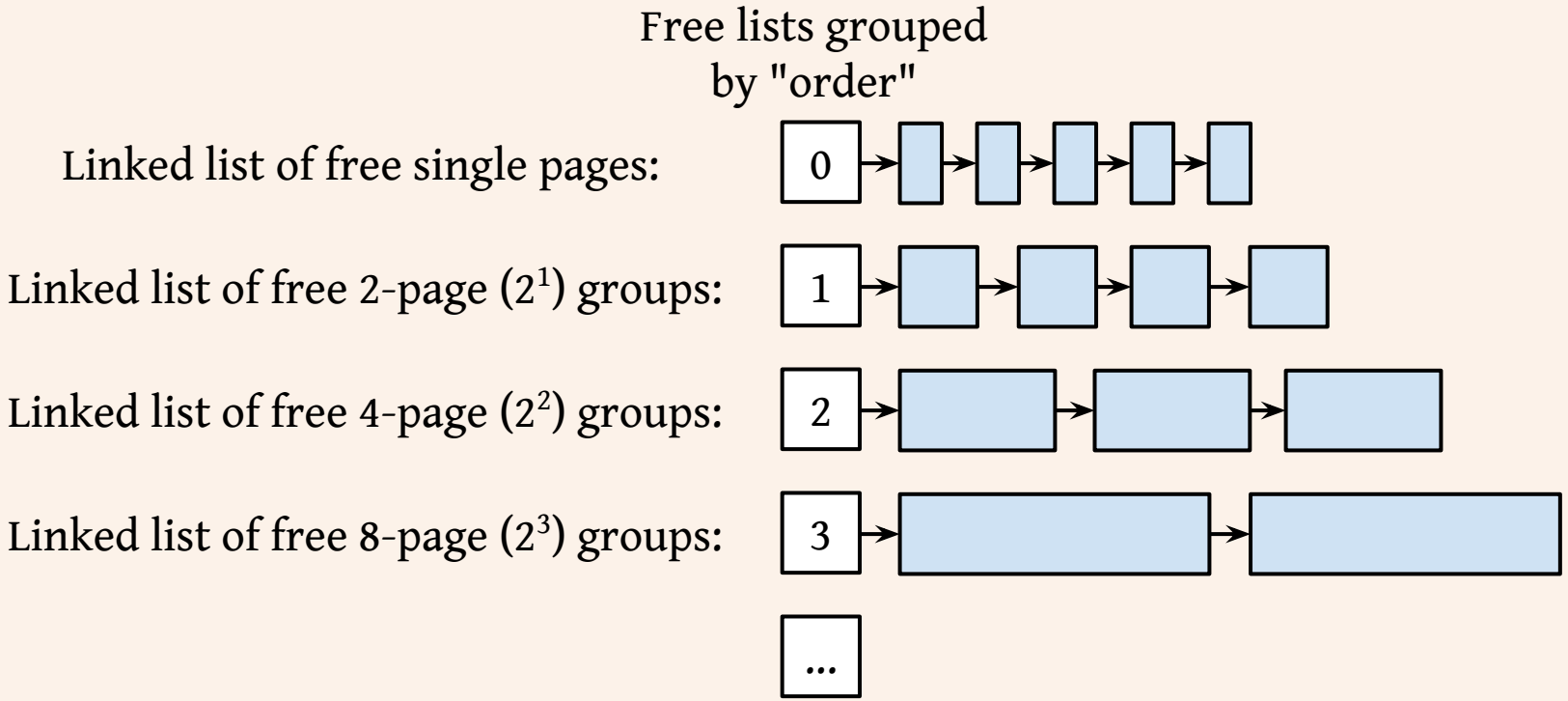


Example: Interference Sources from USB I/O

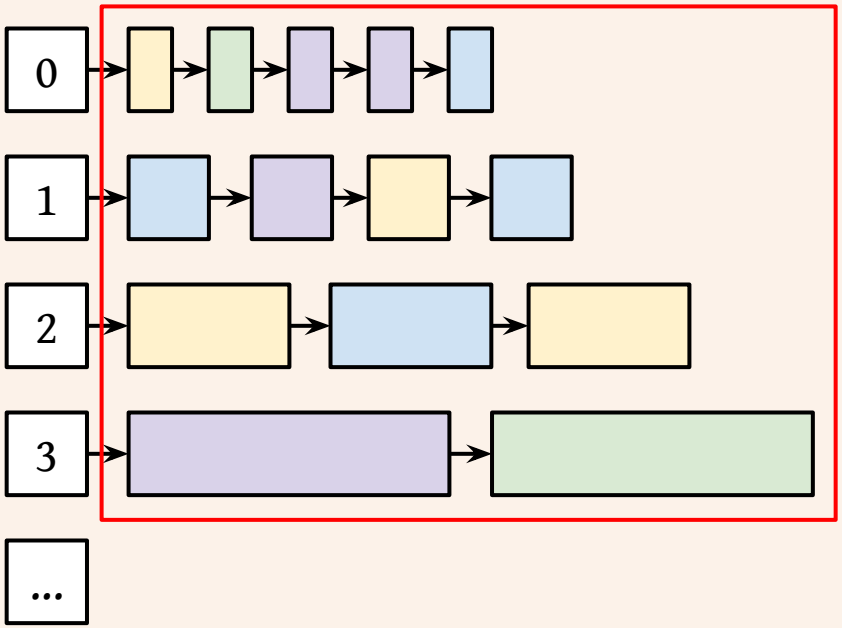


Goal: Allocate the frame and DMA buffers in *correctly-managed* partitions.
This requires partitioning *dynamically-allocated kernel memory*.

Linux's Buddy Allocator

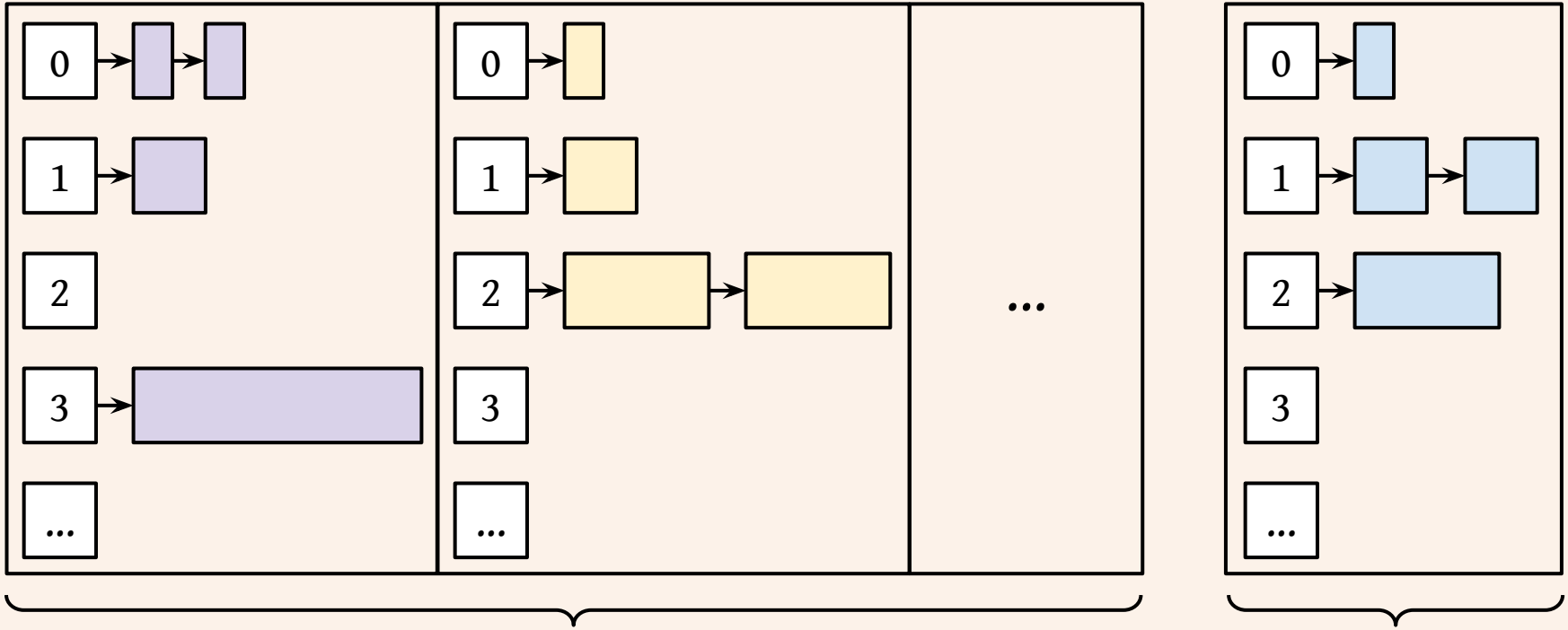


Linux's Buddy Allocator



Initially populated after boot with pages from all partitions!

