TCLOUD

gVulkan: Scalable GPU Pooling for Pixel-Grained Rendering in Ray Tracing

Yicheng Gu, Yun Wang, Yunfan Sun, Yuxin Xiang,
Yufan Jiang, Xuyan Hu, Zhengwei Qi, Haibing Guan

Shanghai Jiao Tong University

Ray Tracing Delivers Realistic Visual Experiences

- **Rendering technologies**
  - Rasterization
  - Ray Tracing

https://www.youtube.com/watch?v=NZdScm6SWYw

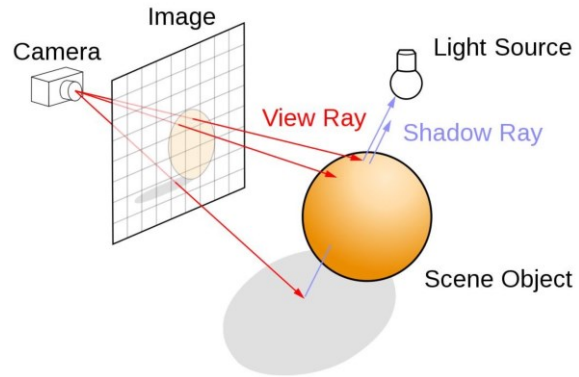
Ray Tracing Delivers Realistic Visual Experiences

- **Rendering technologies**
  - Rasterization
  - Ray Tracing

https://www.youtube.com/watch?v=NZdScm6SWYw

Ray Tracing Delivers Realistic Visual Experiences

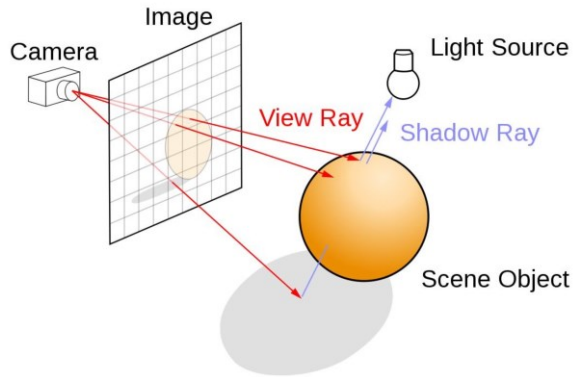
- **Rendering technologies**
  - Rasterization
  - Ray Tracing



https://www.youtube.com/watch?v=NZdScm6SWYw

Ray Tracing Delivers Realistic Visual Experiences

- **Rendering technologies**
  - Rasterization
  - Ray Tracing

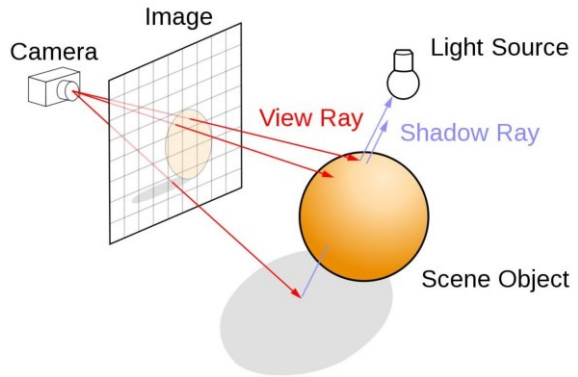


- The advent of hardware ray tracing acceleration
- Engine, game support for real-time ray tracing
- XR demands for immersive experiences

https://www.youtube.com/watch?v=NZdScm6SWYw

Ray Tracing Delivers Realistic Visual Experiences

- **Rendering technologies**
  - Rasterization
  - Ray Tracing

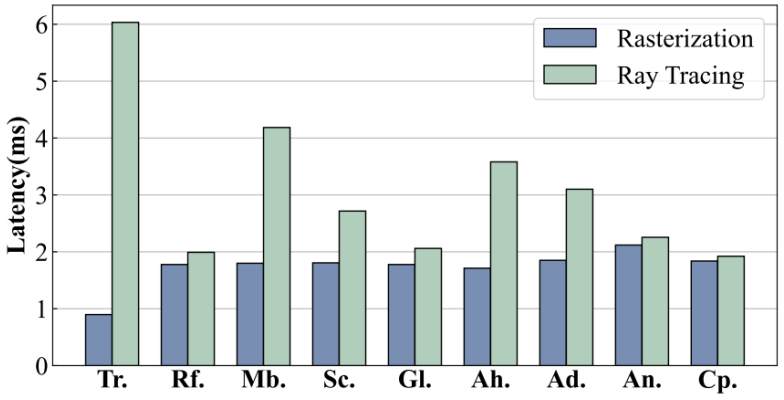


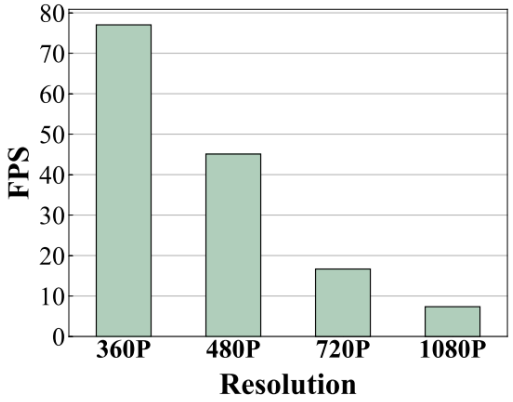
- The advent of hardware ray tracing acceleration
- Engine, game support for real-time ray tracing
- XR demands for immersive experiences

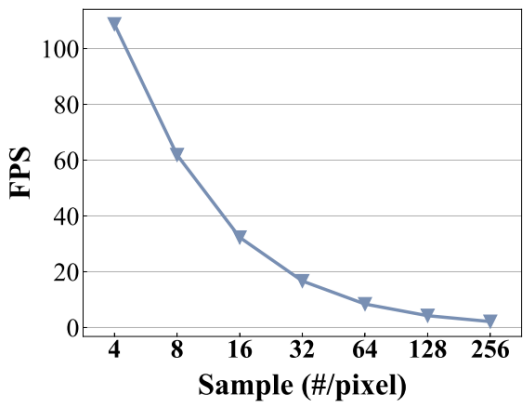
https://www.youtube.com/watch?v=NZdScm6SWYw

Bottlenecks of ray tracing

- **Limitation of users**
- **Higher ray tracing latency compared to rasterization**



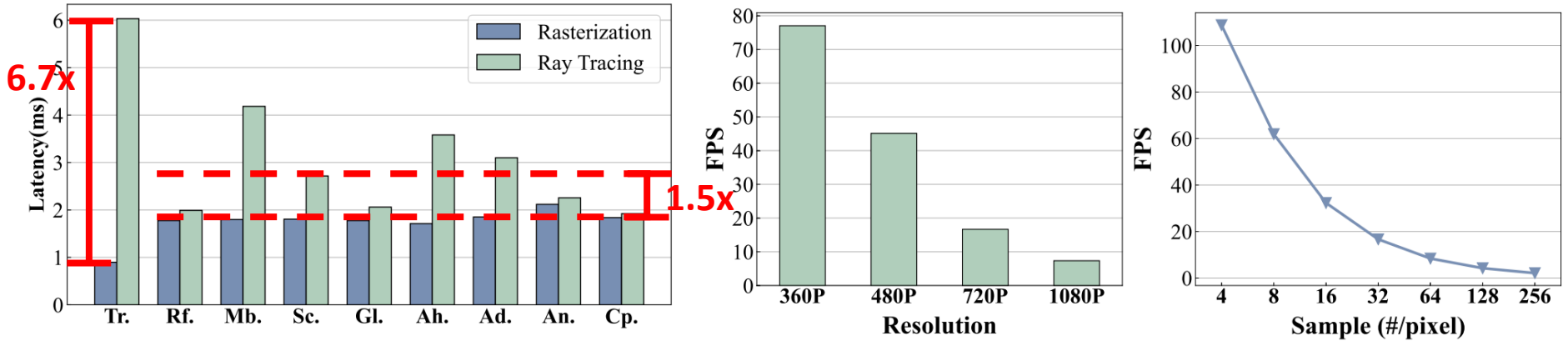




https://github.com/nvpro-samples/vk_raytracing_tutorial_KHR

Bottlenecks of ray tracing

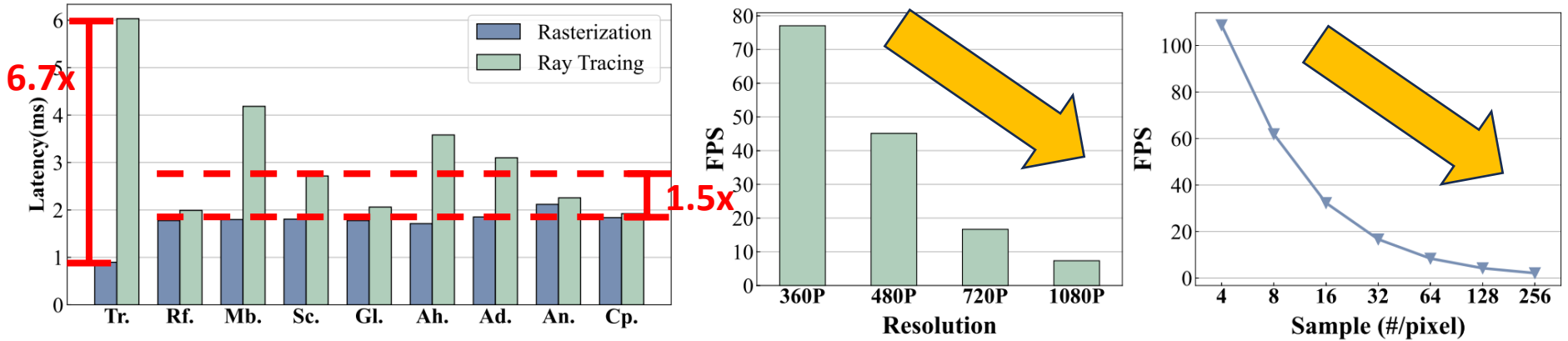
- Limitation of users
- Higher ray tracing latency compared to rasterization



https://github.com/nvpro-samples/vk_raytracing_tutorial_KHR

Bottlenecks of ray tracing

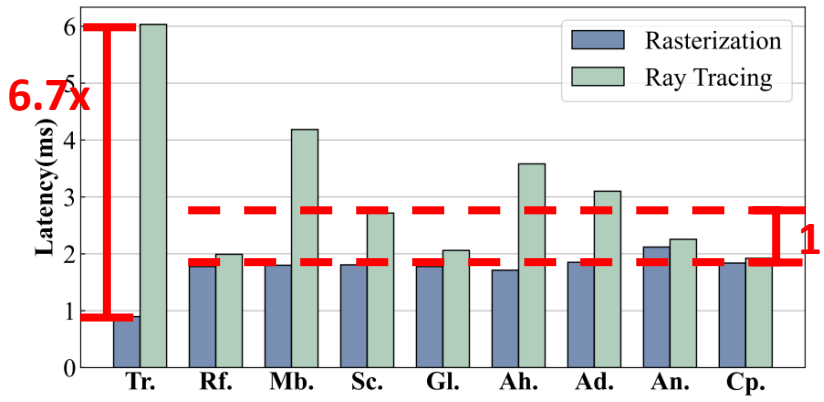
- **Limitation of users**
- **Higher ray tracing latency compared to rasterization**

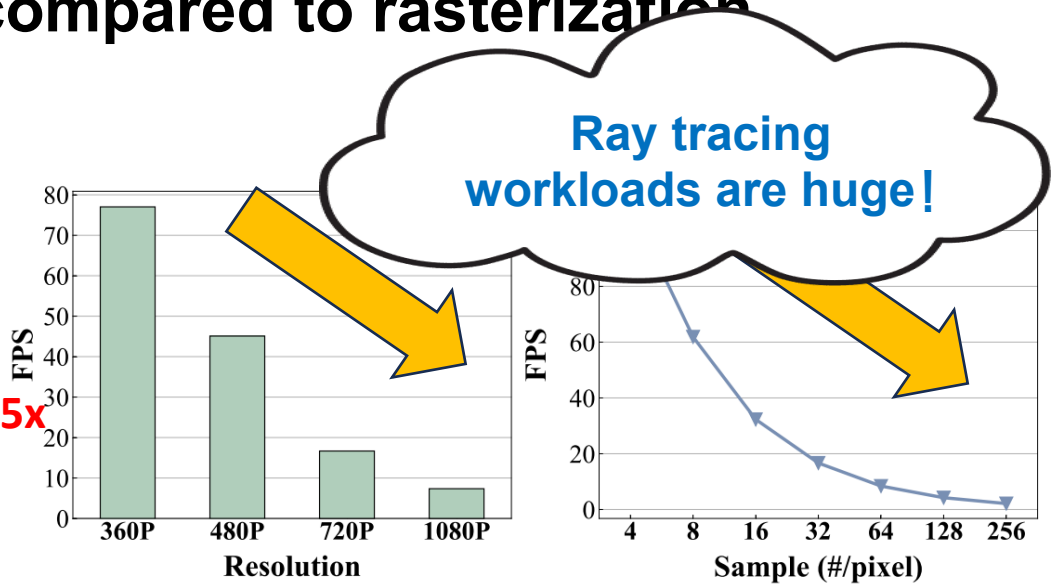


https://github.com/nvpro-samples/vk_raytracing_tutorial_KHR

Bottlenecks of ray tracing

- Limitation of users
- Higher ray tracing latency compared to rasterization

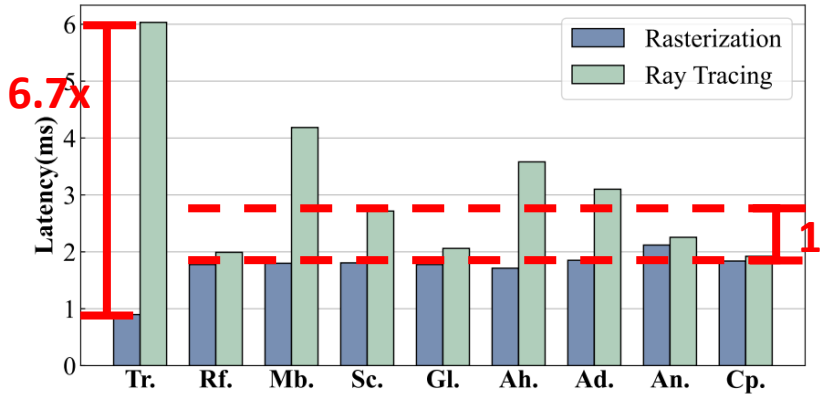


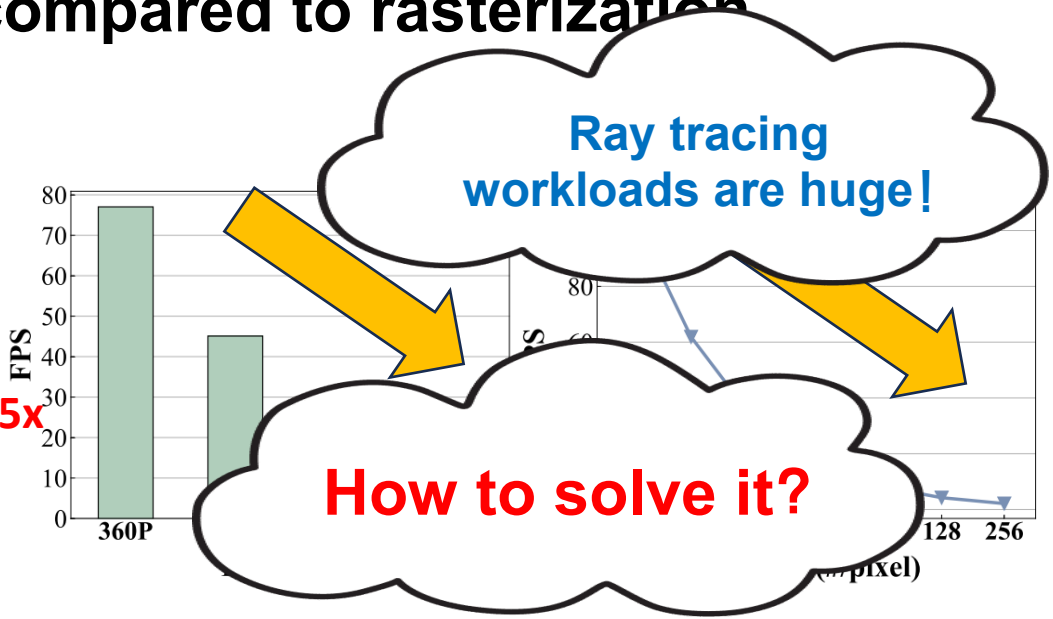


https://github.com/nvpro-samples/vk_raytracing_tutorial_KHR

Bottlenecks of ray tracing

- **Limitation of users**
- **Higher ray tracing latency compared to rasterization**