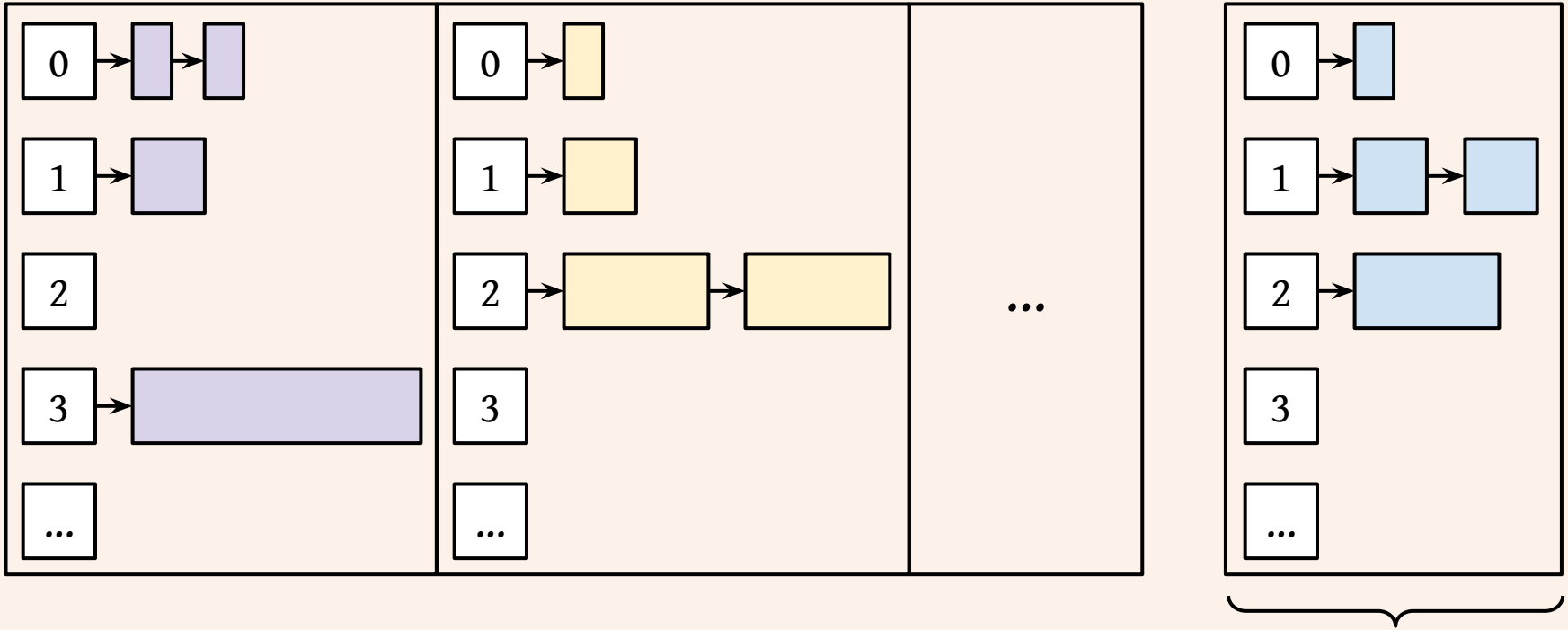
MC^2's Modified Buddy Allocator



m free lists: 1 for each CPU's high-criticality partition

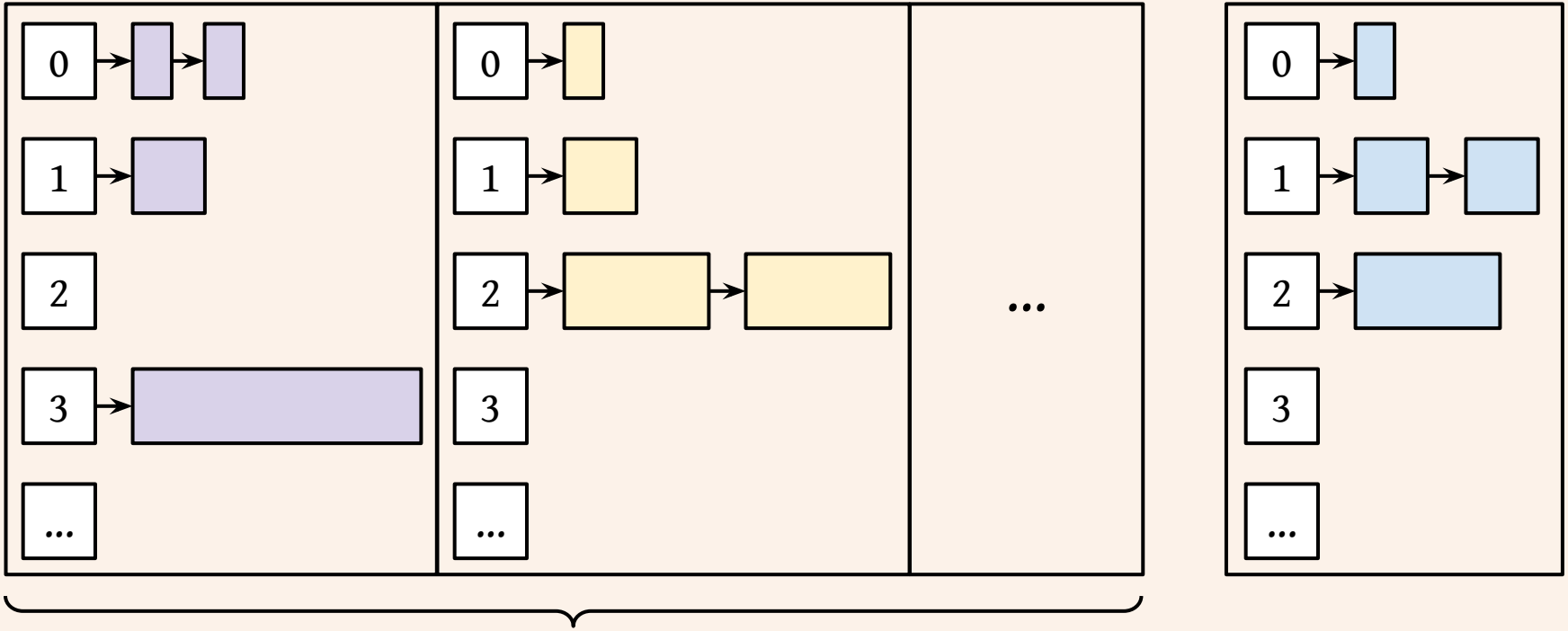
1 free list for the shared low-criticality partition

MC^2's Modified Buddy Allocator



Default allocator; will not interfere with high-criticality tasks

MC^2's Modified Buddy Allocator



Accessed by passing additional flags to the `get_free_pages(...)` function.

Buffer Management Techniques

- Non-DMA shared buffer management
  - SBP: ***S***elective cache ***b***y***p***ass
  - CE: ***C***oncurrency ***e***limination
  - CL: ***C***ache ***l***ocking
- DMA buffer management
  - Allocate in high-criticality partitions
  - Allocate in shared low-criticality partition

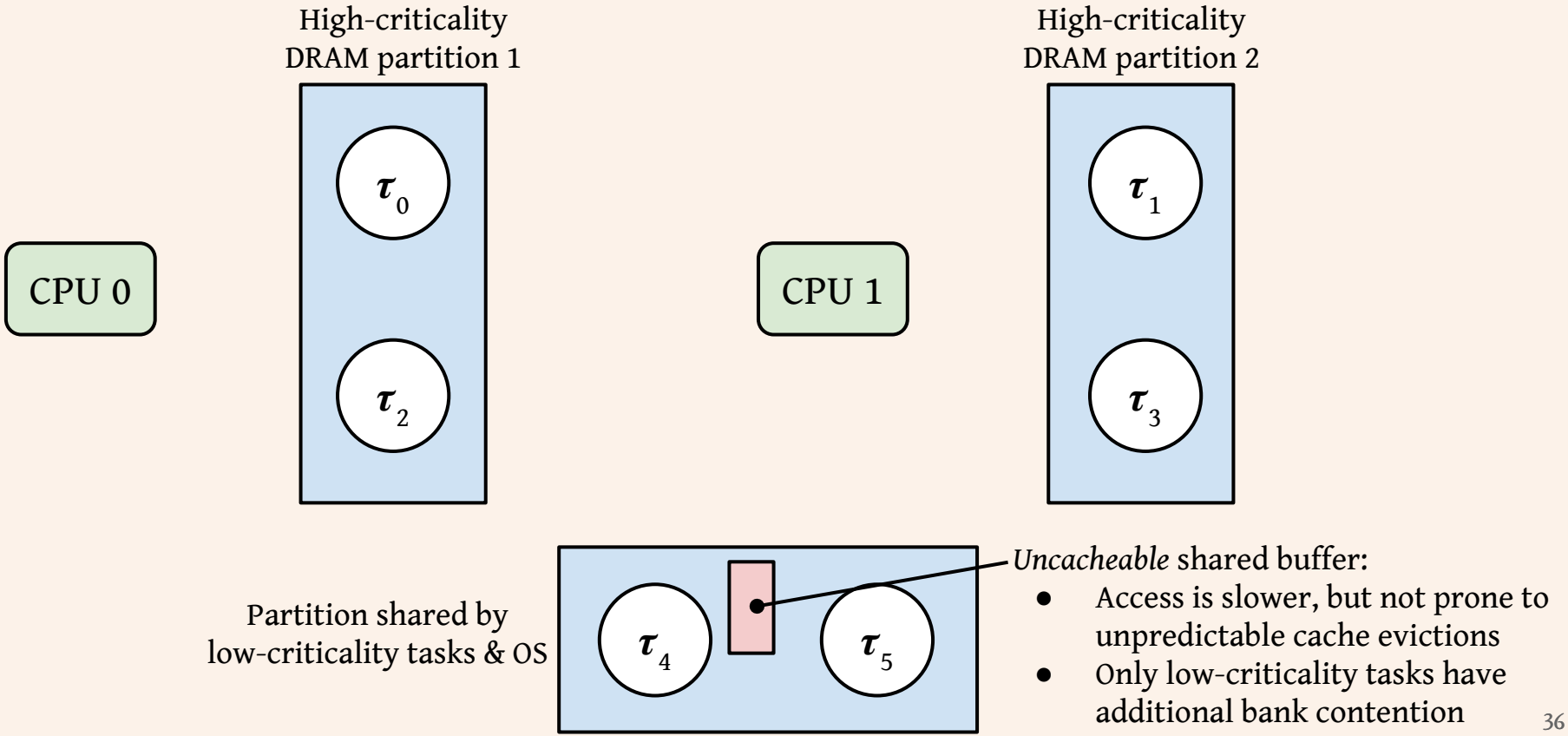
Buffer Management Techniques

- Non-DMA shared buffer management
  - SBP: ***S***elective cache ***b***y***p***ass
  - CE: ***C***oncurrency ***e***limination
  - CL: ***C***ache ***l***ocking
- DMA buffer management
  - Allocate in high-criticality partitions
  - Allocate in shared low-criticality partition

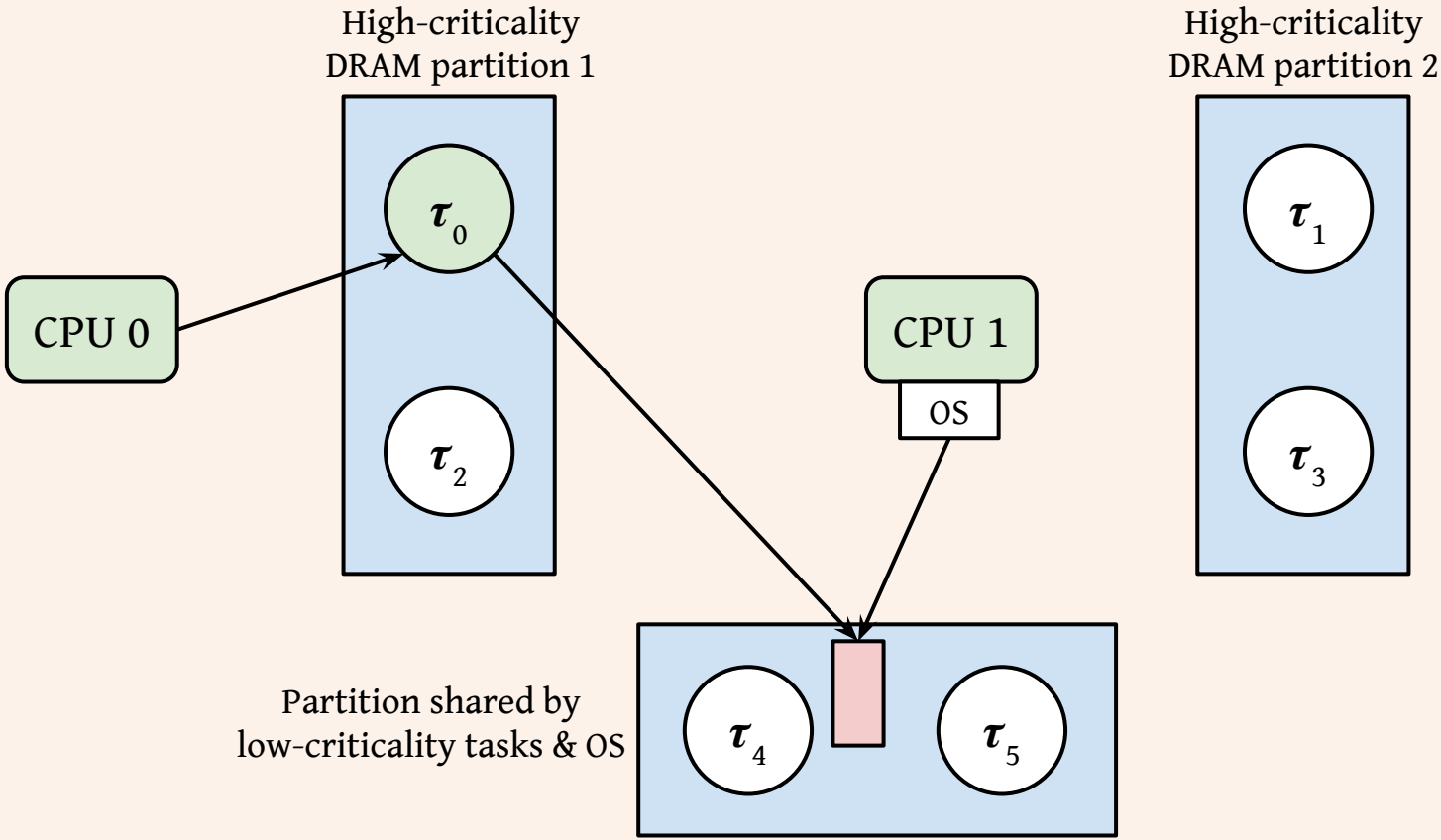
Buffer Management Techniques

- Non-DMA shared buffer management
  - SBP: ***S***elective cache ***b***y***p***ass
  - CE: ***C***oncurrency ***e***limination
  - CL: ***C***ache ***l***ocking
- DMA buffer management
  - Allocate in high-criticality partitions
  - Allocate in shared low-criticality partition