https://github.com/nvpro-samples/vk_raytracing_tutorial_KHR

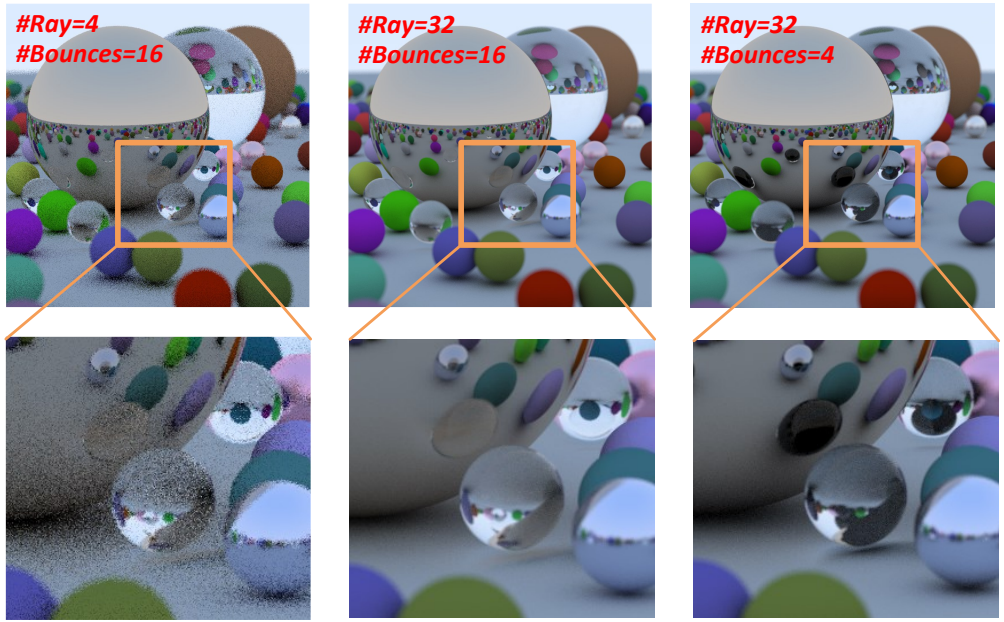
Reduce latency for ray tracing

- **Reduce workloads**
  - Reduce the amount of light
  - Reduce the number of bounce

- **Multi-GPU solutions**
  - AFR (Alternate Frame Rendering)
  - SFR (Split Frame Rendering)

Reduce latency for ray tracing

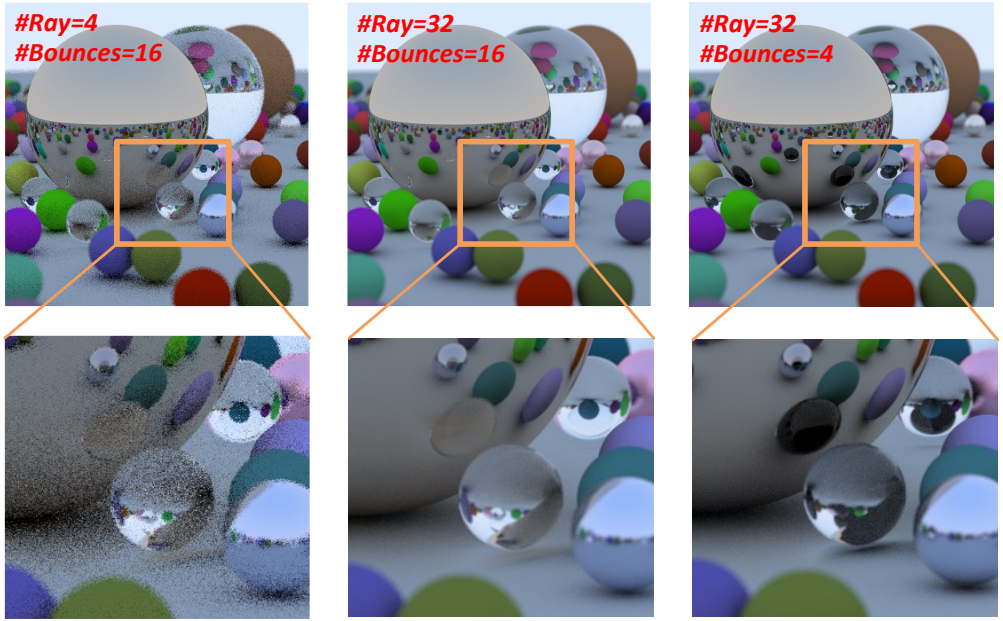
- **Reduce workloads**
  - Reduce the amount of light
  - Reduce the number of bounce



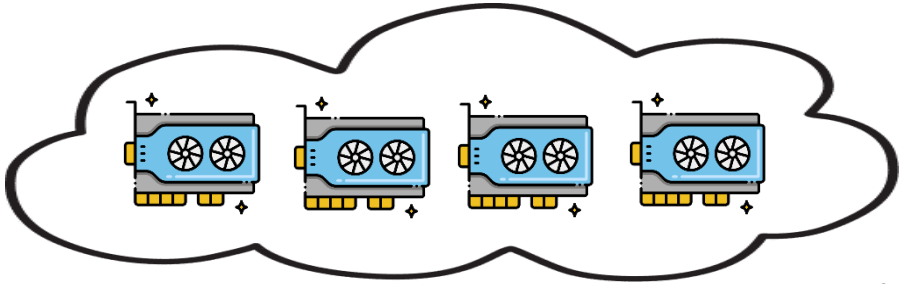
- **Multi-GPU solutions**
  - AFR (Alternate Frame Rendering)
  - SFR (Split Frame Rendering)

Reduce latency for ray tracing

- **Reduce workloads**
  - Reduce the amount of light
  - Reduce the number of bounce

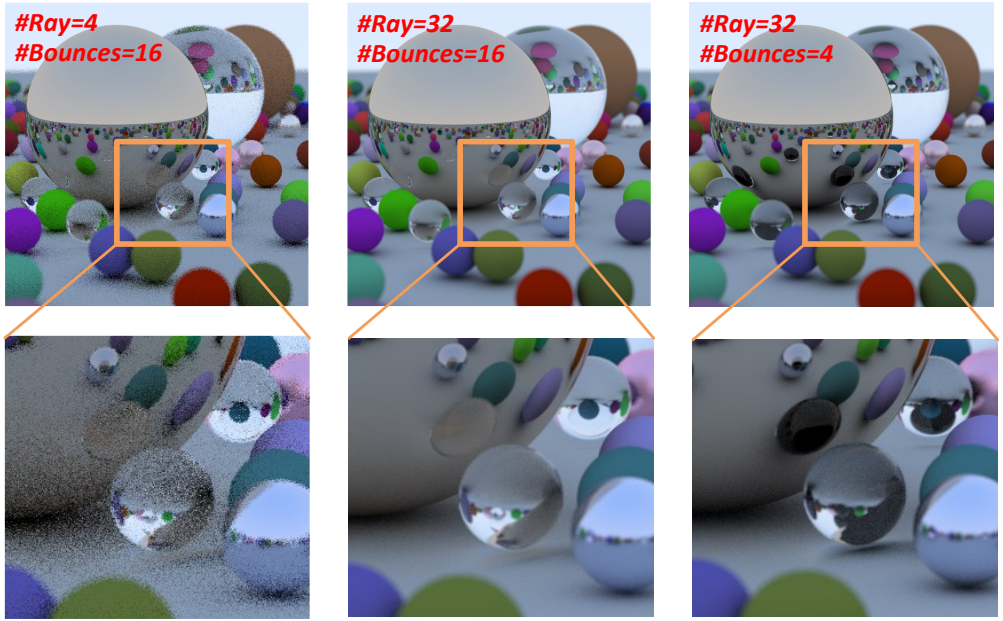


- **Multi-GPU solutions**
  - AFR (Alternate Frame Rendering)
  - SFR (Split Frame Rendering)

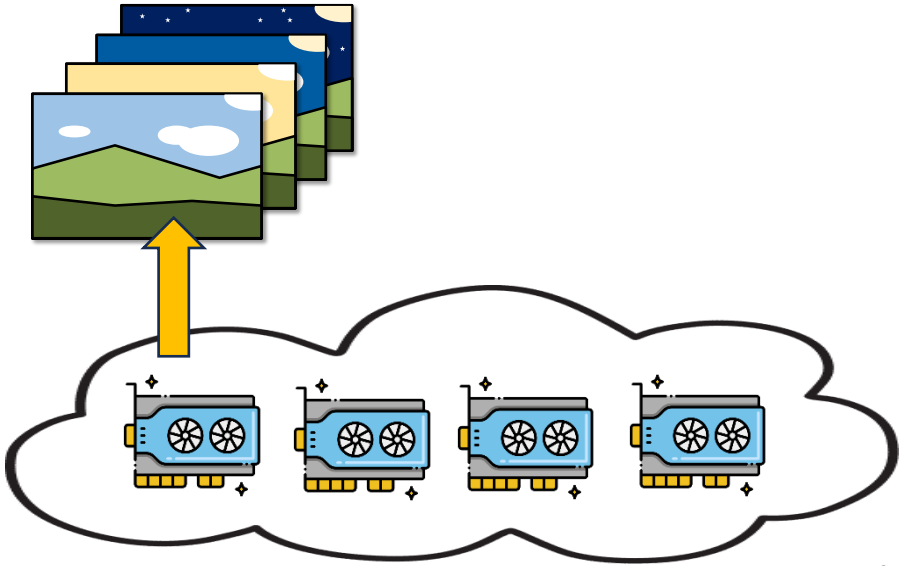


Reduce latency for ray tracing

- **Reduce workloads**
  - Reduce the amount of light
  - Reduce the number of bounce

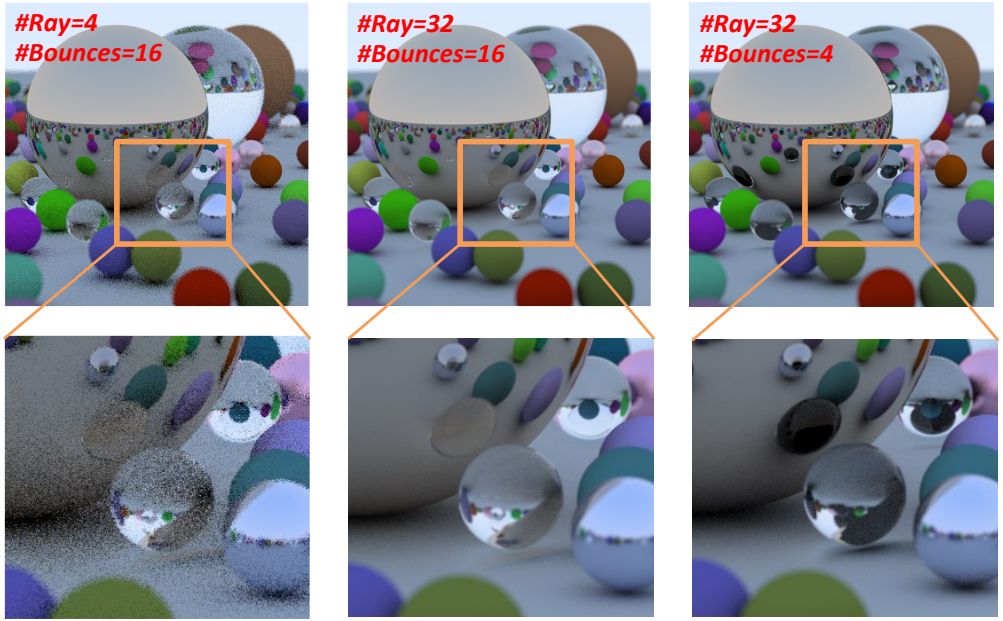


- **Multi-GPU solutions**
  - AFR (Alternate Frame Rendering)
  - SFR (Split Frame Rendering)

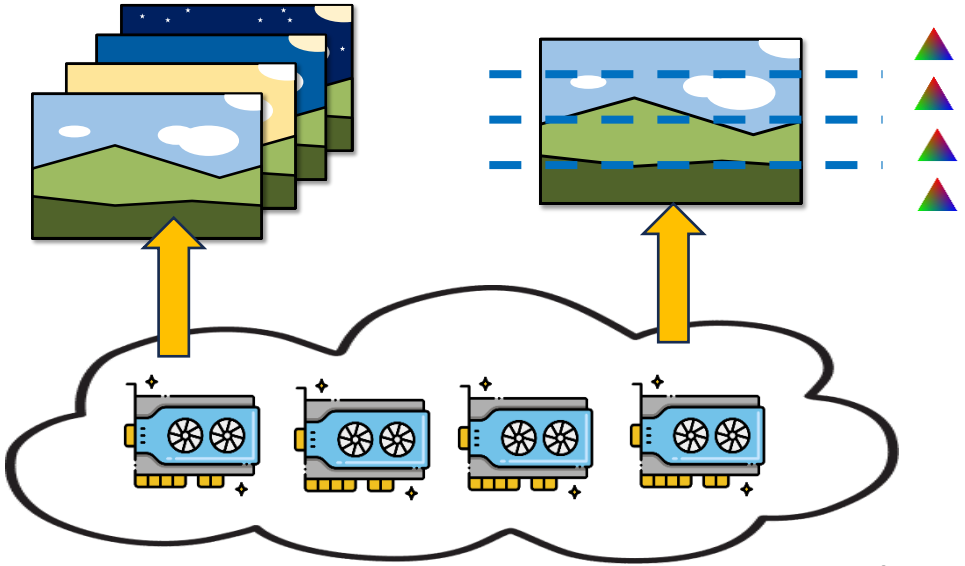


Reduce latency for ray tracing

- **Reduce workloads**
  - Reduce the amount of light
  - Reduce the number of bounce



- **Multi-GPU solutions**
  - AFR (Alternate Frame Rendering)
  - SFR (Split Frame Rendering)



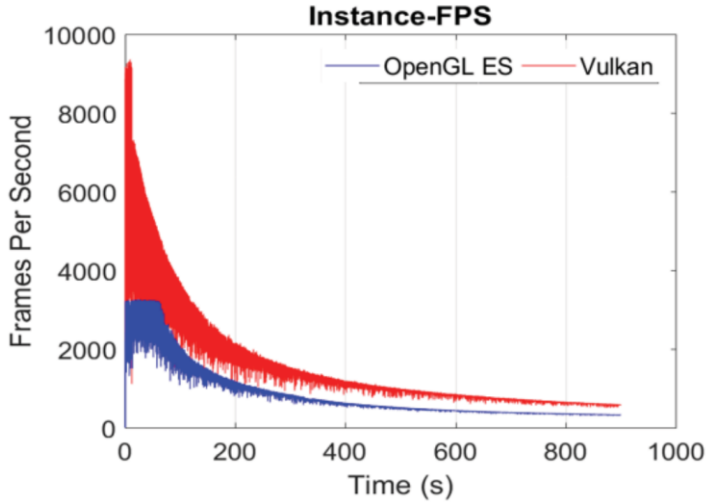
Why is Vulkan

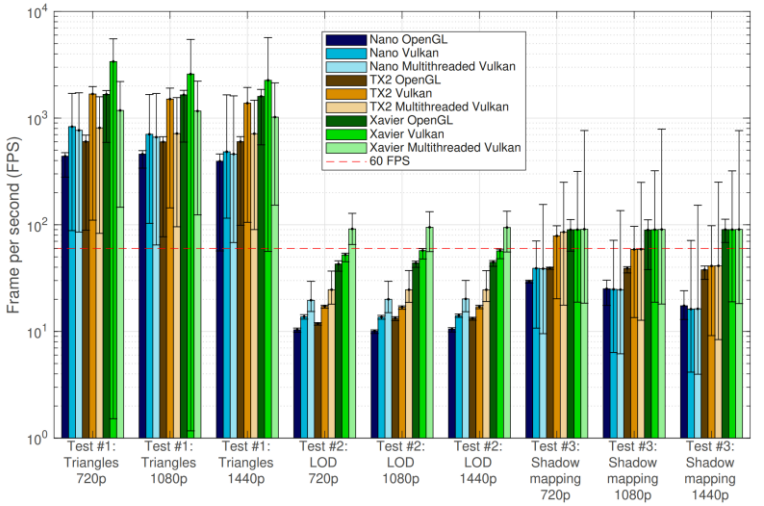
- **Increase Efficiency with Vulkan**
  - the Next-Generation Graphics API

Ferraz O, Menezes P, Silva V, et al. Benchmarking Vulkan vs OpenGL Rendering on Low-Power Edge GPUs[C]//2021 International Conference on Graphics and Interaction (ICGI). IEEE, 2021: 1-8.
Lujan M, Baum M, Chen D, et al. Evaluating the performance and energy efficiency of opengl and vulkan on a graphics rendering server[C]//2019 International Conference on Computing, Networking and Communications (ICNC). IEEE, 2019: 777-781.