SBP: Selective LLC Bypass



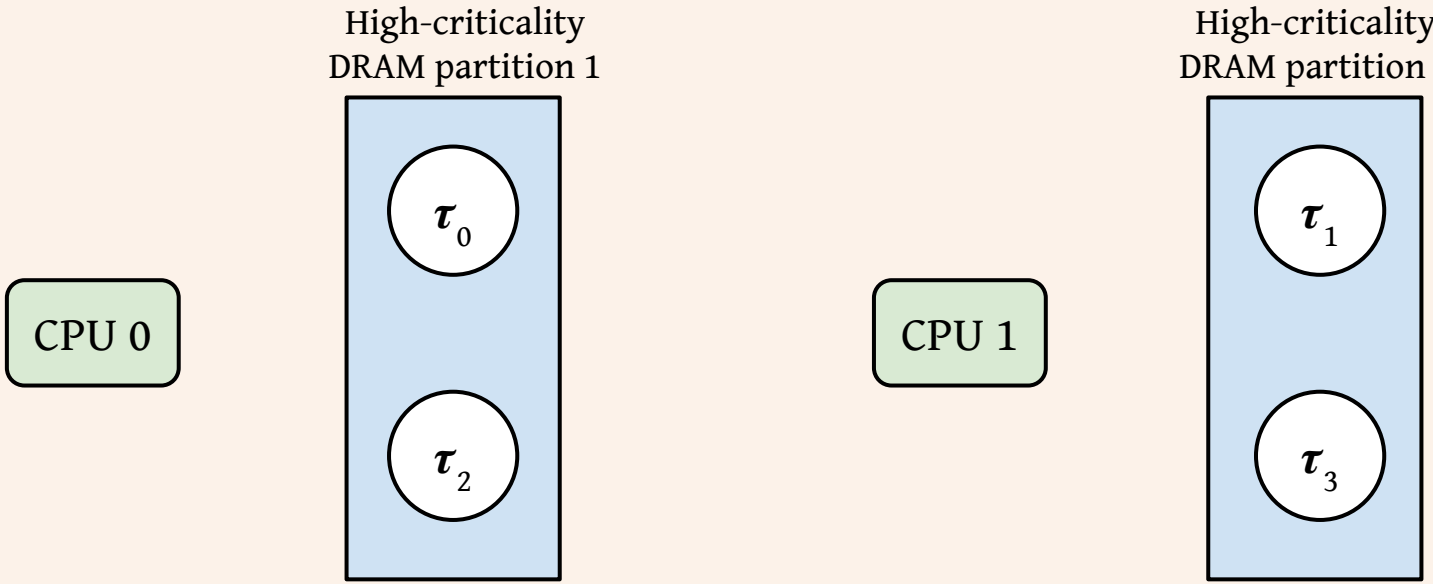
SBP: Selective LLC Bypass



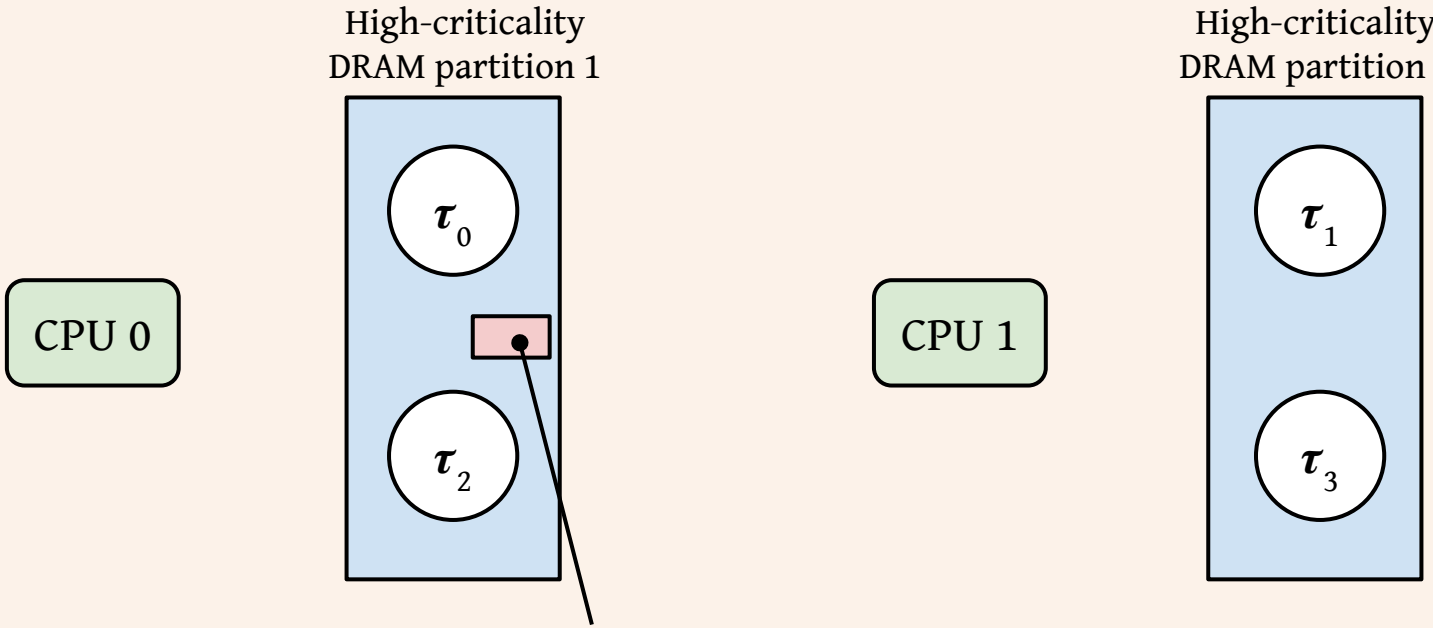
Buffer Management Techniques

- Non-DMA shared buffer management
  - SBP: ***S***elective cache ***b***y***p***ass
  - CE: ***C***oncurrency ***e***limination
  - CL: ***C***ache ***l***ocking
- DMA buffer management
  - Allocate in high-criticality partitions
  - Allocate in shared low-criticality partition

CE: Concurrency Elimination



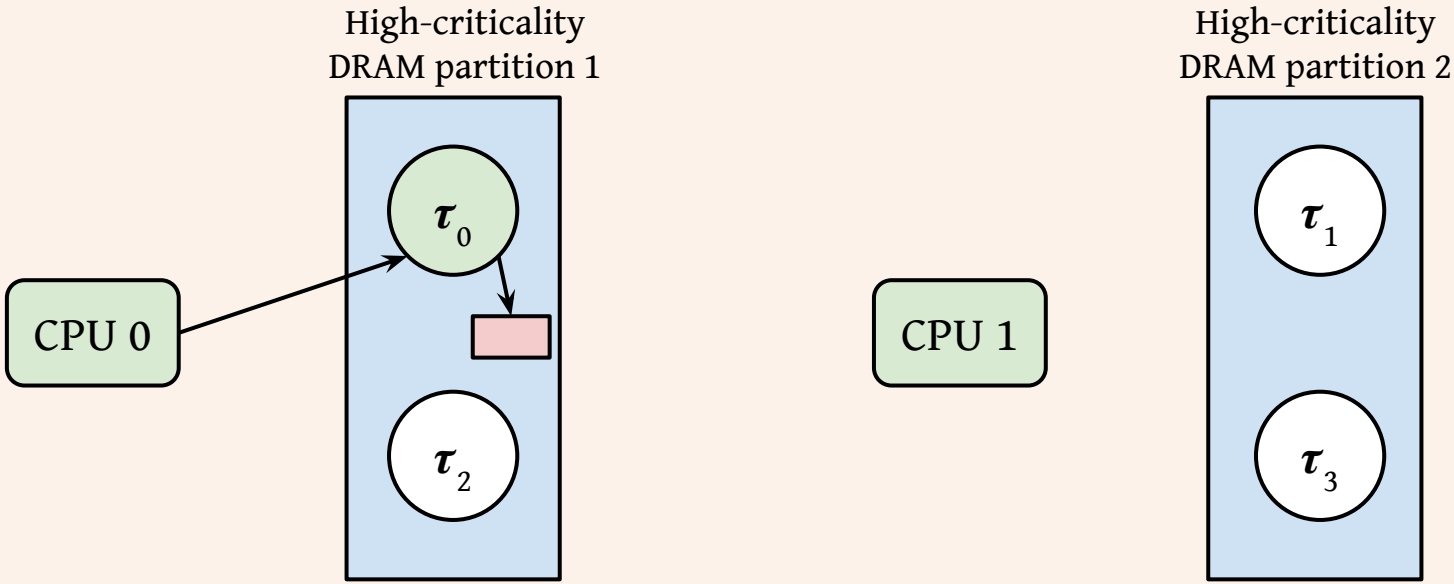
CE: Concurrency Elimination



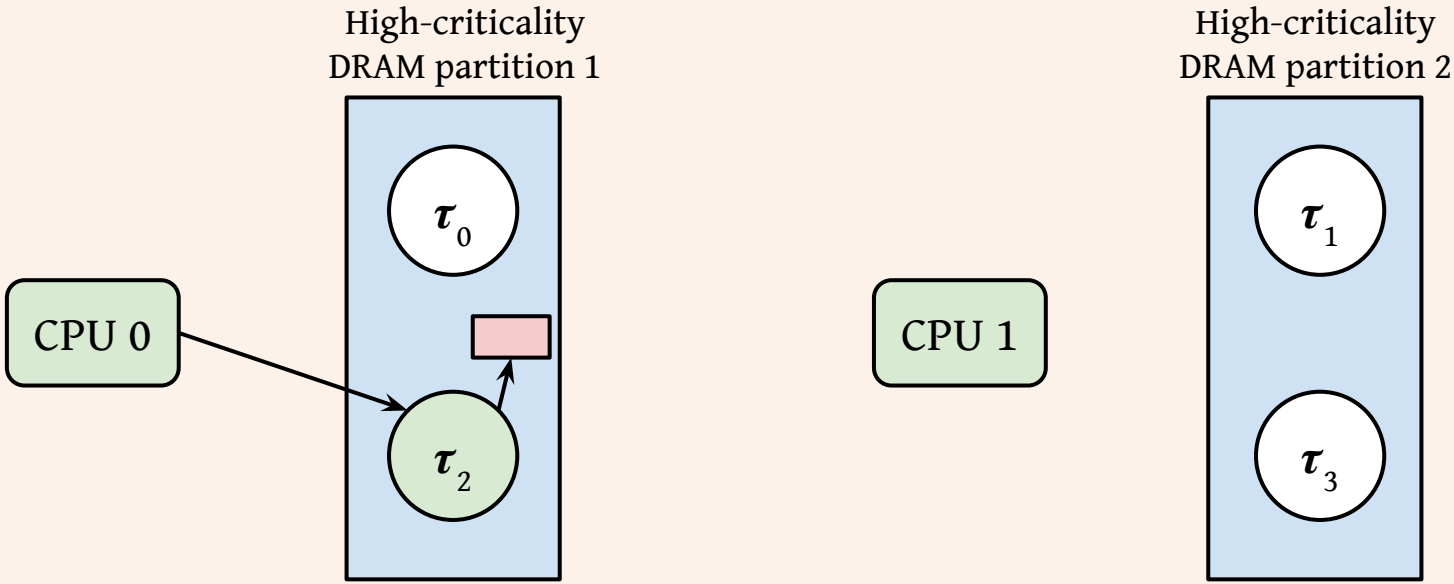
Buffer must only be used by CPU 0

- Will never be concurrently accessed
- No unpredictable cache evictions
- No unpredictable DRAM contention

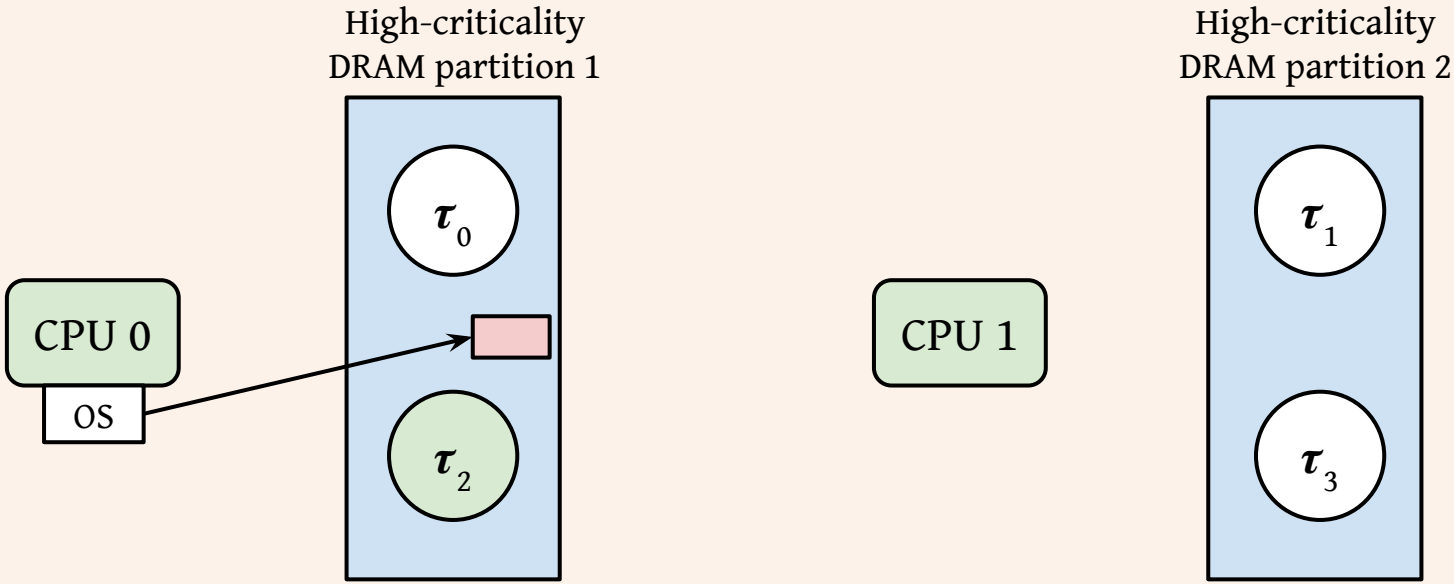
CE: Concurrency Elimination



CE: Concurrency Elimination



CE: Concurrency Elimination

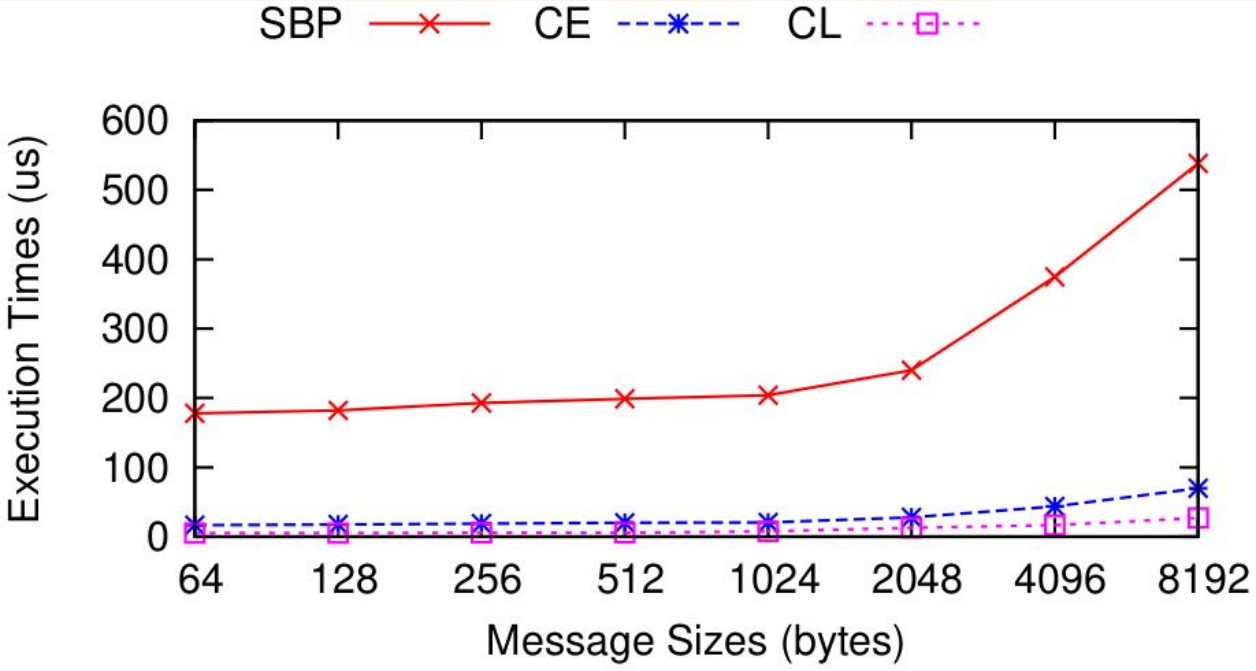


Buffer Management Techniques

- Non-DMA shared buffer management
  - SBP: ***S***elective cache ***b***y***p***ass
  - CE: ***C***oncurrency ***e***limination
  - CL: ***C***ache ***l***ocking
- DMA buffer management
  - Allocate in high-criticality partitions
  - Allocate in shared low-criticality partition

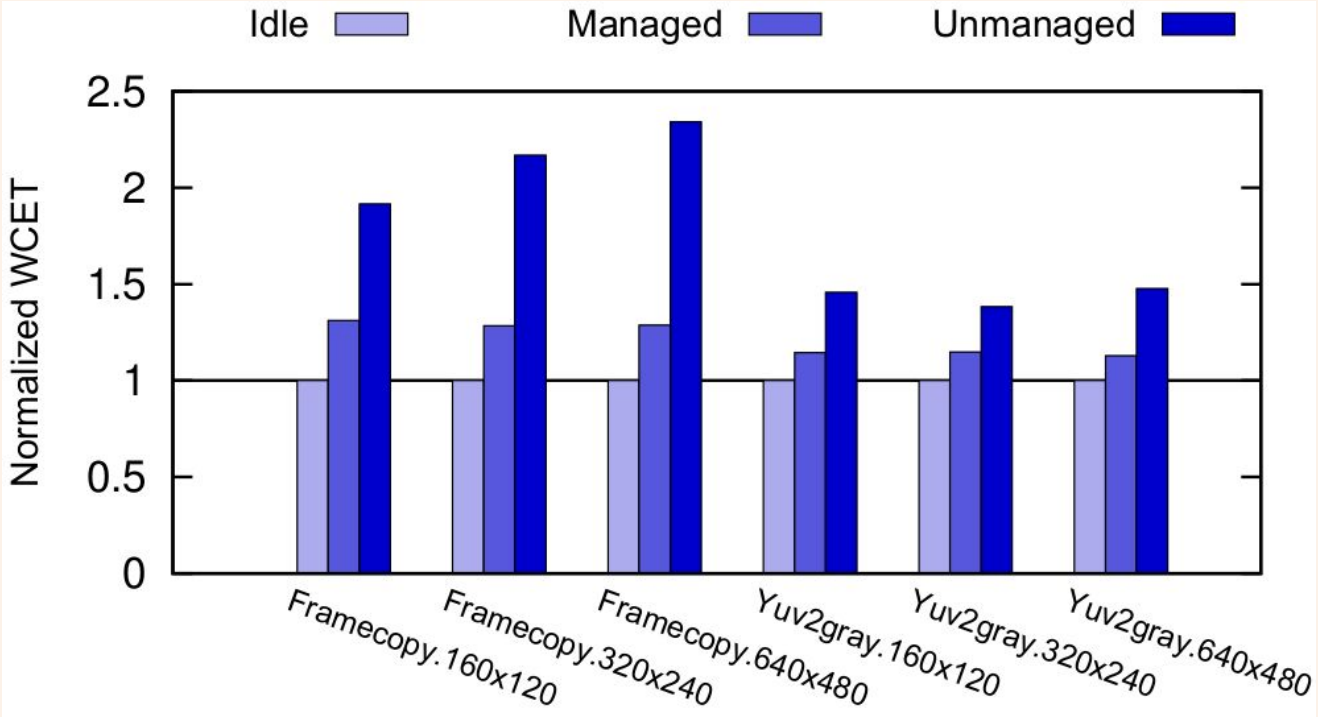
Buffer Management Techniques

Measured worst-case times of Linux's `load_msg` function (used for message-passing IPC):



Impact of CPU-Sourced Interference

Relative worst-case times of high-criticality tasks when accessing IPC buffers in the low-criticality partition (unmanaged) vs. their own partition (managed):



Buffer Management Techniques

- Non-DMA shared buffer management
  - SBP: ***S***elective cache ***b***y***p***ass
  - CE: ***C***oncurrency ***e***limination
  - CL: ***C***ache ***l***ocking
- DMA buffer management
  - Allocate in shared low-criticality partition
  - Allocate in high-criticality partitions

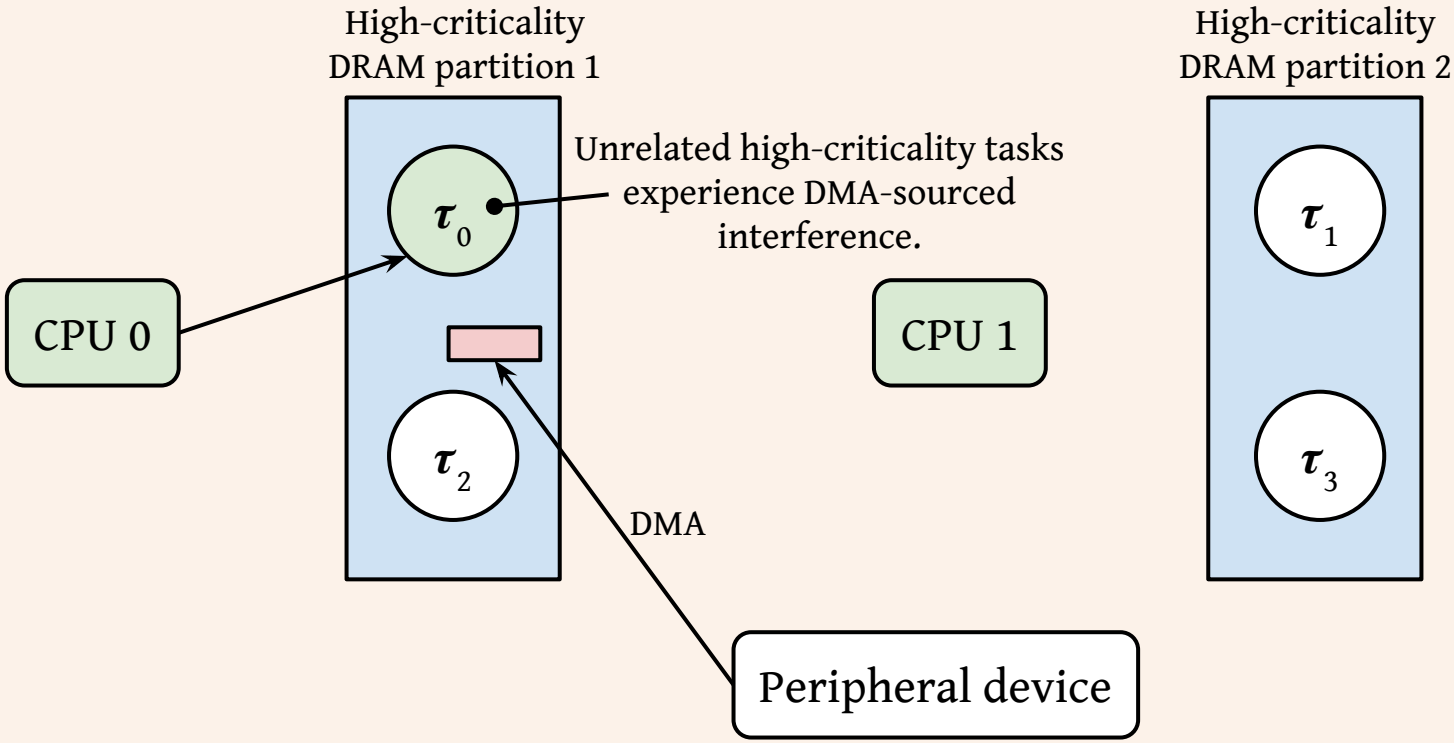
Buffer Management Techniques

- Non-DMA shared buffer management
  - SBP: ***S***elective cache ***b***y***p***ass
  - CE: ***C***oncurrency ***e***limination
  - CL: ***C***ache ***l***ocking
- DMA buffer management
  - Allocate in shared low-criticality partition
  - Allocate in high-criticality partitions