Why is Vulkan

- **Increase Efficiency with Vulkan**
  - the Next-Generation Graphics API

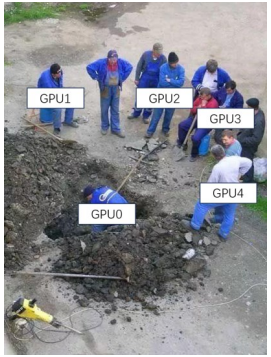




Ferraz O, Menezes P, Silva V, et al. Benchmarking Vulkan vs OpenGL Rendering on Low-Power Edge GPUs[C]//2021 International Conference on Graphics and Interaction (ICGI). IEEE, 2021: 1-8.

Lujan M, Baum M, Chen D, et al. Evaluating the performance and energy efficiency of opengl and vulkan on a graphics rendering server[C]//2019 International Conference on Computing, Networking and Communications (ICNC). IEEE, 2019: 777-781.

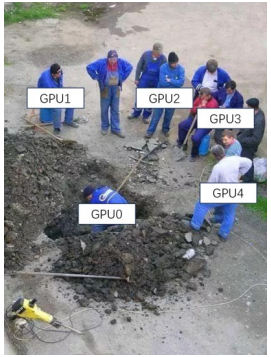
Challenges of multi-GPU ray tracing



Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources



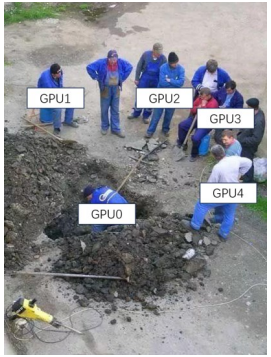
Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources

Vulkan

Intra-GPU : **Unnecessary waiting and work**



Challenges of multi-GPU ray tracing

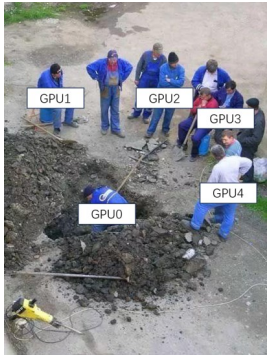
Ecosystem compatibility :

Modifying code or wasting resources

Vulkan

Intra-GPU : **Unnecessary waiting and work**

Inter-GPU : **Uneven GPU utilization**



Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources

Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization

Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources

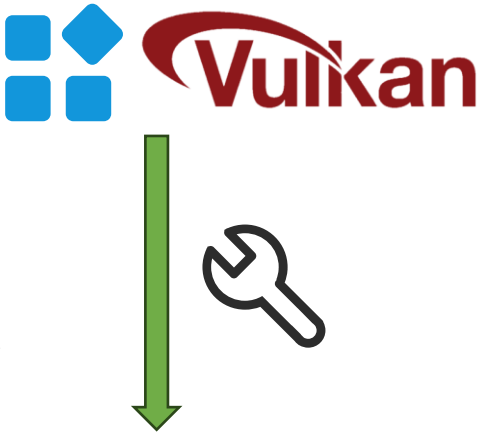
Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization

Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources



Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization

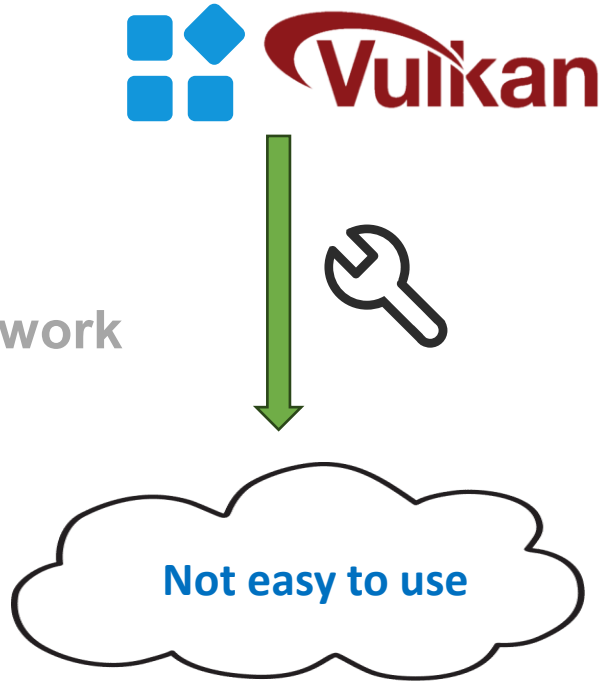
Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources

Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization



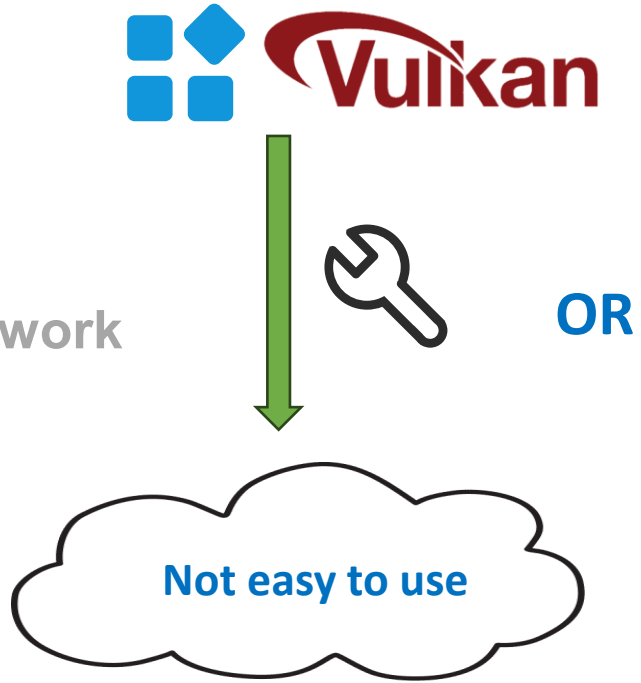
Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources

Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization



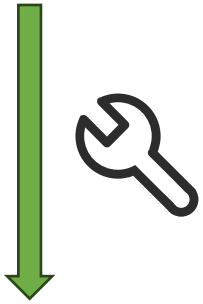
Challenges of multi-GPU ray tracing

Ecosystem compatibility :

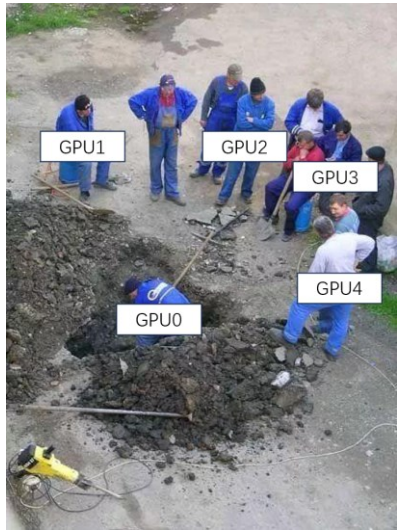
Modifying code or wasting resources

Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization



OR



Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources

Intra-GPU : **Unnecessary waiting and work**

Inter-GPU : Uneven GPU utilization

Challenges of multi-GPU ray tracing

GPU CPUs

Ecosystem compatibility :

Modifying code or wasting resources

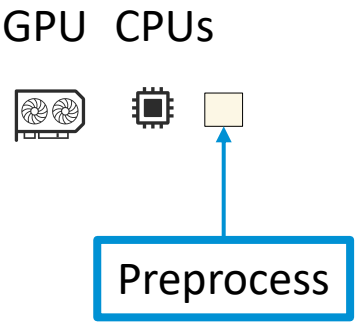
Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization

Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources



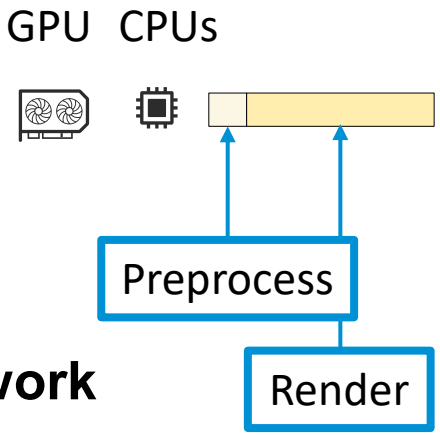
Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization

Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources



Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization

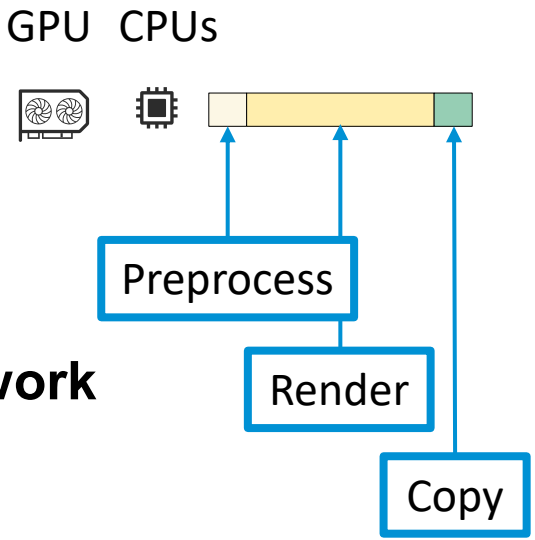
Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources

Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization



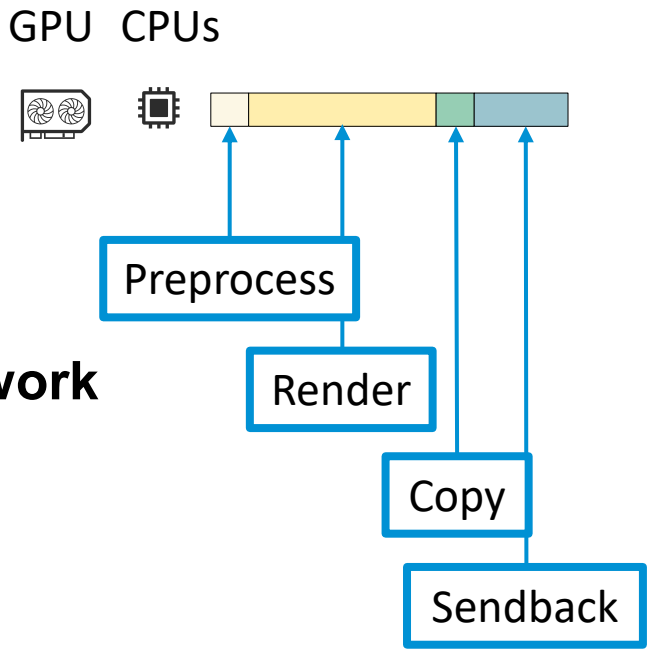
Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources

Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization



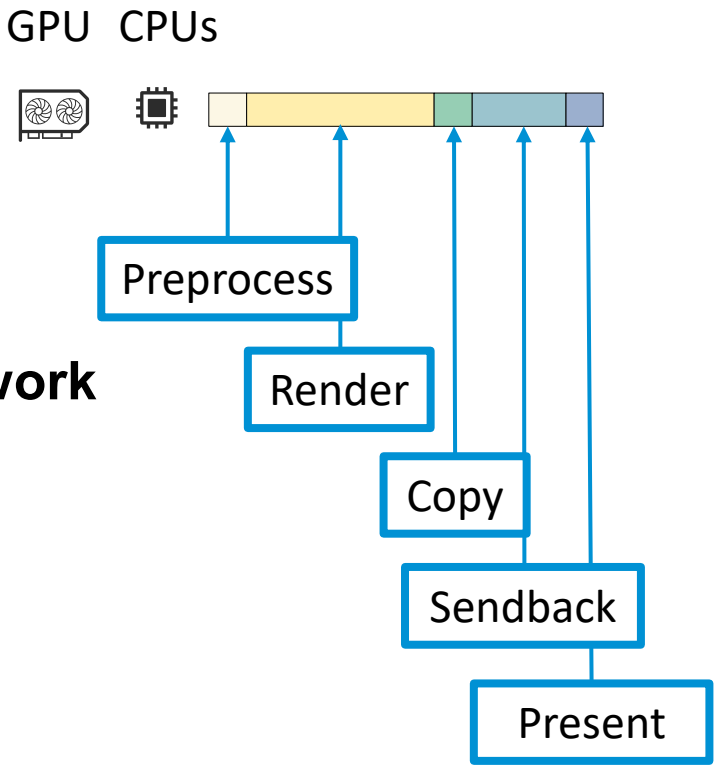
Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources

Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization



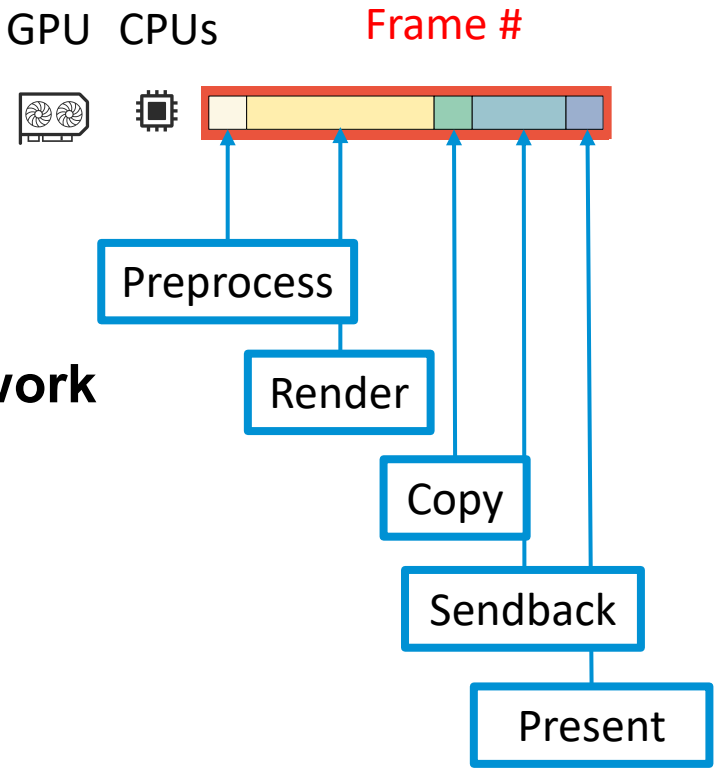
Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources

Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization



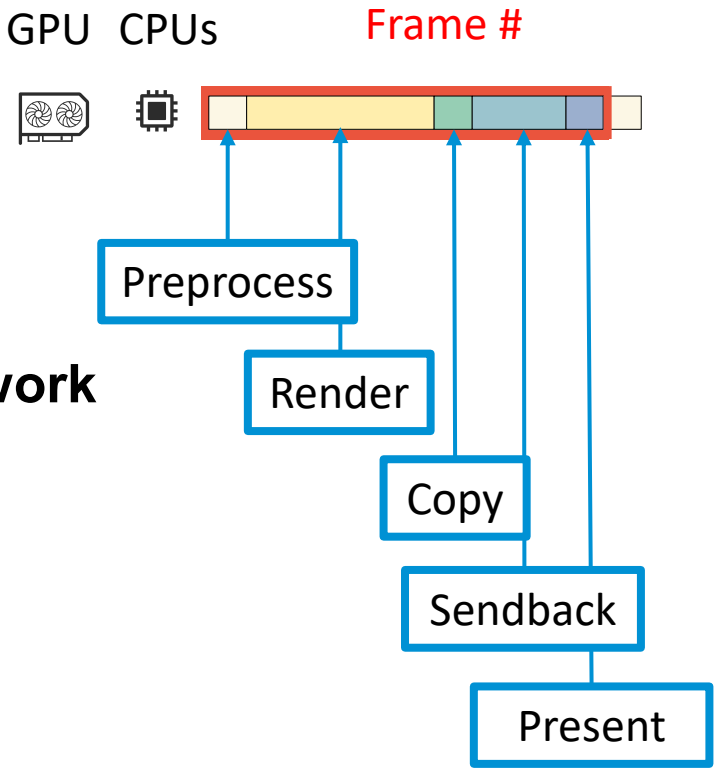
Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources

Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization



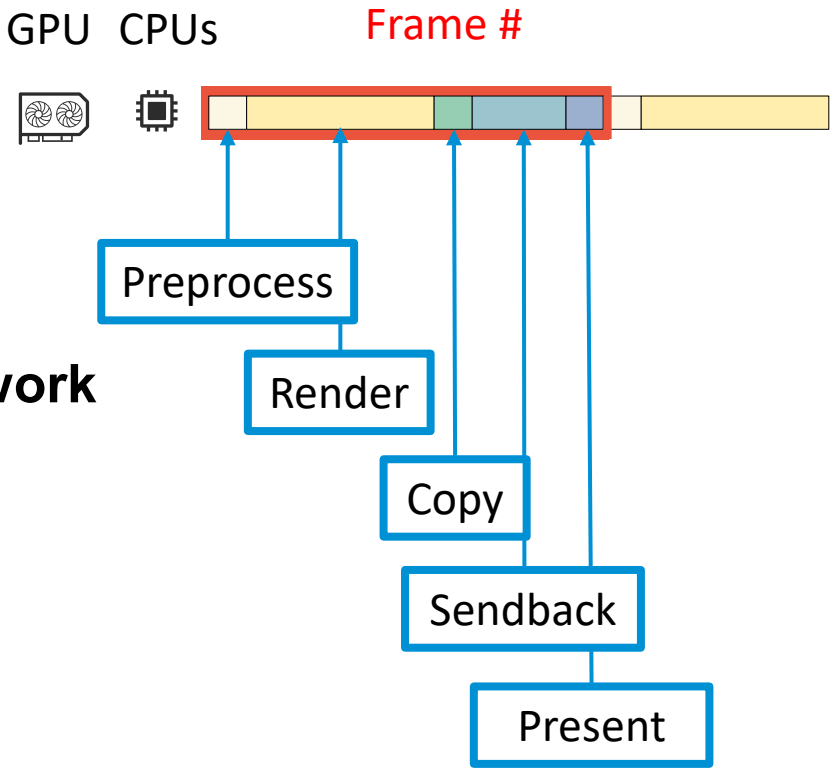
Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources

Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization



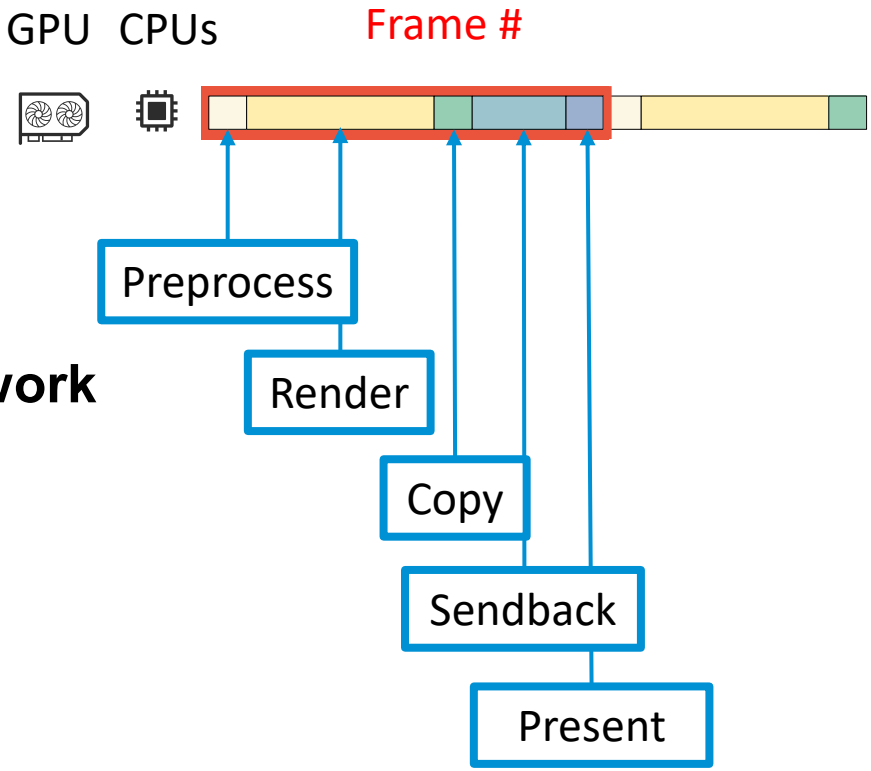
Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources

Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization



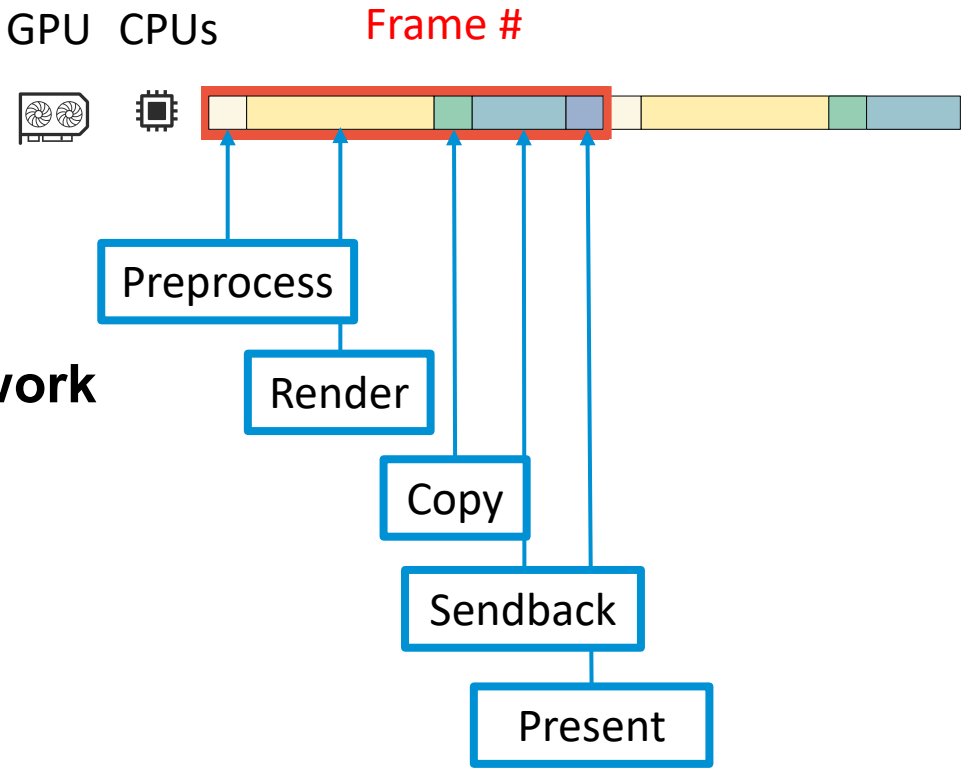
Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources

Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization



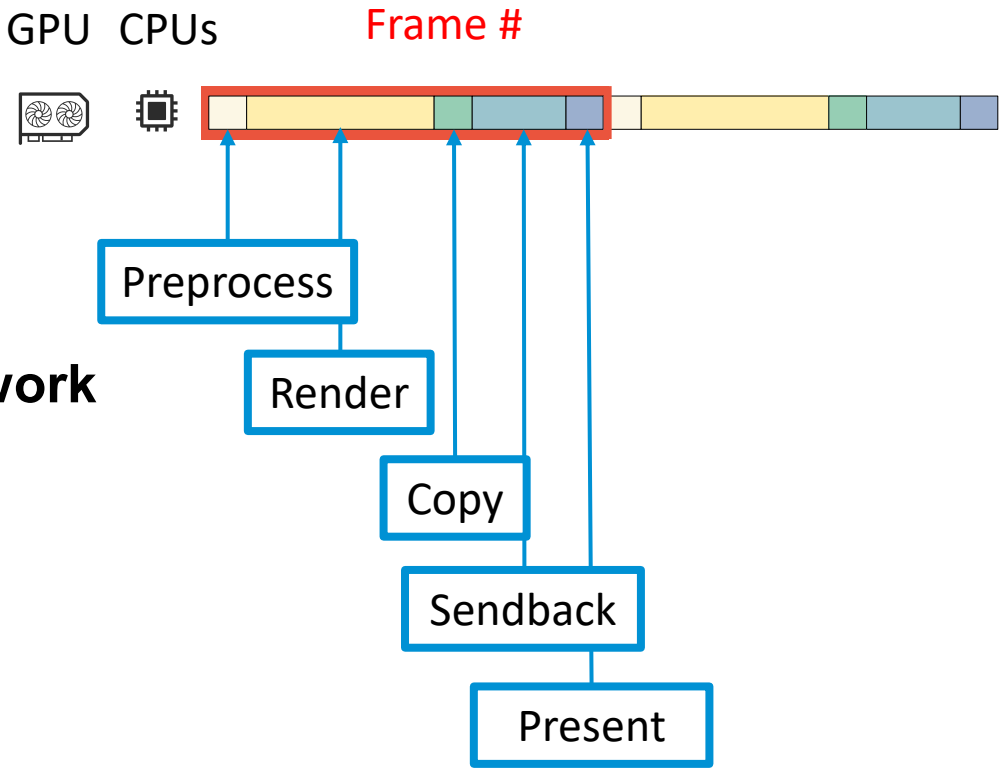
Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources

Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization



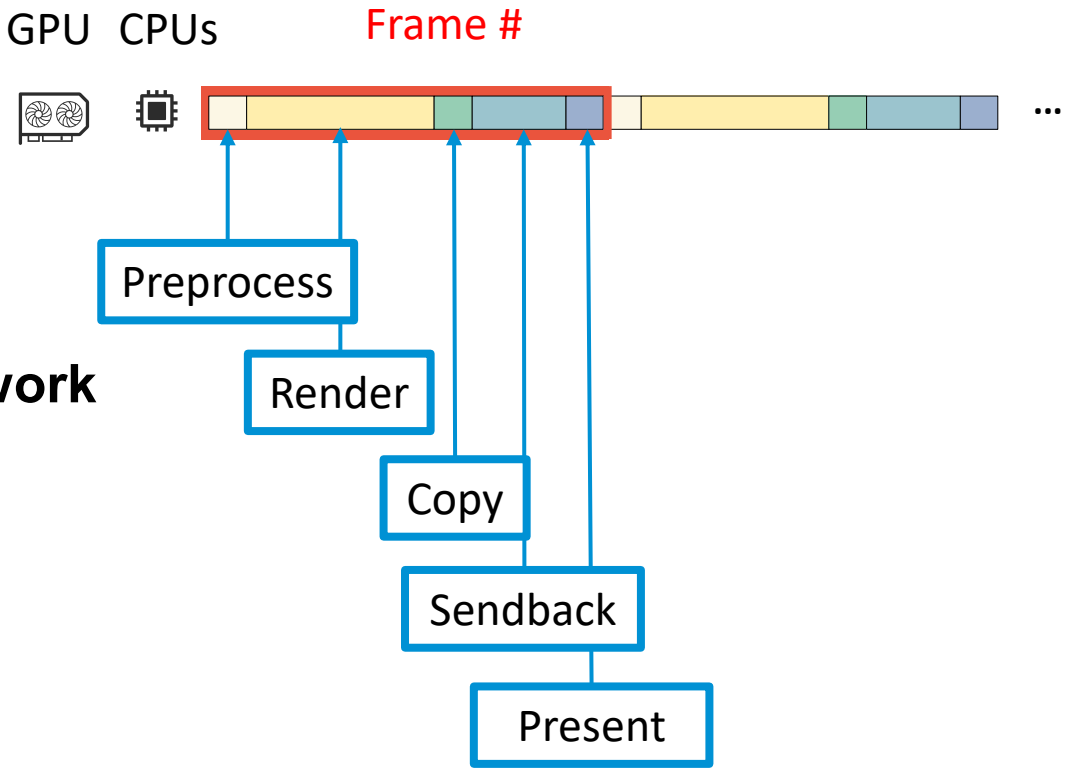
Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources

Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization



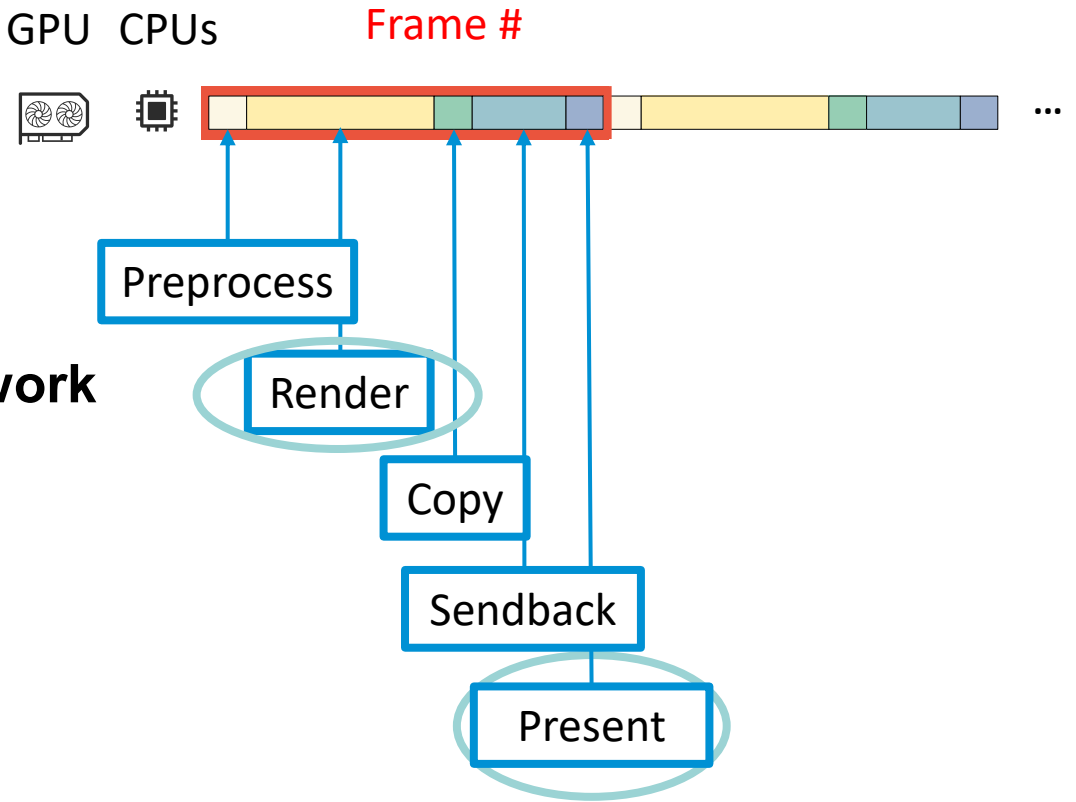
Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources

Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization



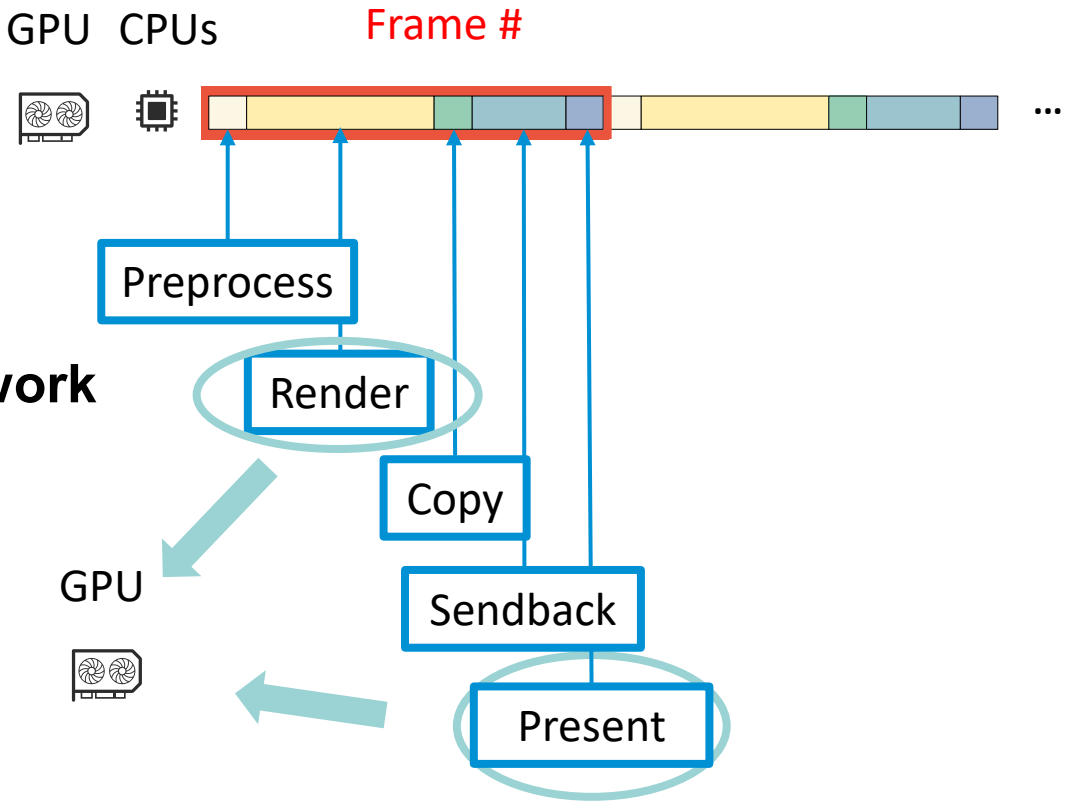
Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources

Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization



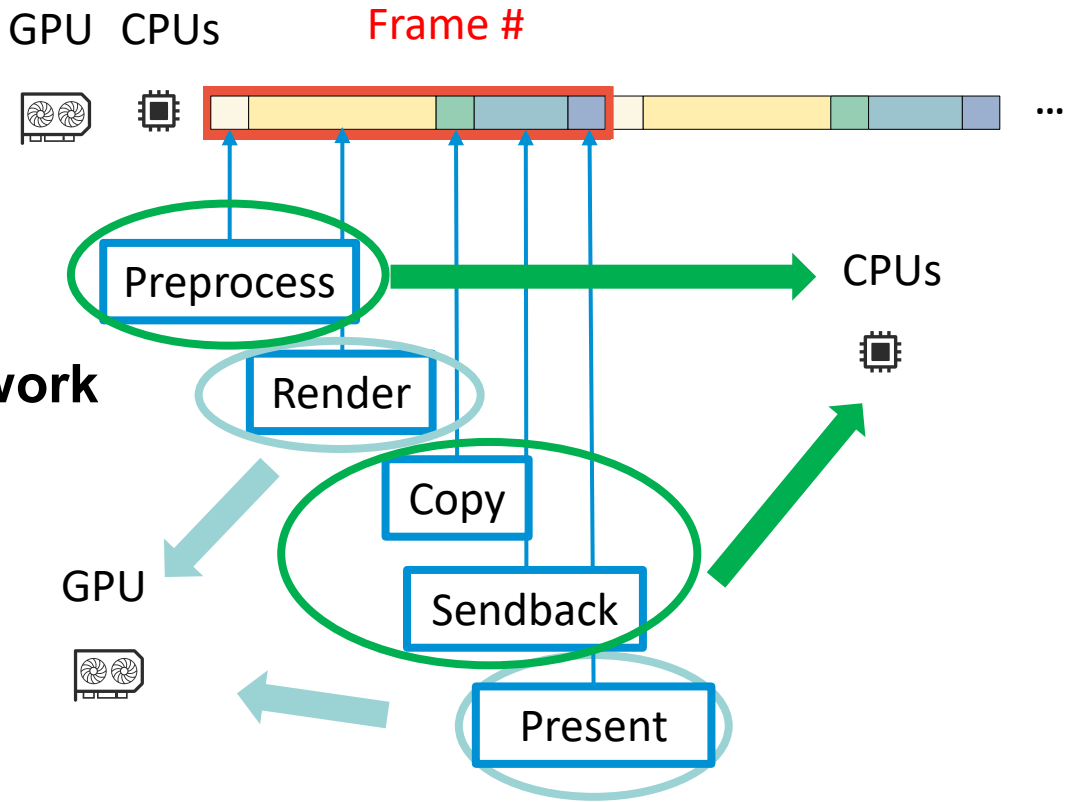
Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources

Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization



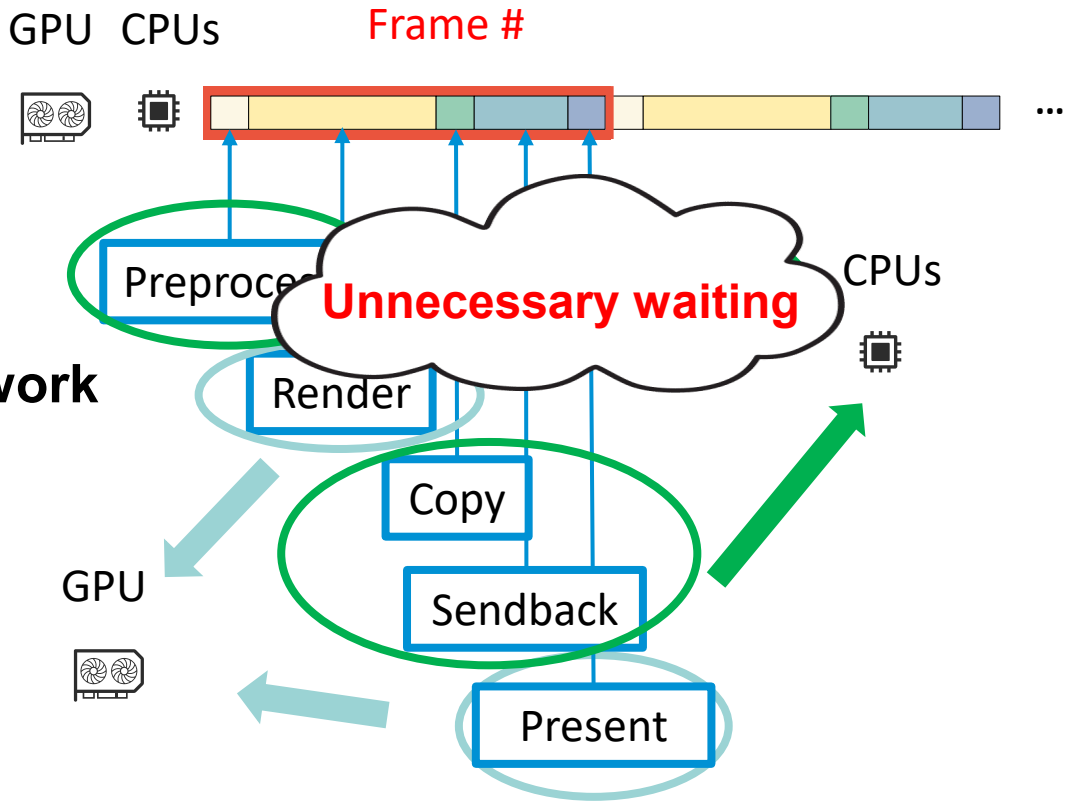
Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources

Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization



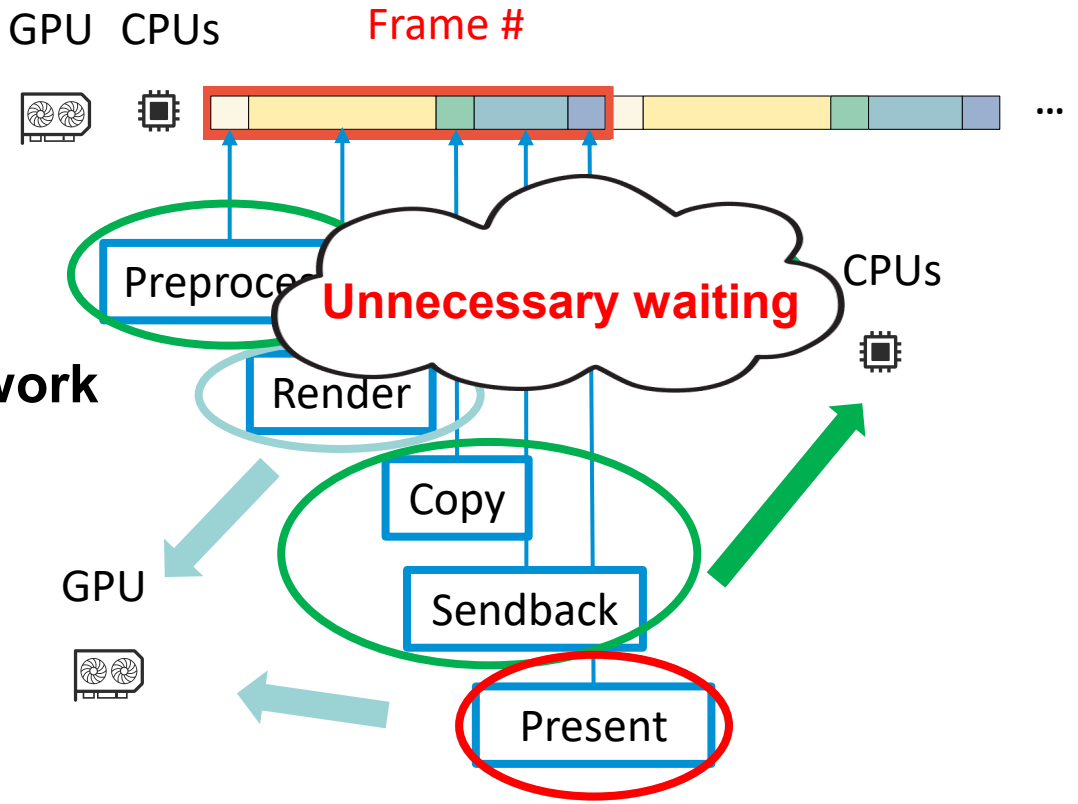
Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources

Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization



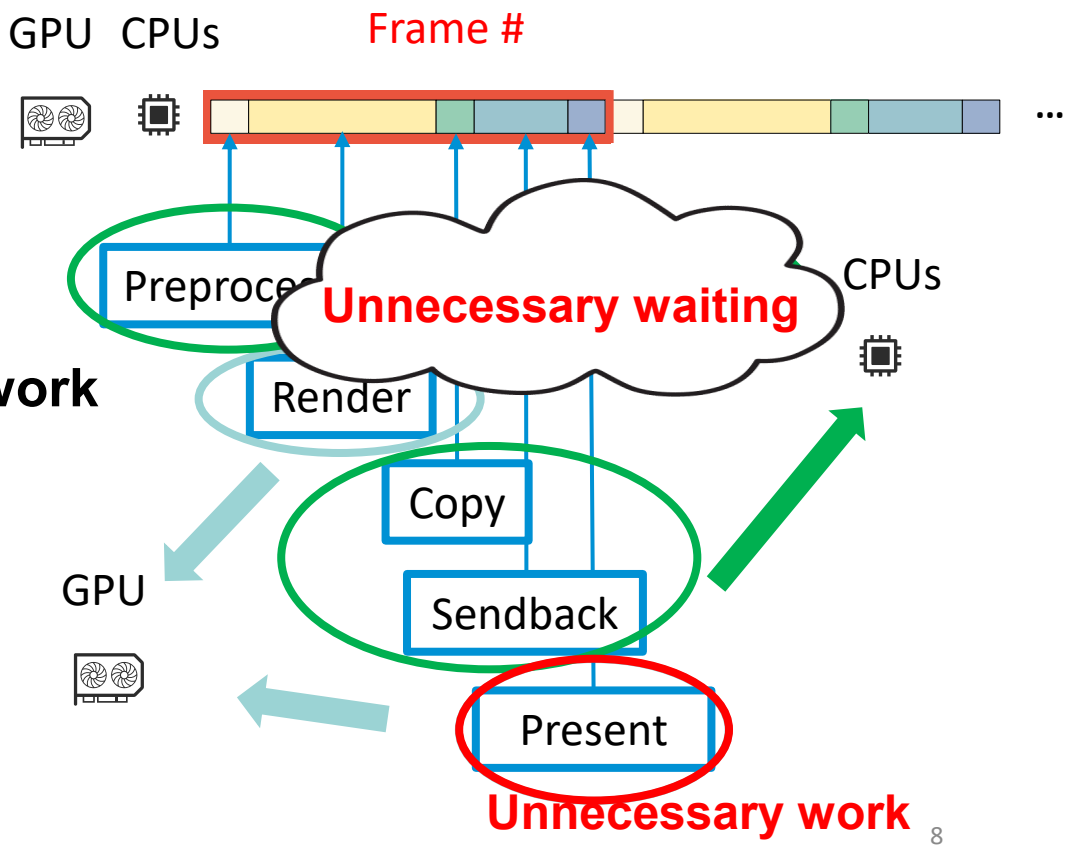
Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources

Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization



Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resources

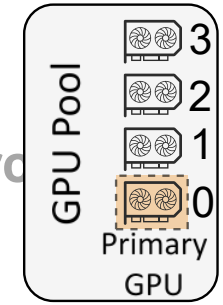
Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization

Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resourc



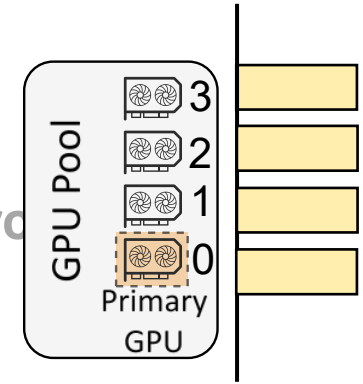
Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization

Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resourc



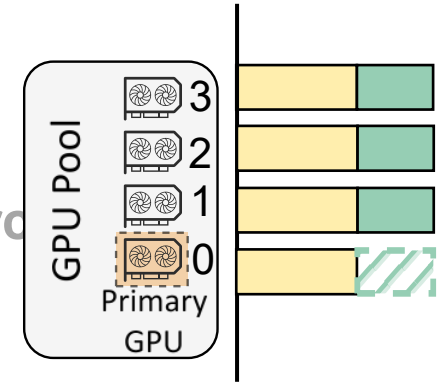
Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization

Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resourc



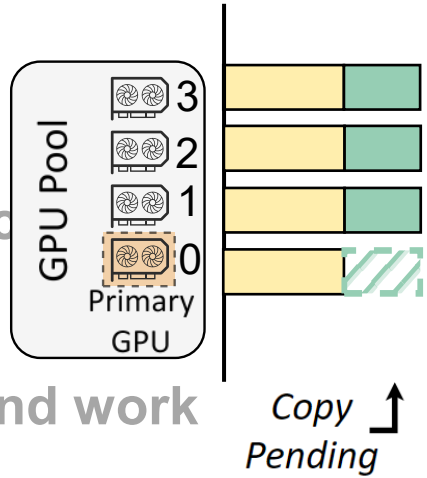
Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization

Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resourc



Intra-GPU : Unnecessary waiting and work

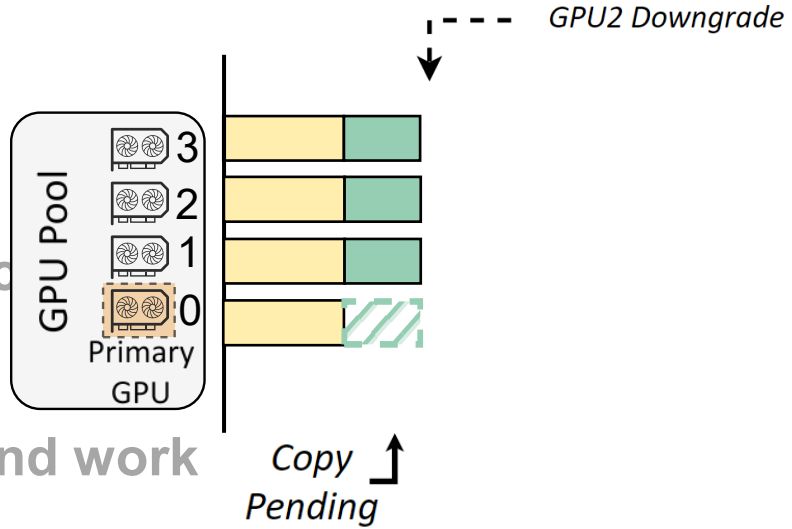
Inter-GPU : Uneven GPU utilization

Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resourc

Intra-GPU : Unnecessary waiting and work



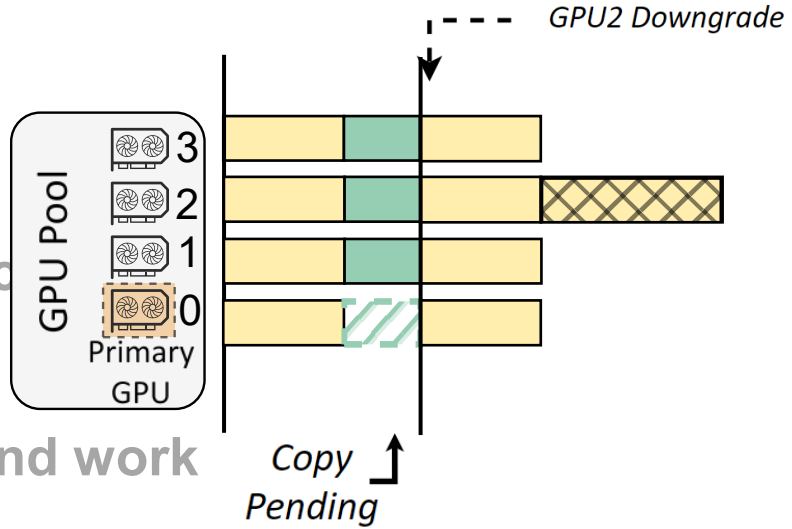
Inter-GPU : Uneven GPU utilization

Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resourc

Intra-GPU : Unnecessary waiting and work



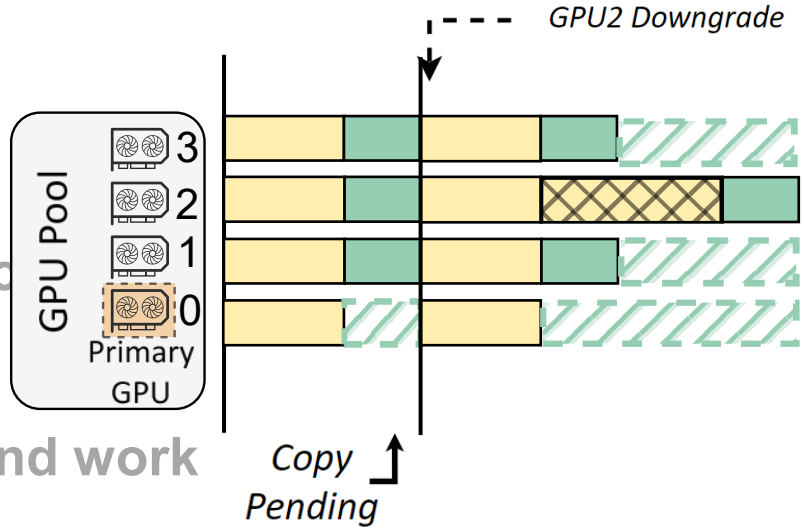
Inter-GPU : Uneven GPU utilization

Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resourc

Intra-GPU : Unnecessary waiting and work



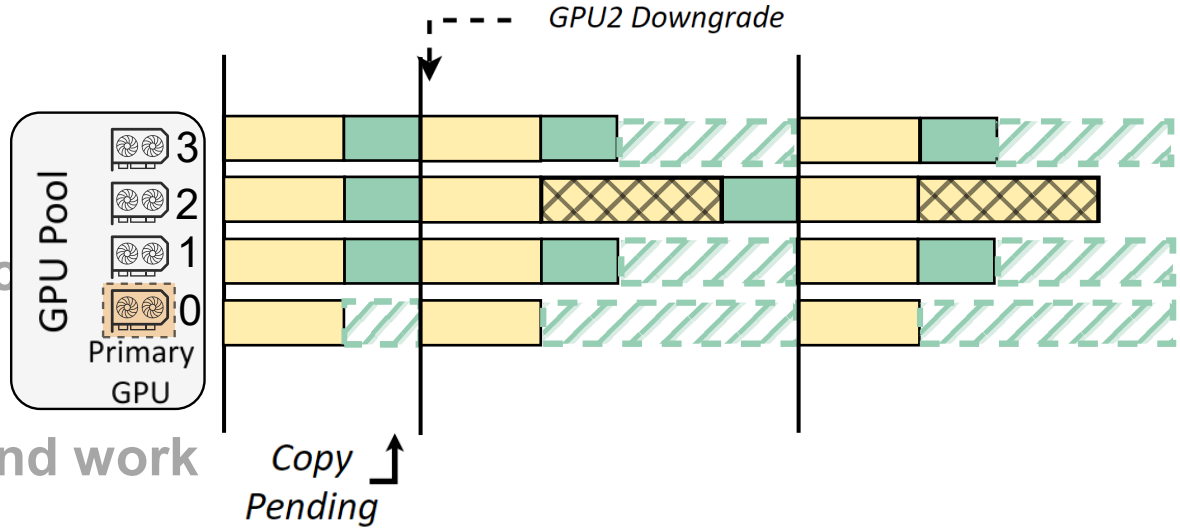
Inter-GPU : Uneven GPU utilization

Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resourc

Intra-GPU : Unnecessary waiting and work



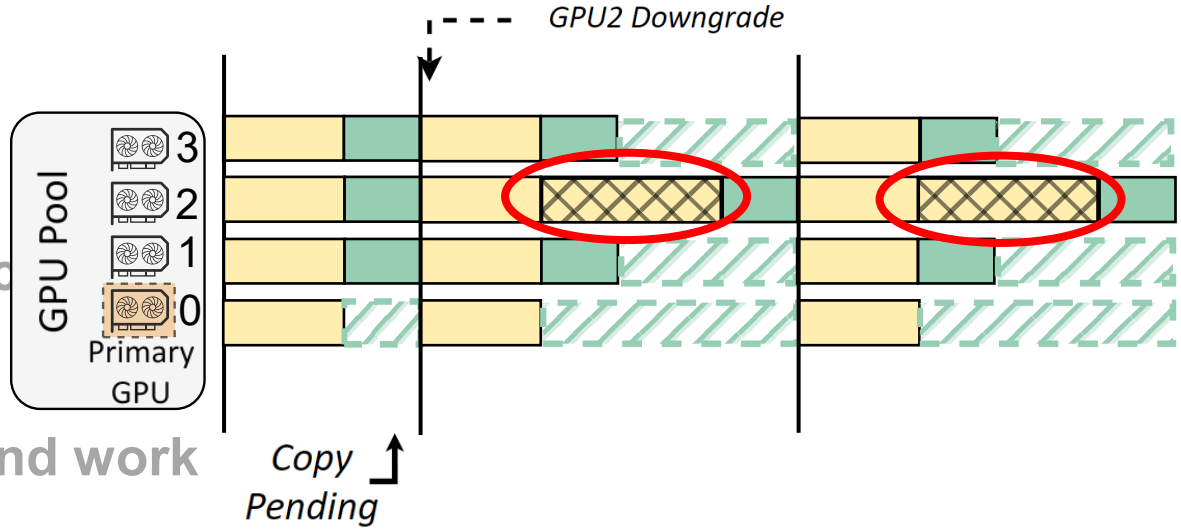
Inter-GPU : Uneven GPU utilization

Challenges of multi-GPU ray tracing

Ecosystem compatibility :

Modifying code or wasting resourc

Intra-GPU : Unnecessary waiting and work



Inter-GPU : Uneven GPU utilization

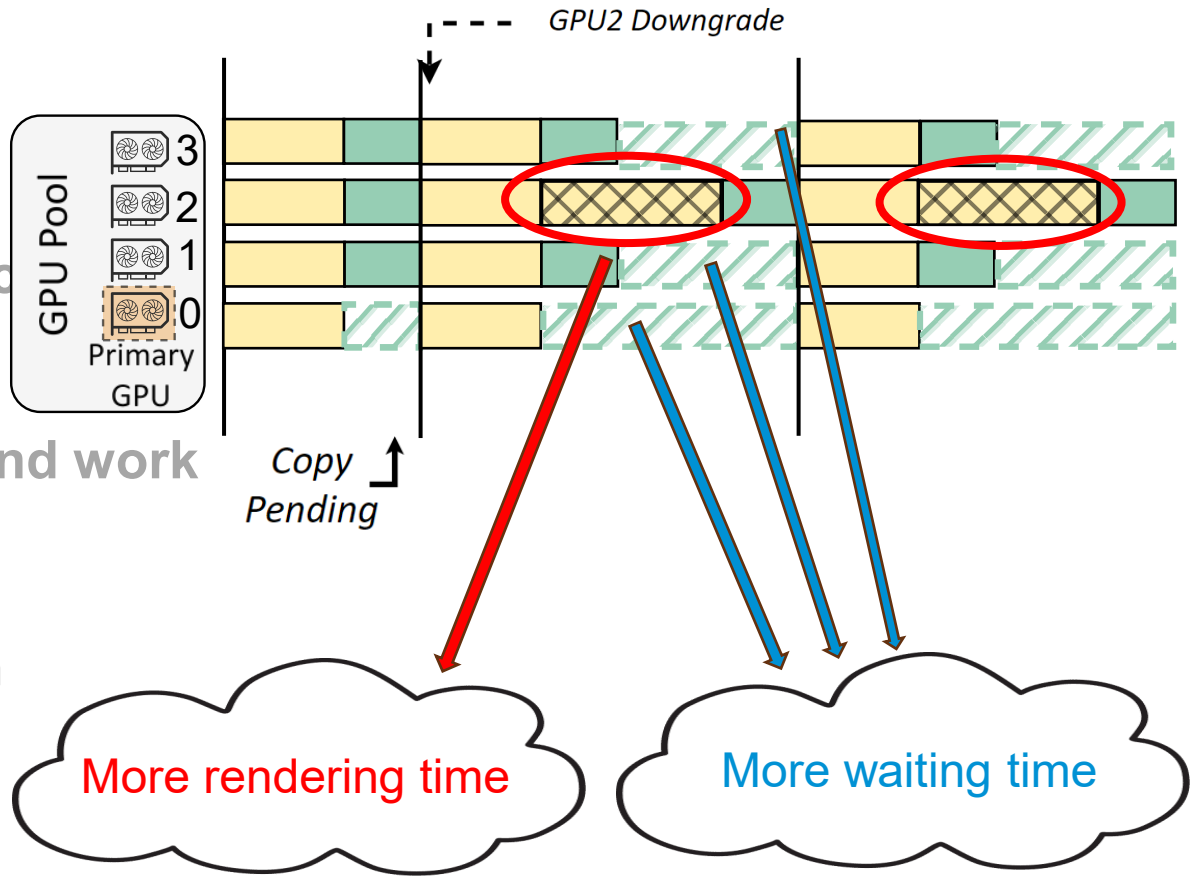
Challenges of multi-GPU ray tracing

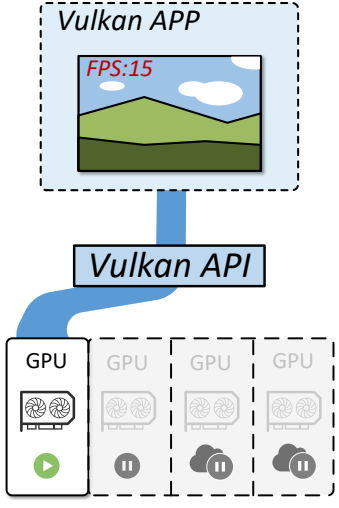
Ecosystem compatibility :

Modifying code or wasting resourc

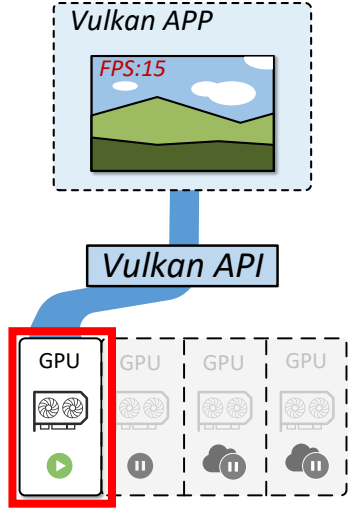
Intra-GPU : Unnecessary waiting and work

Inter-GPU : Uneven GPU utilization

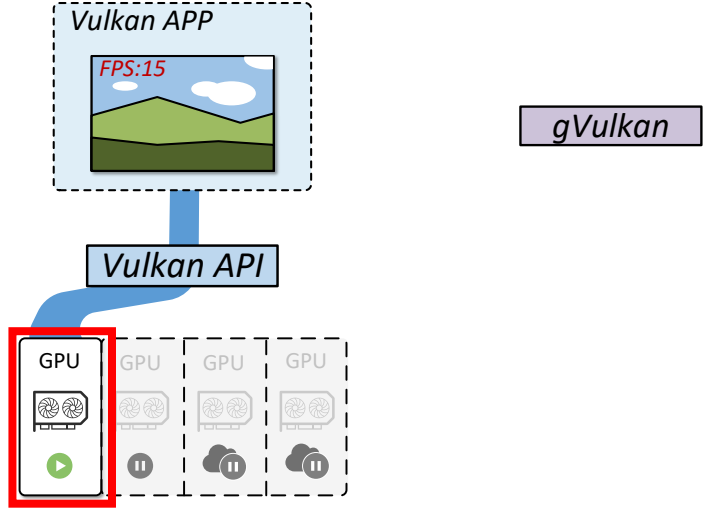




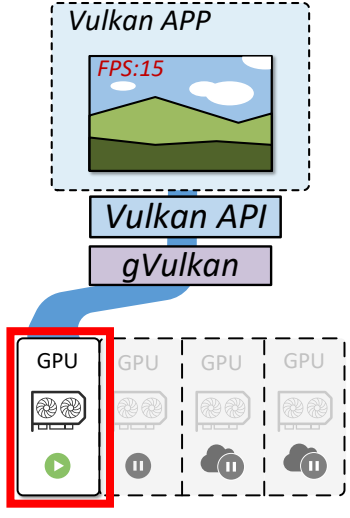
Our Design : gVulkan



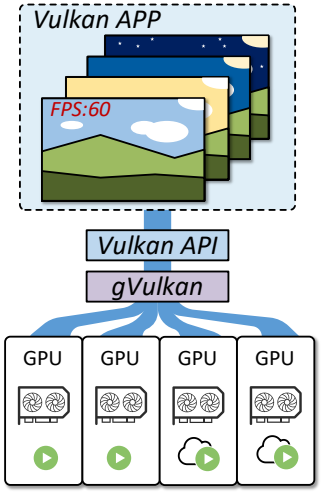
Our Design : gVulkan



Our Design : gVulkan

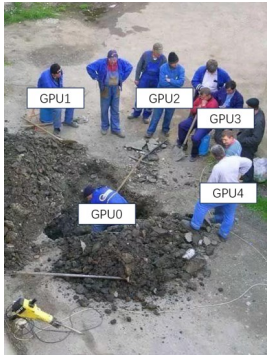


Our Design : gVulkan



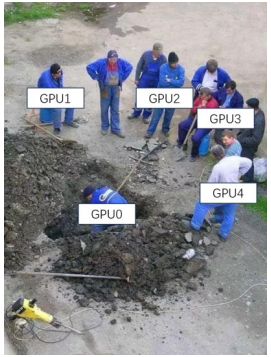
Our Design : gVulkan

gVulkan goals



gVulkan goals

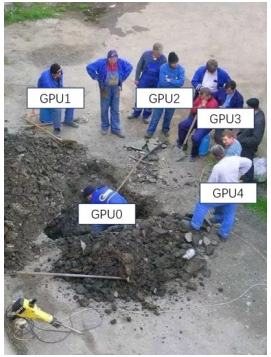
Ecosystem compatibility : API-forwarding, Cloud Rendering