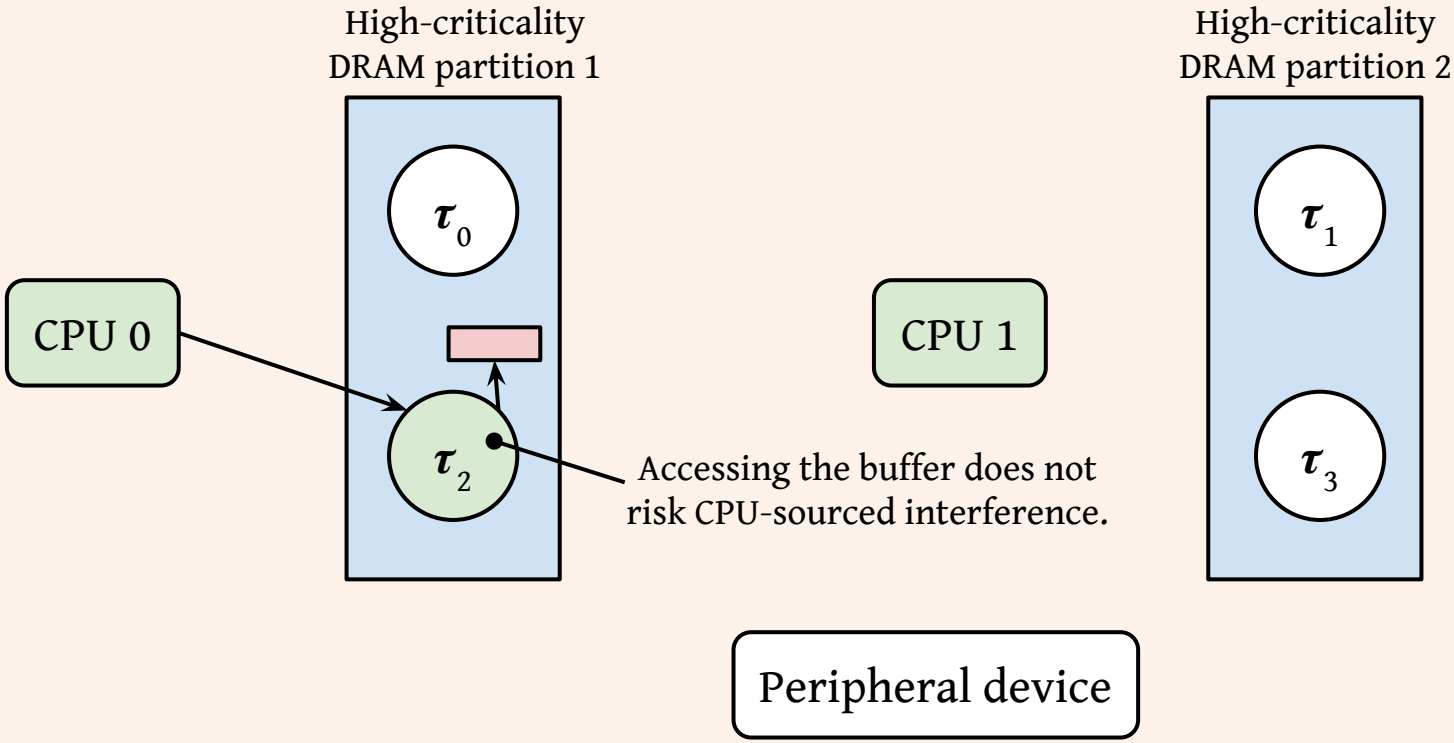
Buffer Management Techniques

- Non-DMA shared buffer management
  - SBP: ***S***elective cache ***b***y***p***ass
  - CE: ***C***oncurrency ***e***limination
  - CL: ***C***ache ***l***ocking
- DMA buffer management
  - Allocate in shared low-criticality partition
  - Allocate in high-criticality partitions

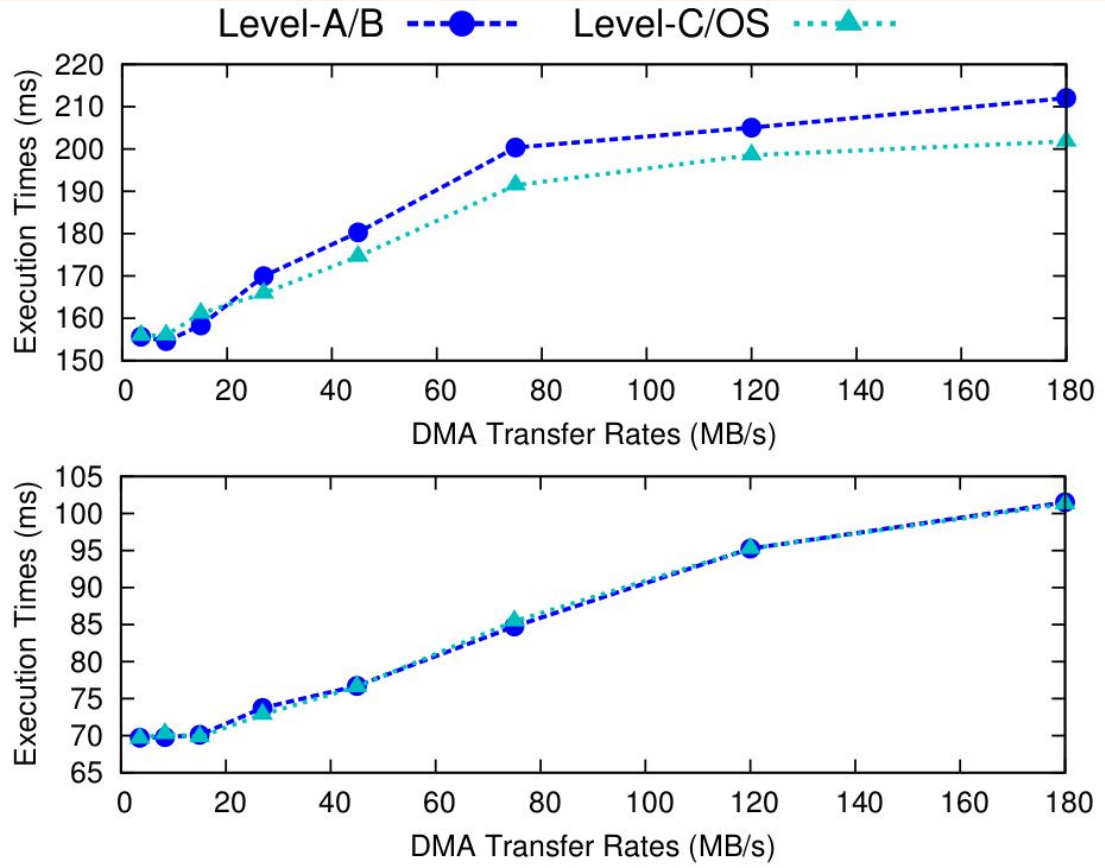
Placing DMA Buffers in High-Criticality Partitions



Placing DMA Buffers in High-Criticality Partitions



Impact on DMA-sourced Interference



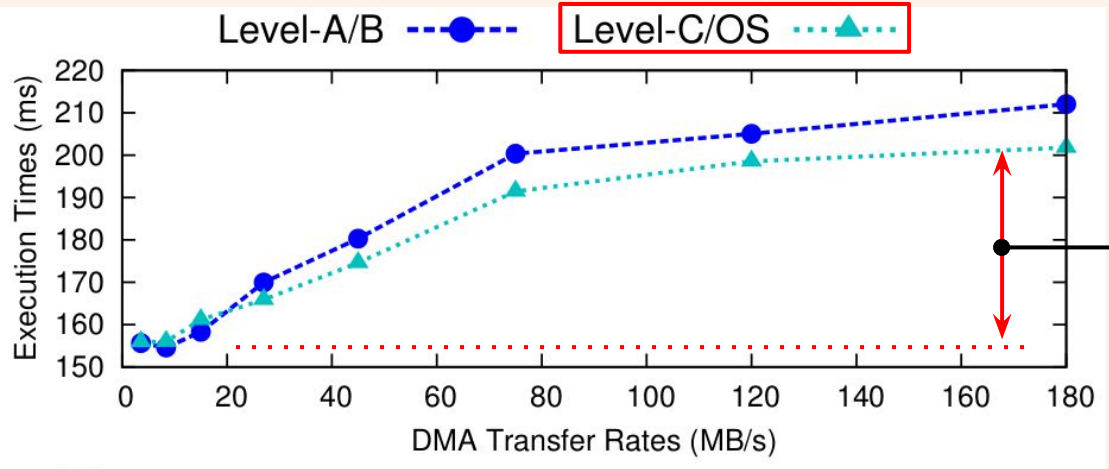
WCETs with a small LLC partition

(Execution time of a synthetic task during unrelated DMA.)

WCETs with a large LLC partition

Impact on DMA-sourced Interference

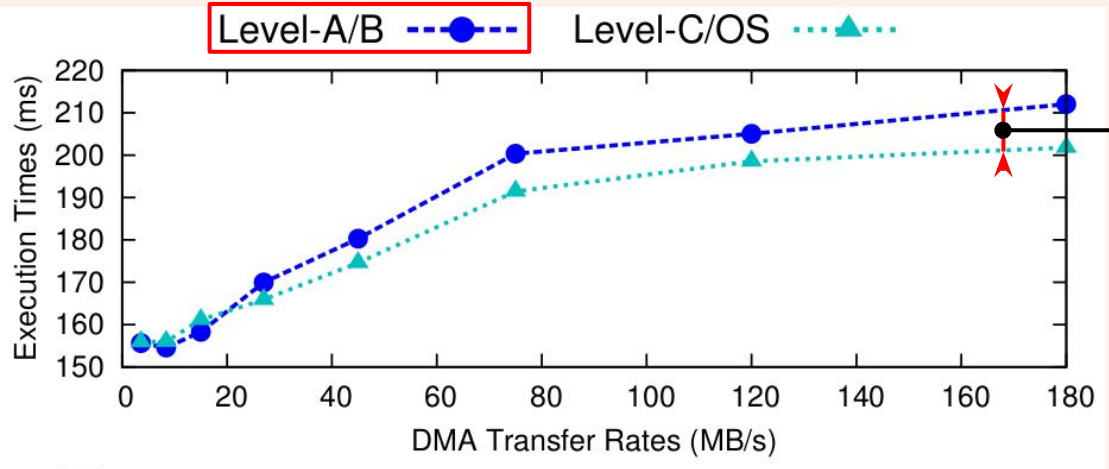
(The DMA buffer is in the shared low-criticality partition.)



Overhead due to bus contention.

Impact on DMA-sourced Interference

(The DMA buffer is in the same high-criticality partition as the measurement task.)

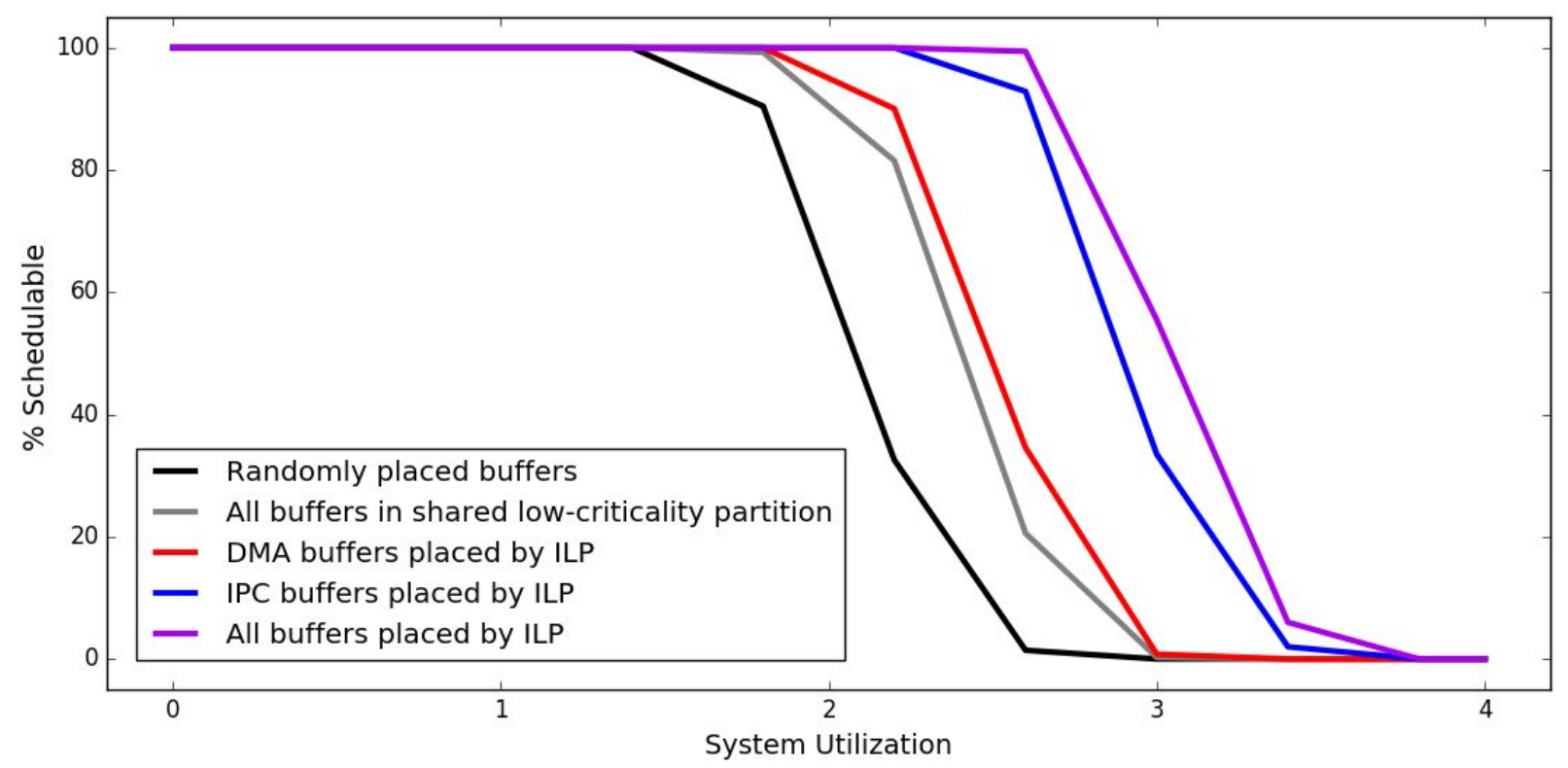


Overhead due to bank contention.
(preventable DMA-sourced interference)