gVulkan goals

Ecosystem compatibility : **API-forwarding, Cloud Rendering**

Intra-GPU : **Optimize the current framework**

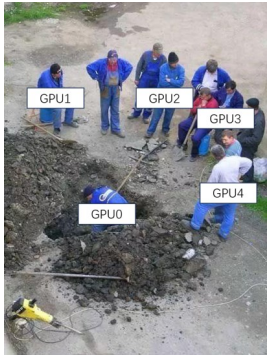


gVulkan goals

Ecosystem compatibility : API-forwarding, Cloud Rendering

Intra-GPU : Optimize the current framework

Inter-GPU : workload self-rebalancing

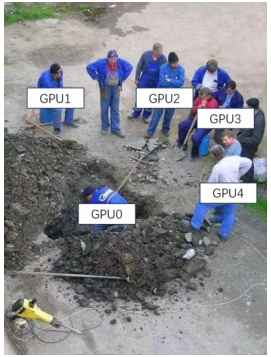


gVulkan goals

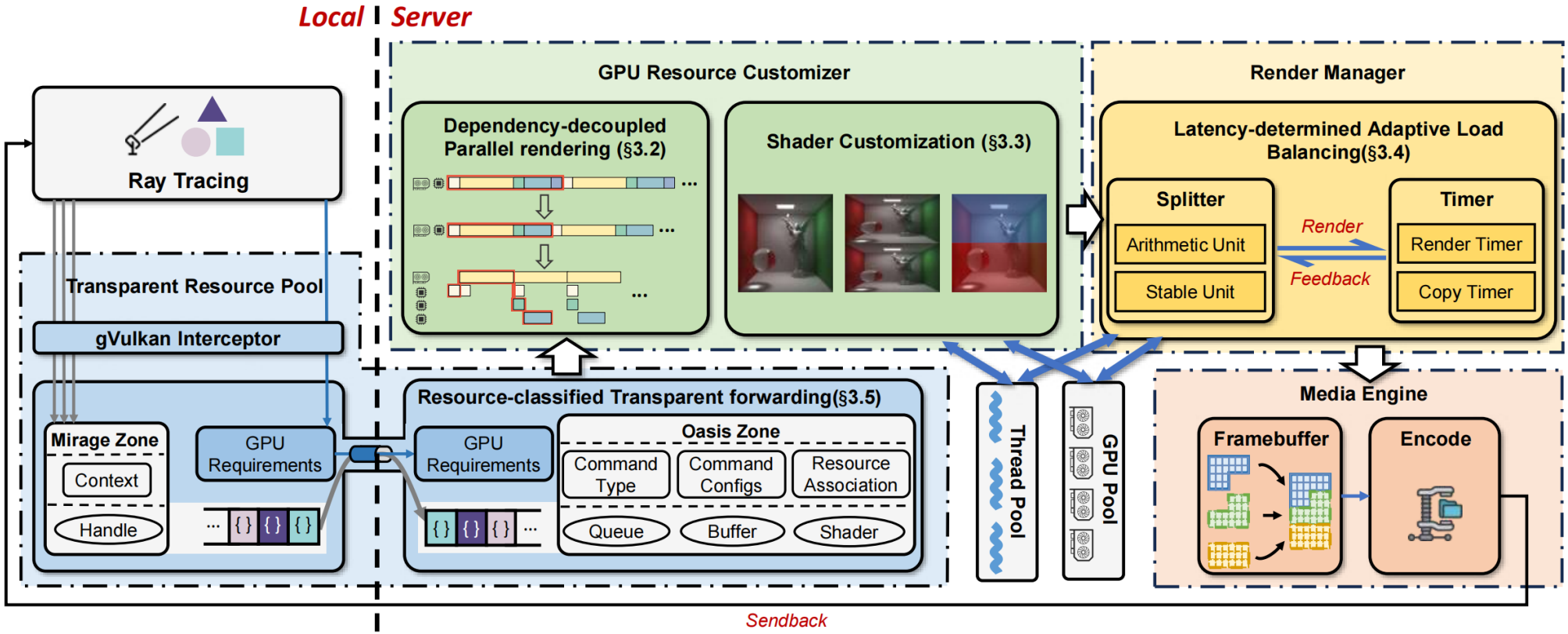
Ecosystem compatibility : API-forwarding, Cloud Rendering

Enables cloud-based code conversion from single-GPU to multi-GPU rendering tasks locally without modifying native Vulkan applications.