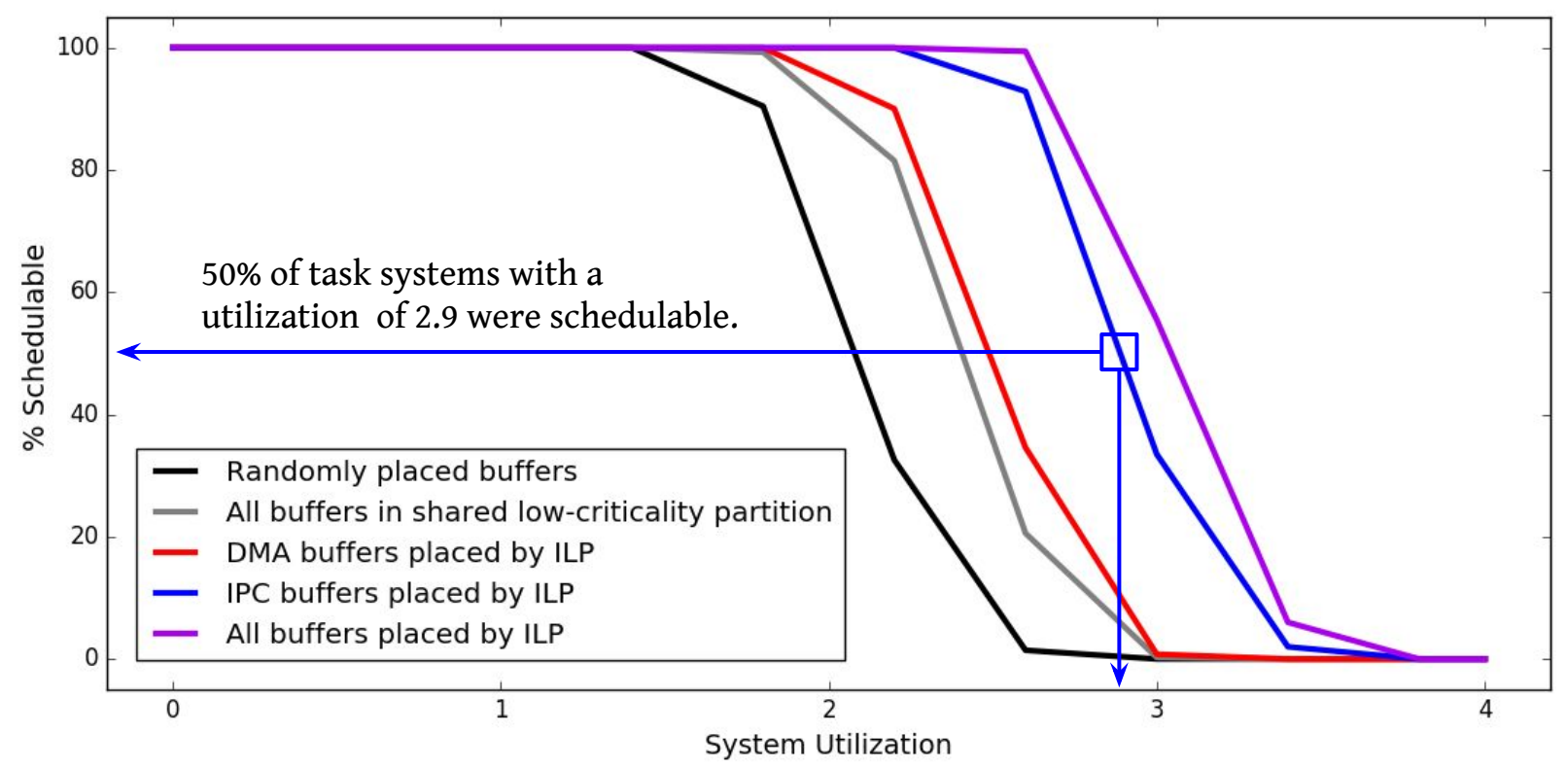
Schedulability Study

- Answer the following questions analytically:
  - How bad is unmanaged I/O and IPC?
  - Does the new default allocation scheme (shared partition) help?
  - How much do special management schemes for DMA and IPC buffers improve schedulability?

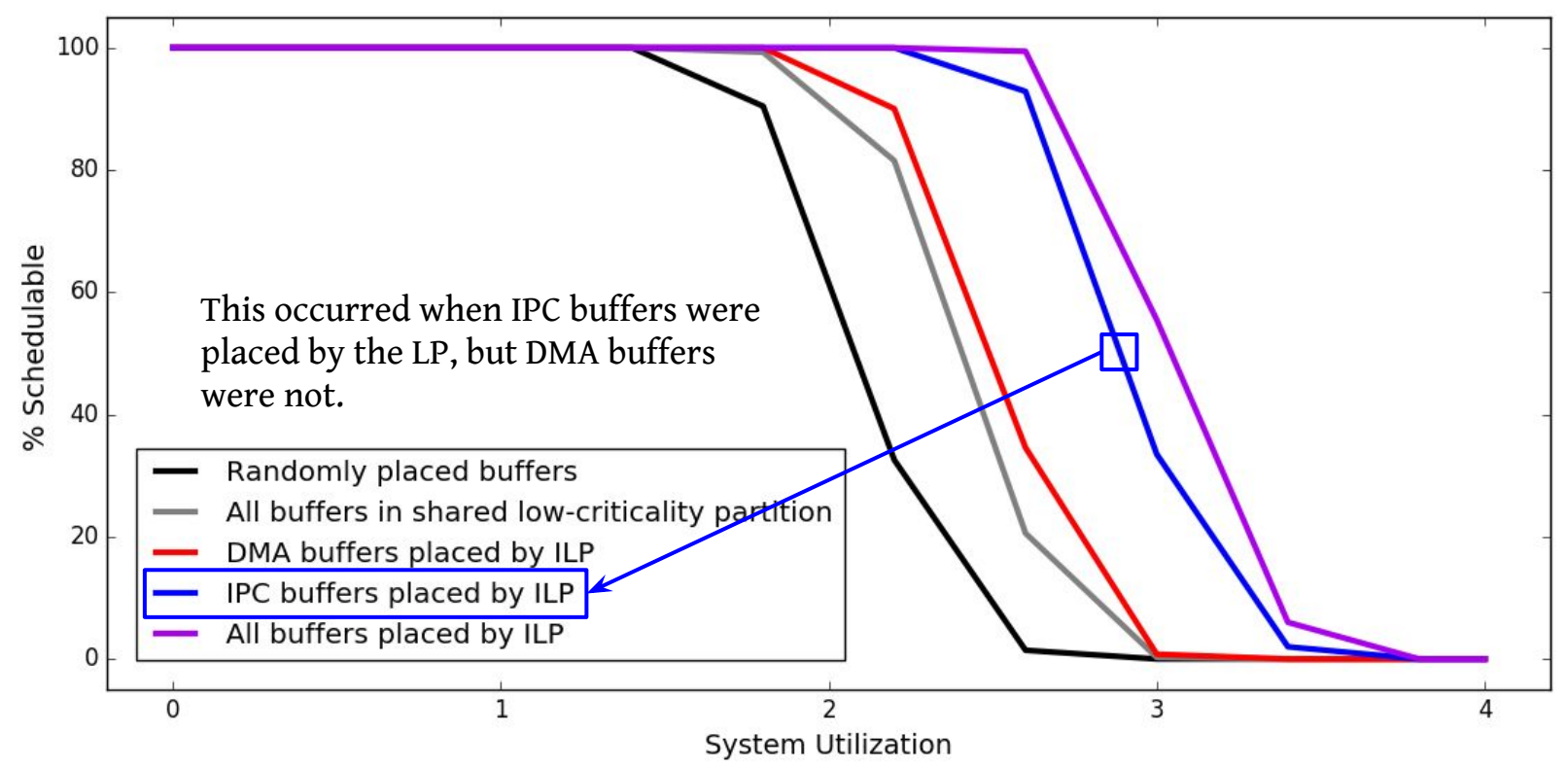
Impact on Schedulability



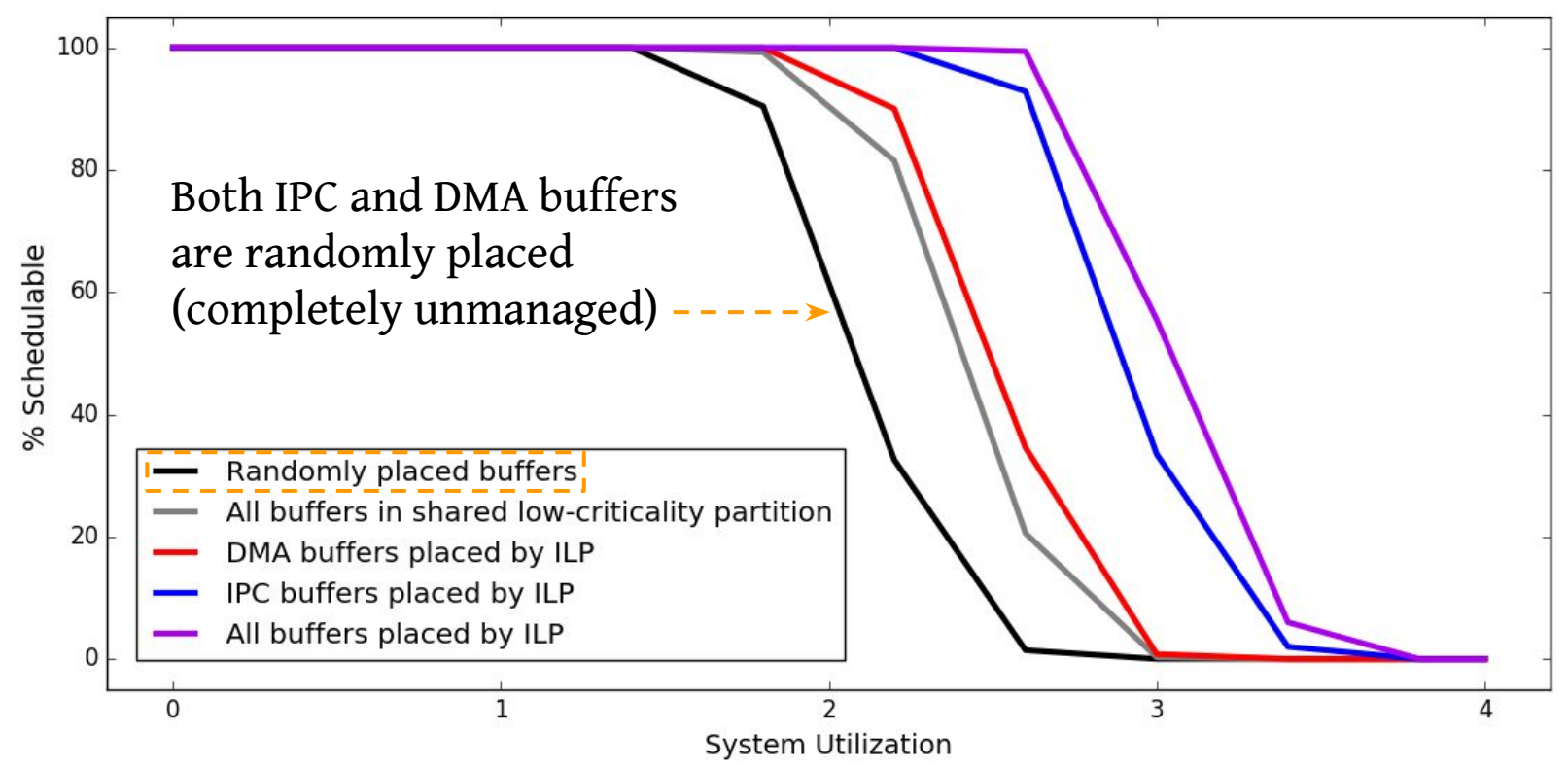
Impact on Schedulability



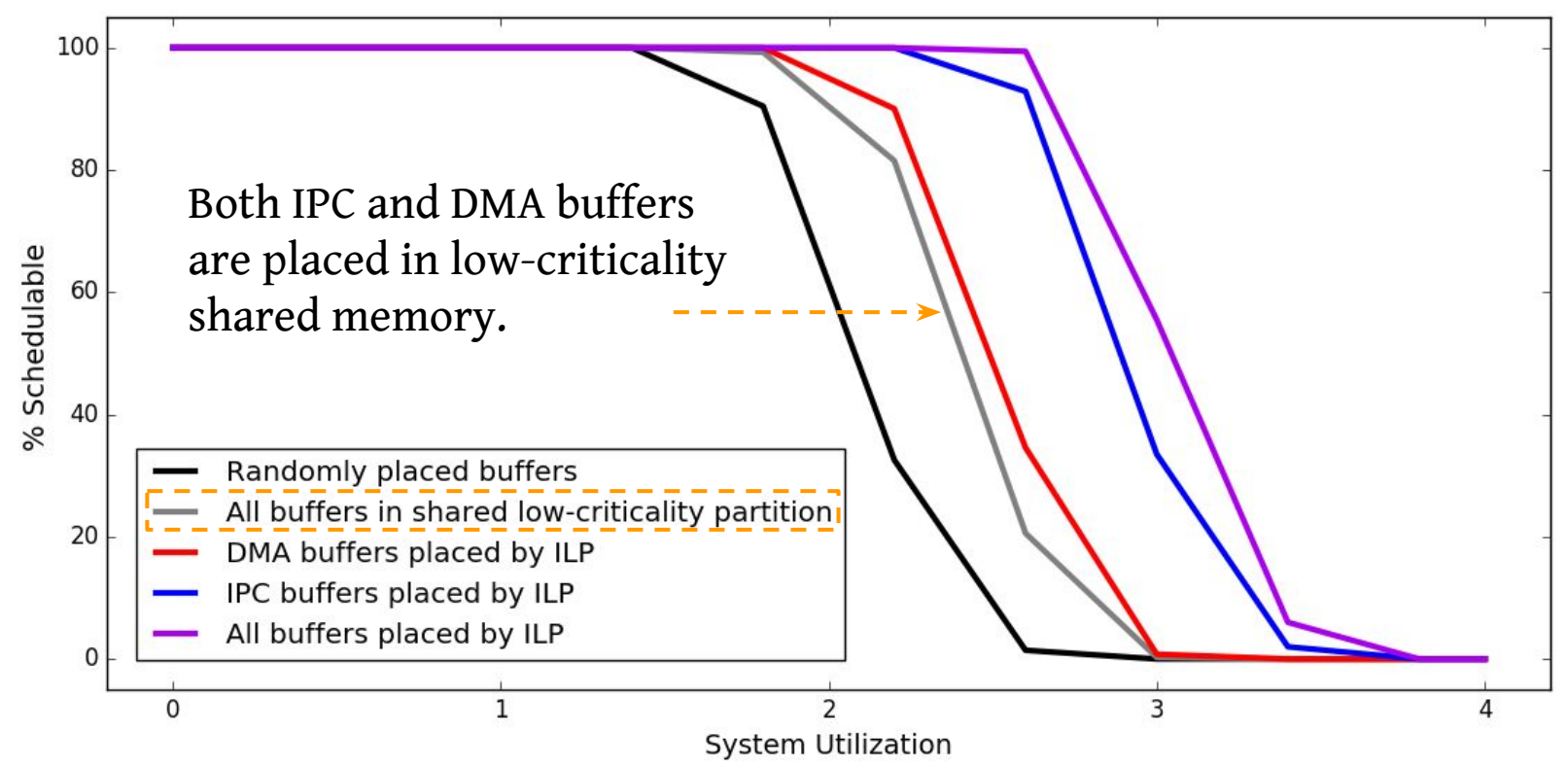
Impact on Schedulability



Impact on Schedulability

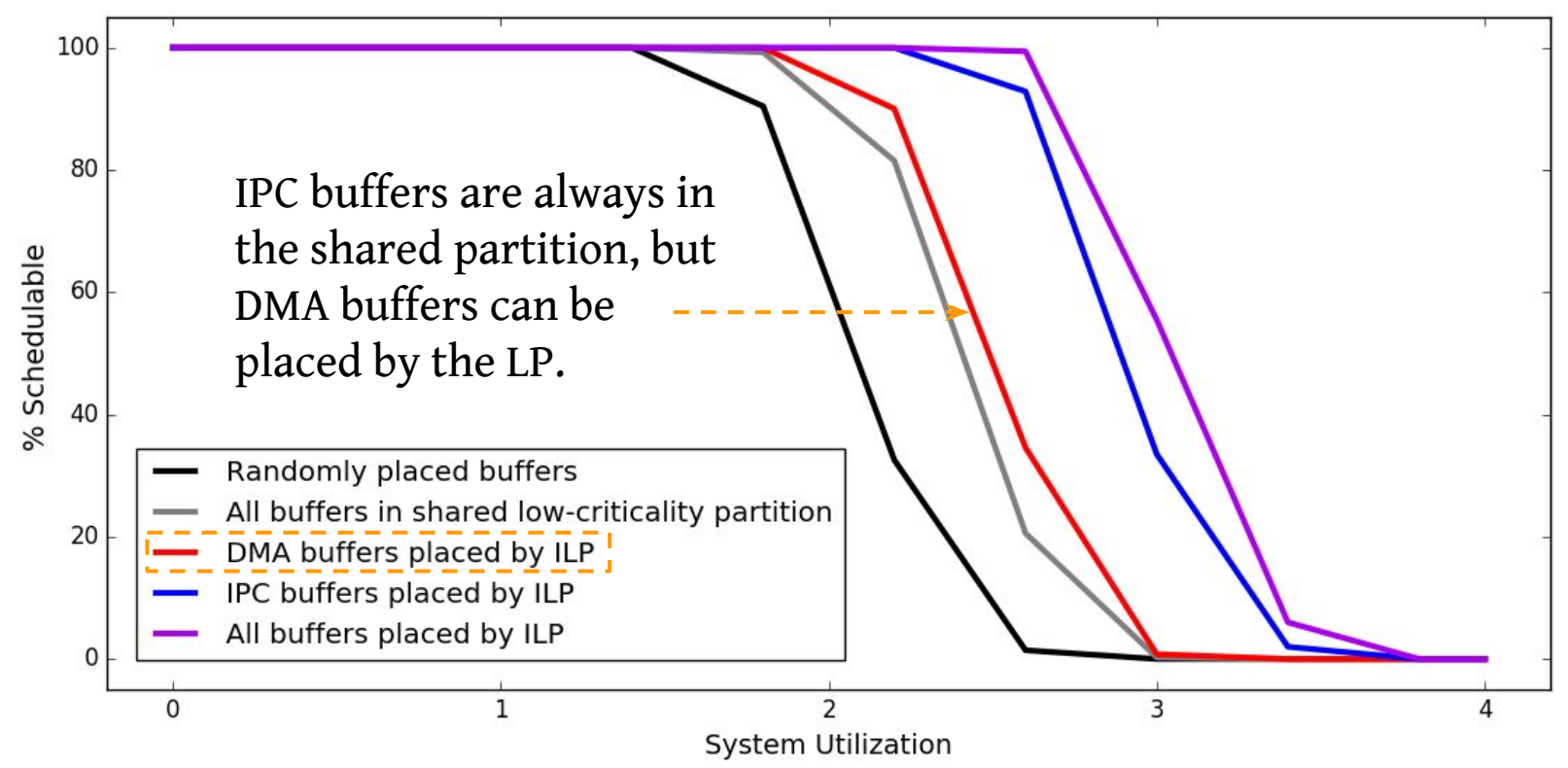


Impact on Schedulability



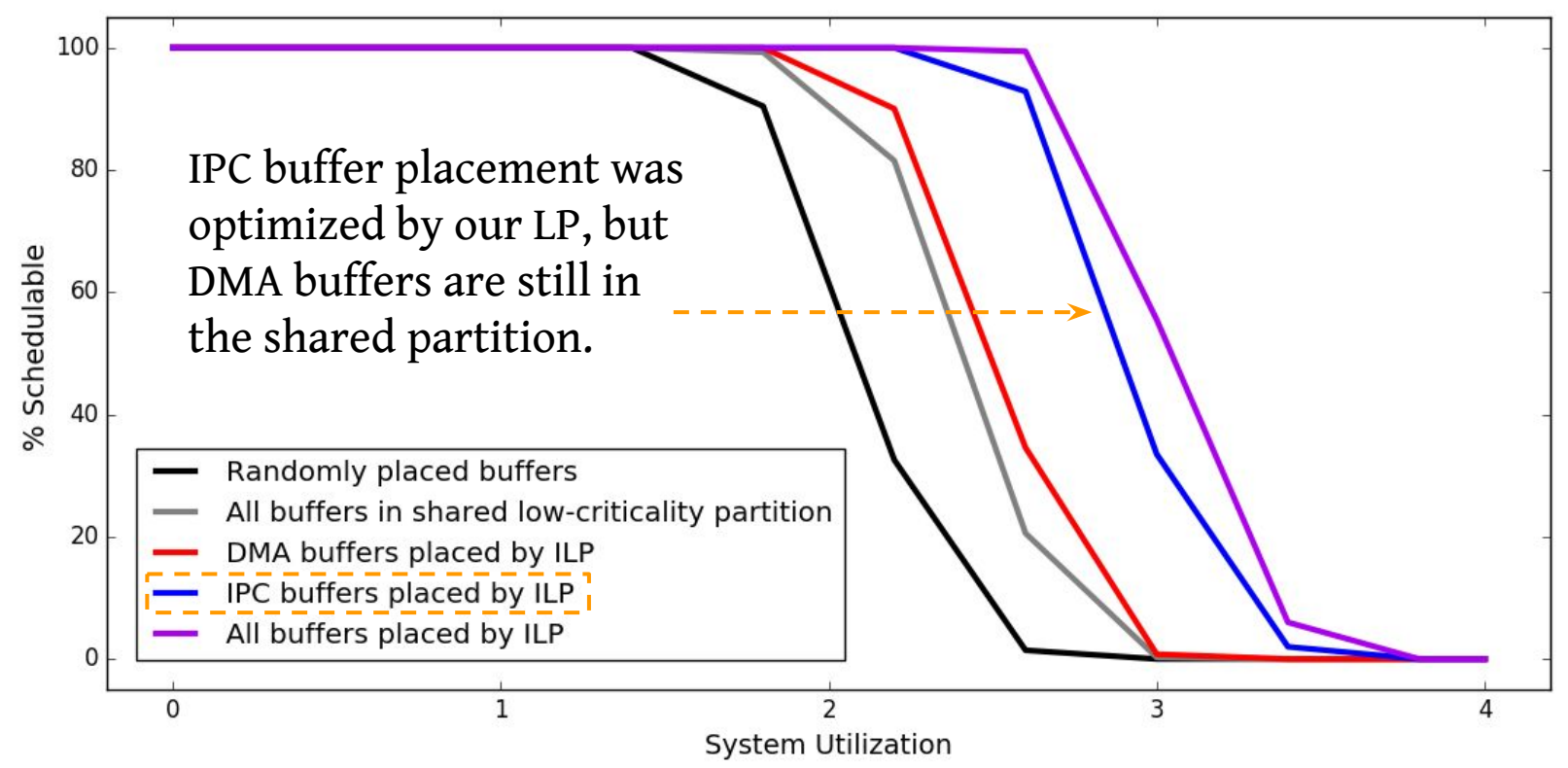
This approach was 6% better than random placement on average

Impact on Schedulability



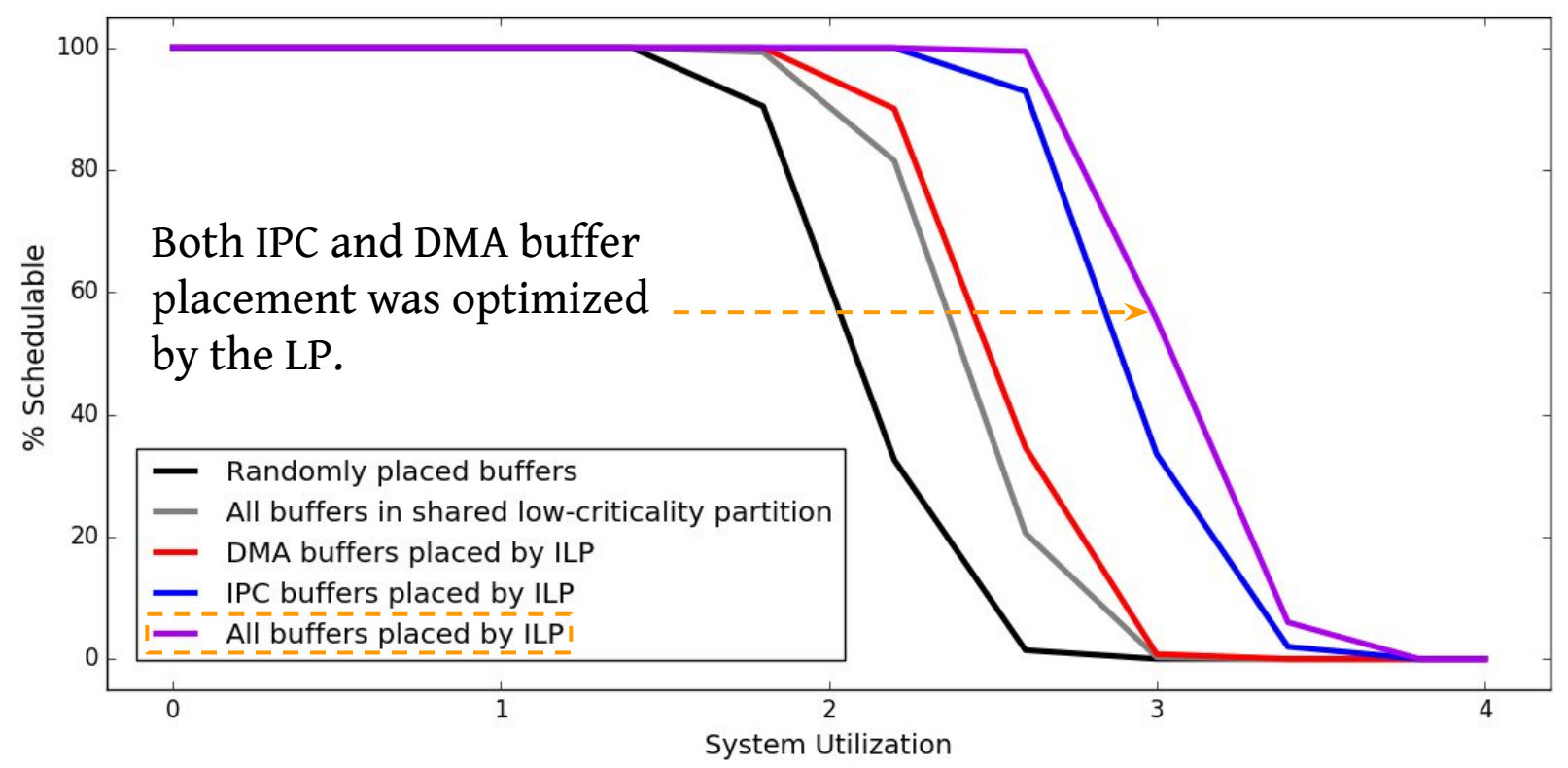
DMA buffer placement had a relatively small schedulability impact.

Impact on Schedulability



Enabling the *SBP*, *CE*, or *CL* resulted in large schedulability improvements.

Impact on Schedulability



Our additions handled DMA and OS-supported IPC 21% better than random placement.

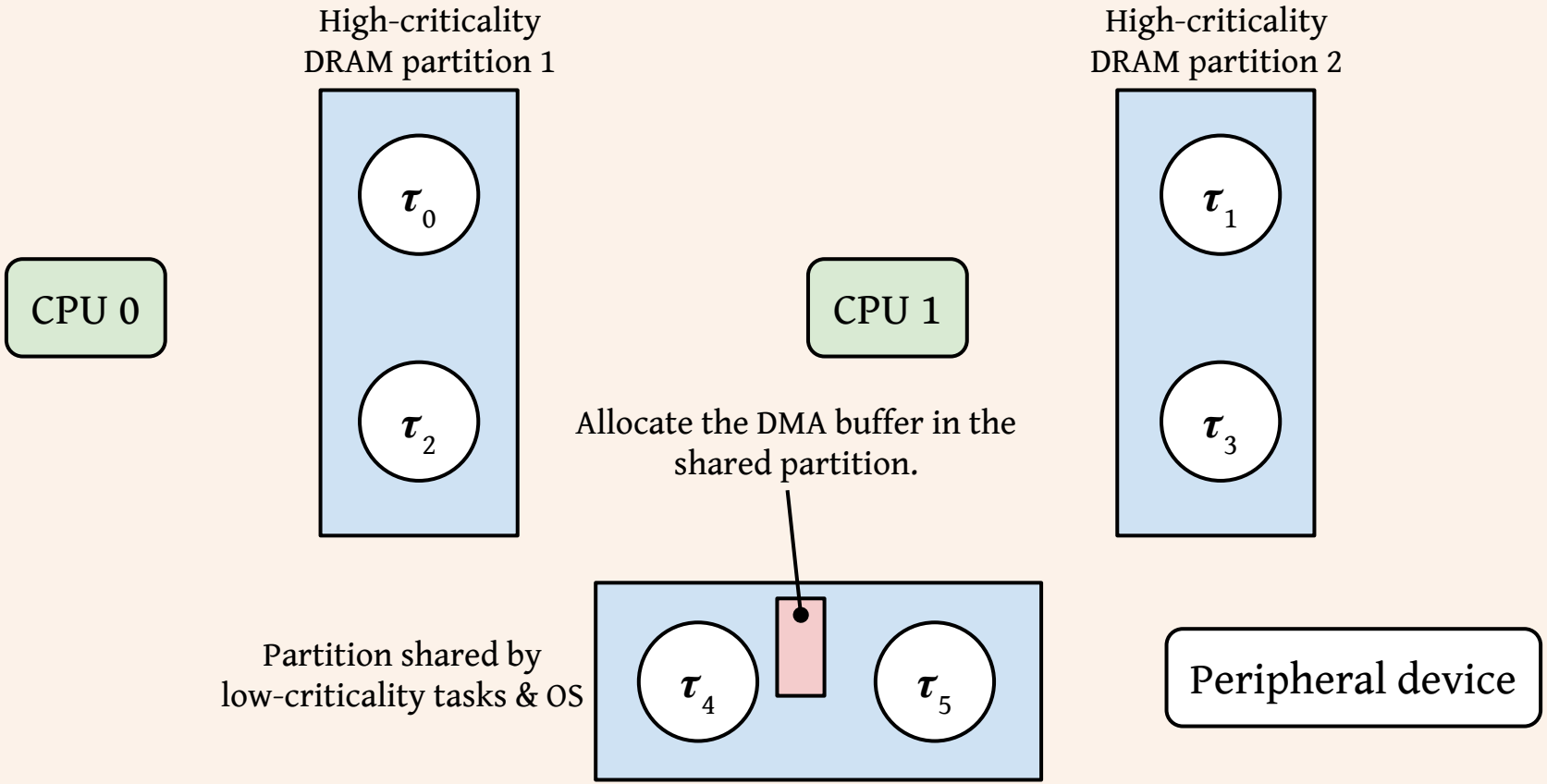
Conclusion

- OS and DMA-device activity cause shared-hardware interference.
- Isolating OS and DMA memory requires isolation-aware memory management in the kernel.
- Using a quad-core ARM model, our management techniques improve schedulability by over 20%.

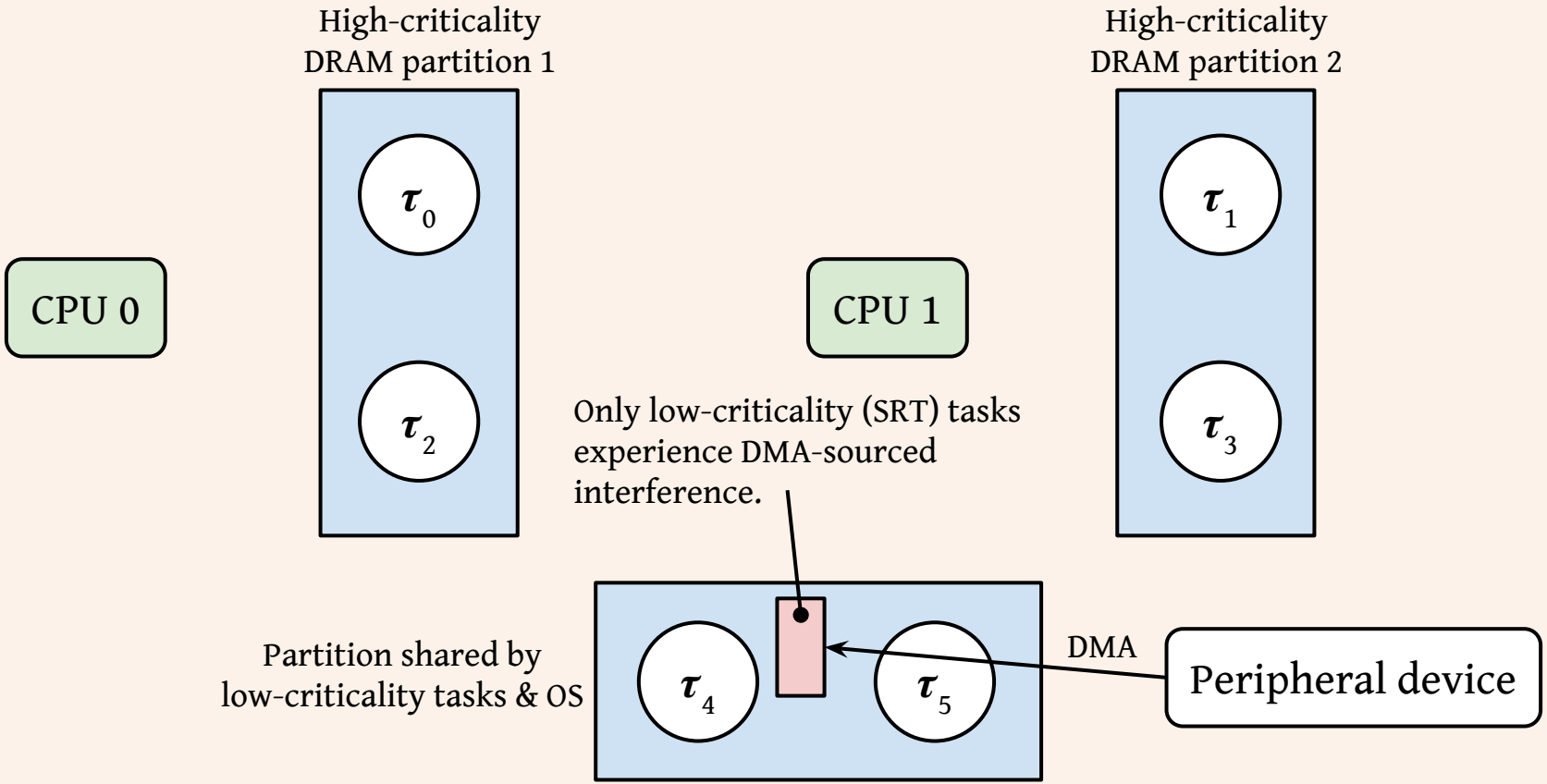
Schedulability Study Parameters

Category	Choice	Level A	Level B	Level C
1: Criticality Utilization Percent (%)	C-Light	[35, 45)	[35, 45)	[10, 30)
	C-Heavy	[10, 30)	[10, 30)	[50, 70)
	All-Mod.	[28, 39)	[28, 39)	[28, 39)
2: Period (ms)	Short	{12, 24}	{24, 48}	[12, 100)
	Moderate	{20, 40}	{40, 80}	[20, 100)
	Long	{48, 96}	{96, 192}	[50, 500)
3: Task Utilization	Light	[0.001, 0.03)	[0.001, 0.05)	[0.001, 0.1)
	Medium	[0.02, 0.1)	[0.05, 0.2)	[0.1, 0.4)
	Heavy	**[0.1, 0.2)**	[0.2, 0.4)	[0.4, 0.6)
4: Max Reload Time(%)	Quick	[1, 10)	[1, 10)	[1, 10)
	Slow	[25, 50)	[25, 50)	[25, 50)
5: IPC Size (bytes)	Small	{128, 256}	{128, 256}	{128,256}
	Large	{4096, 8192}	{4096, 8192}	{4096, 8192}
6: I/O Bandwidth (MB/s)	Low	[0,20]		
	Medium	[40, 60]		
	High	[180,200]		
7: Direct I/O Tasks(%)	Few	[0, 30]	[0, 30]	[0, 30]
	Many	[70, 100]	[70, 100]	[70,100]

Default DMA-buffer Allocation



Default DMA-buffer Allocation



Default DMA-buffer Allocation