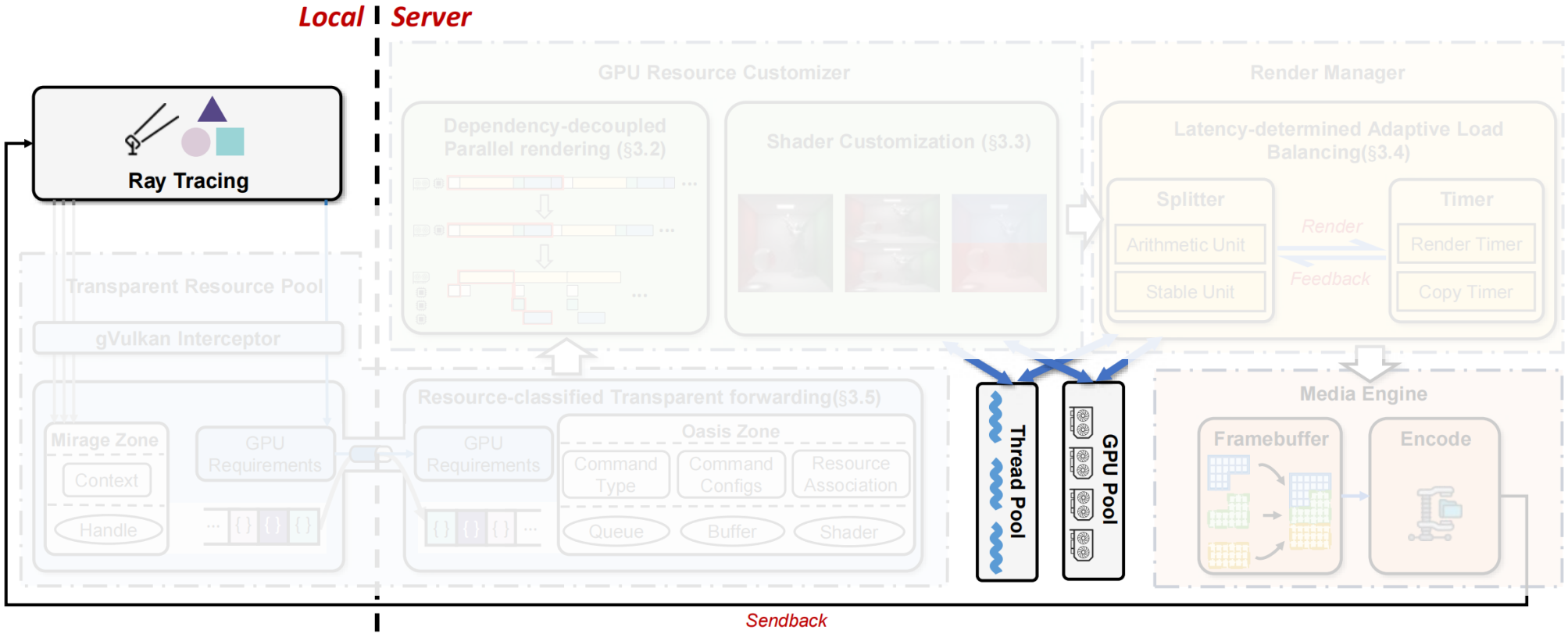
Inter-GPU : workload self-rebalancing



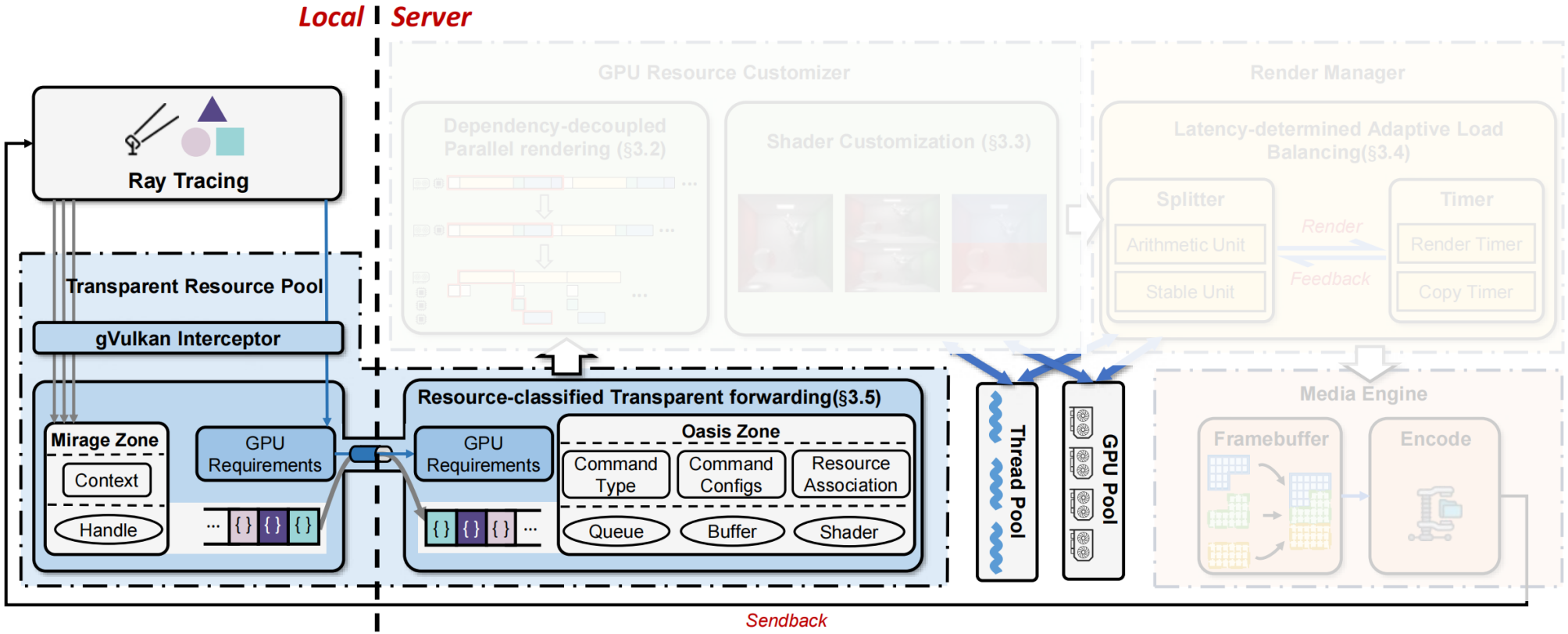
Architecture of gVulkan



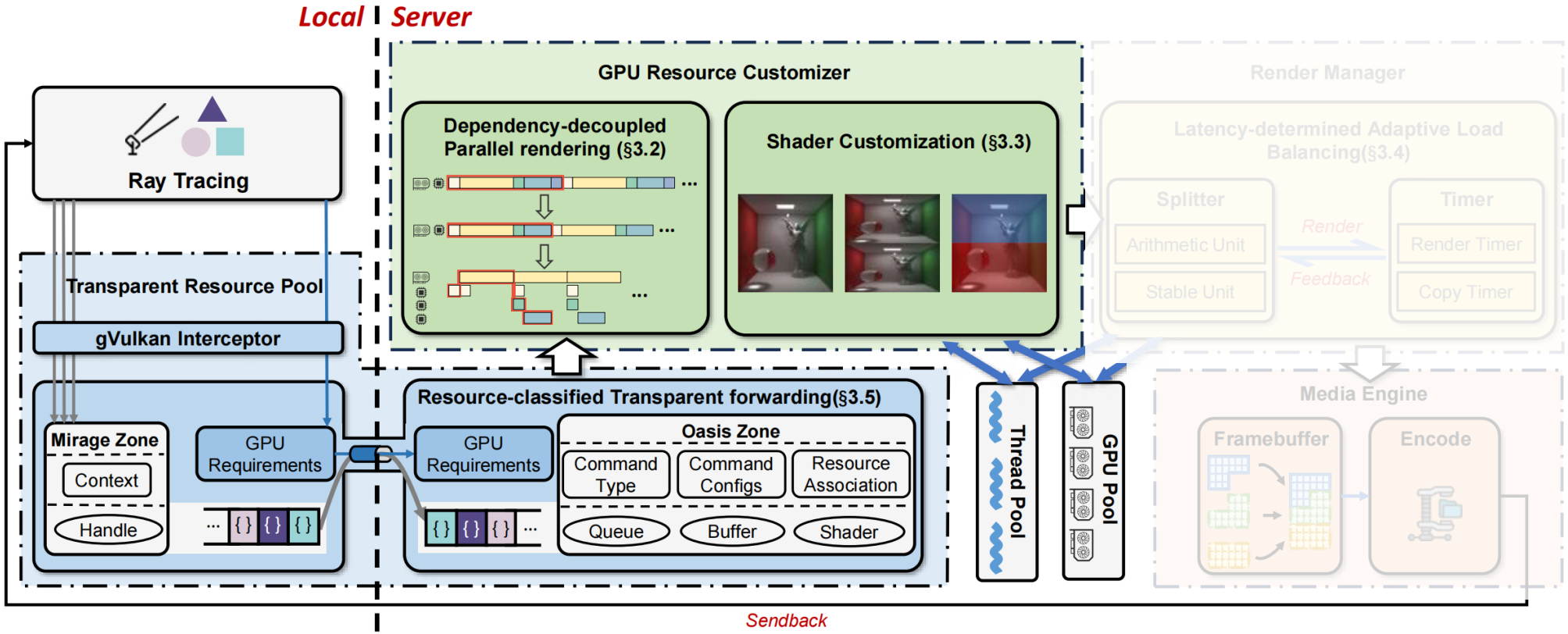
Architecture of gVulkan



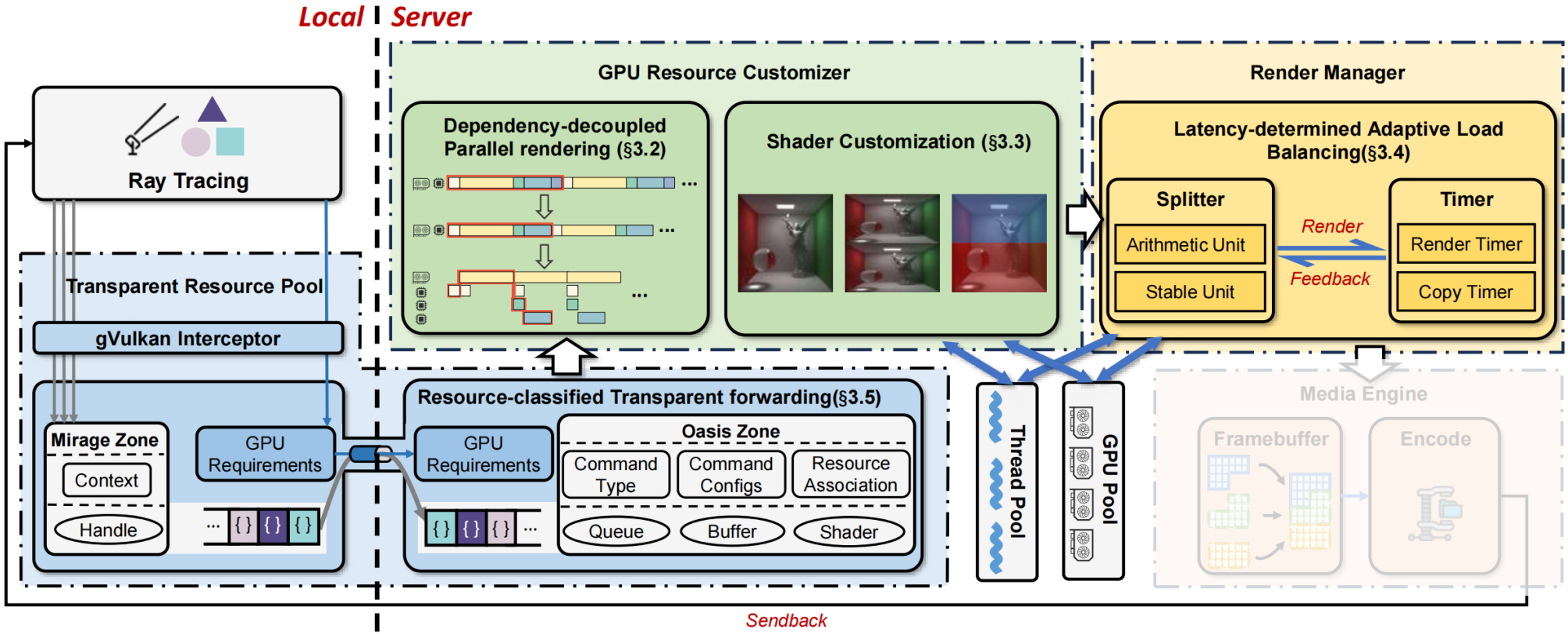
Architecture of gVulkan



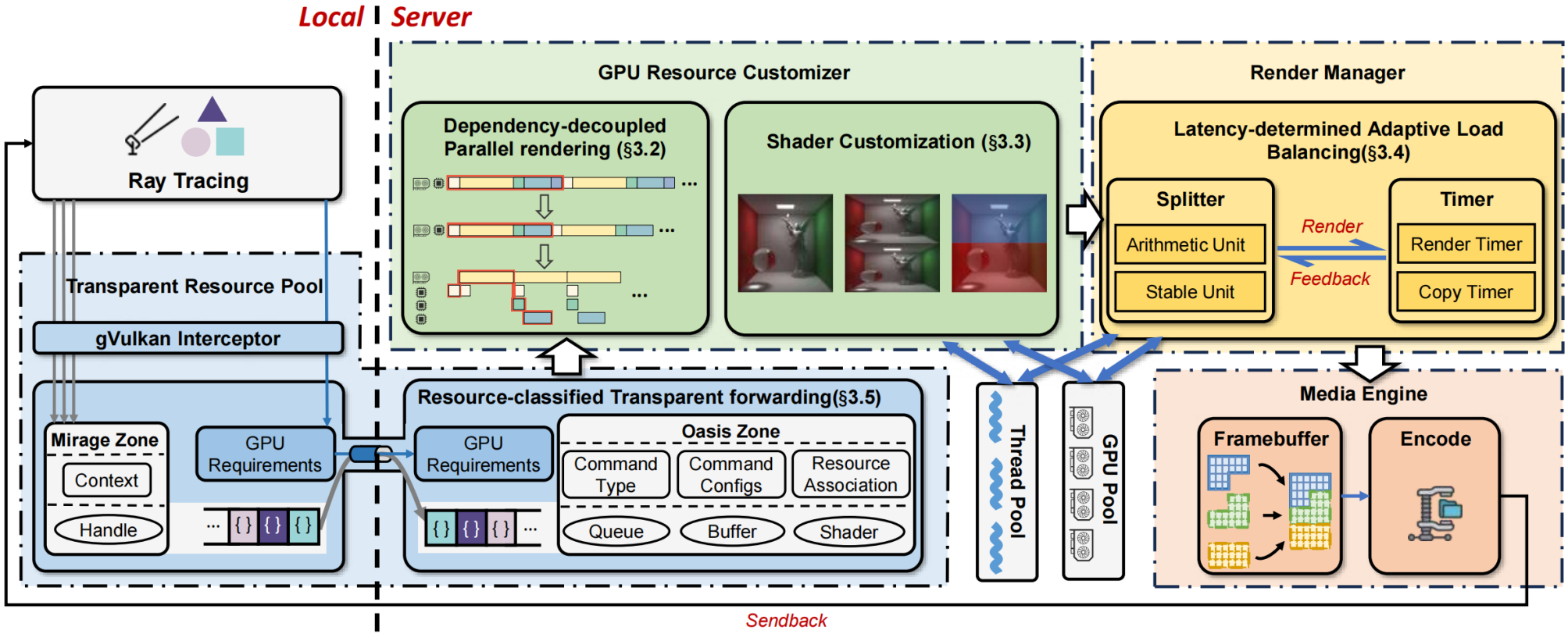
Architecture of gVulkan



Architecture of gVulkan

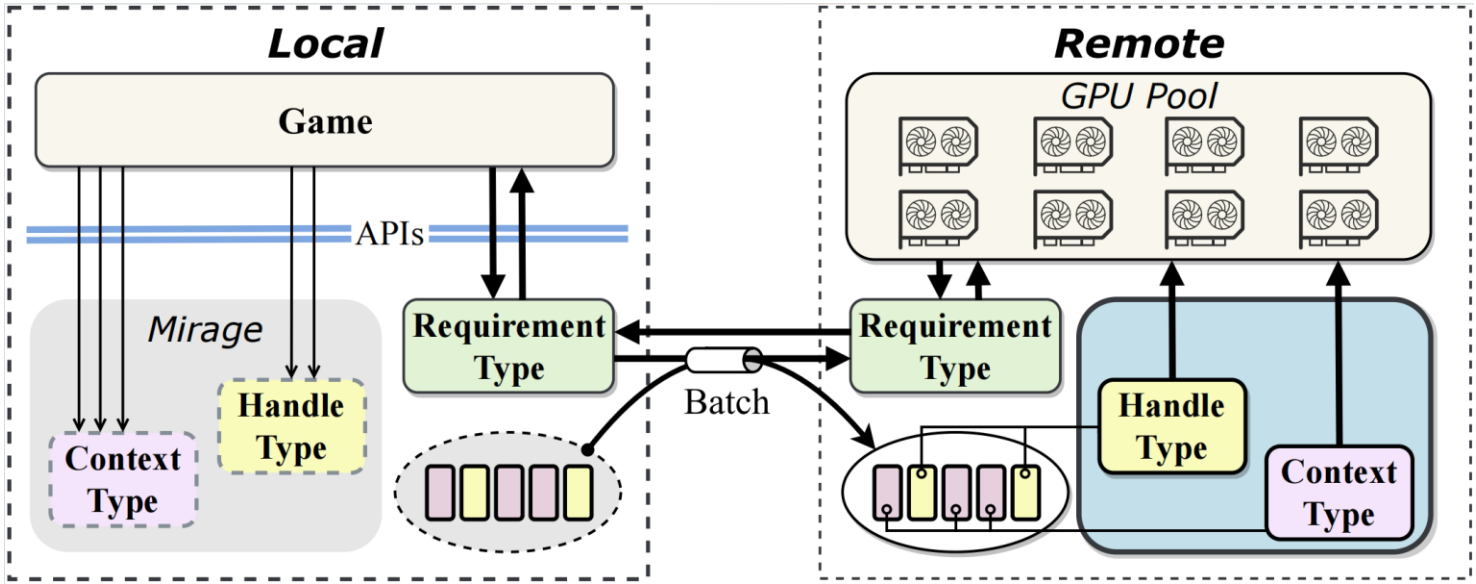


Architecture of gVulkan



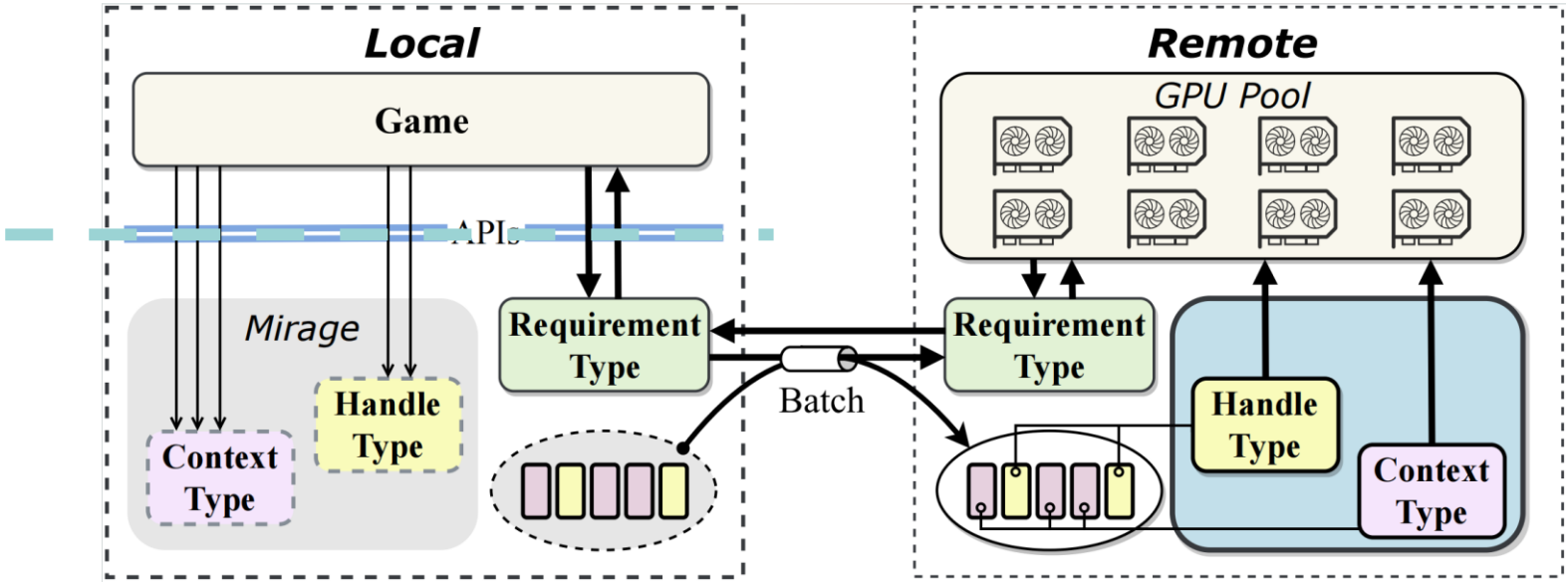
Resource-classified Transparent Forwarding

- Context Type : No need to change any parameters
- Handle Type : Need to change handle to real value
- Requirement Type : Need to return immediately



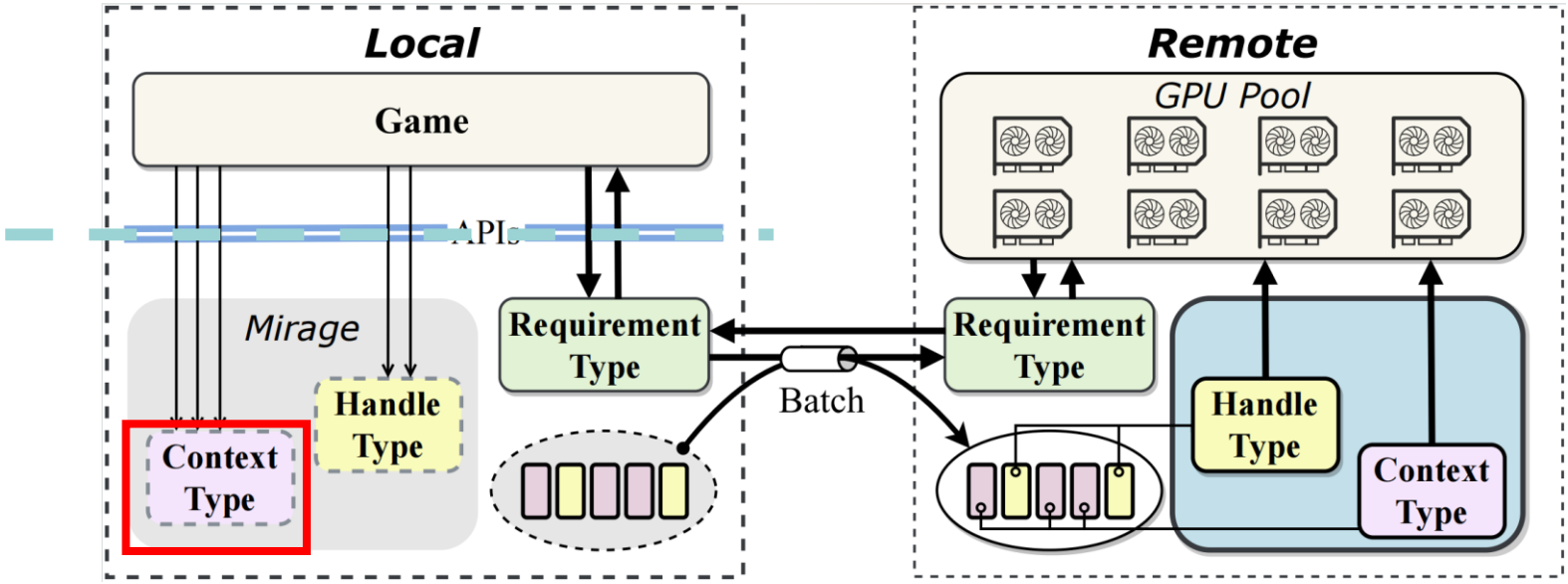
Resource-classified Transparent Forwarding

- Context Type : No need to change any parameters
- Handle Type : Need to change handle to real value
- Requirement Type : Need to return immediately



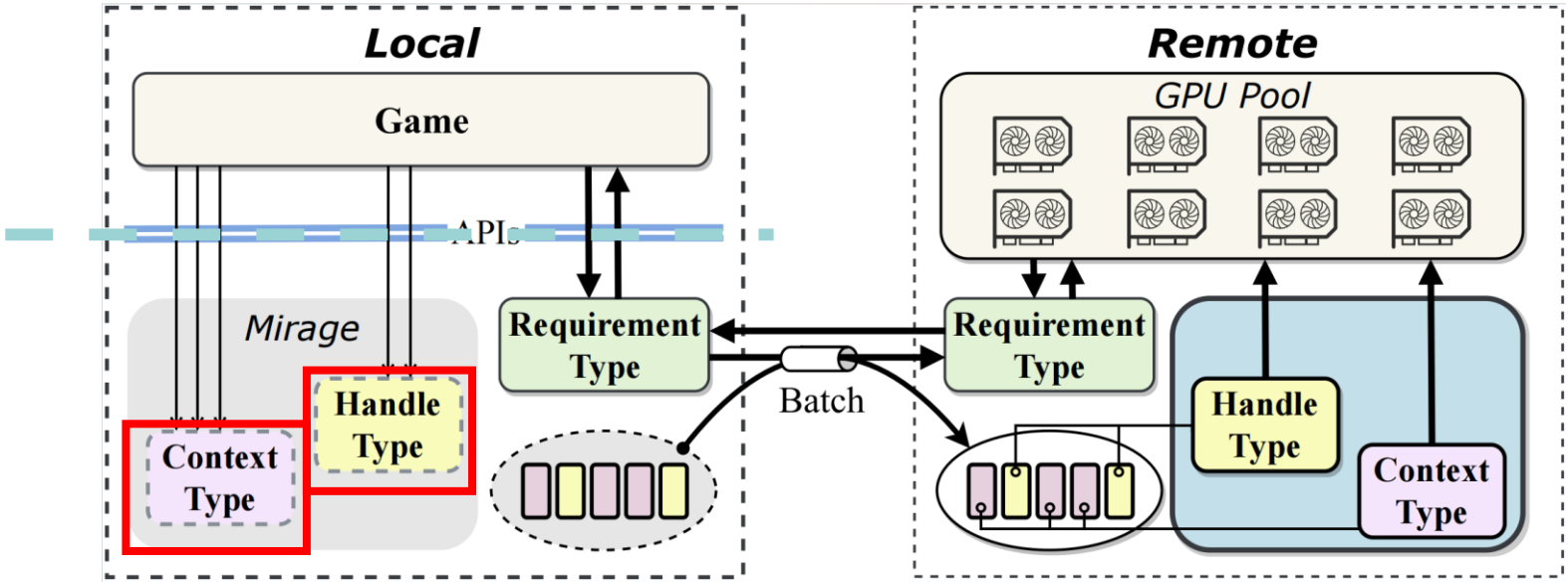
Resource-classified Transparent Forwarding

- Context Type : No need to change any parameters
- Handle Type : Need to change handle to real value
- Requirement Type : Need to return immediately



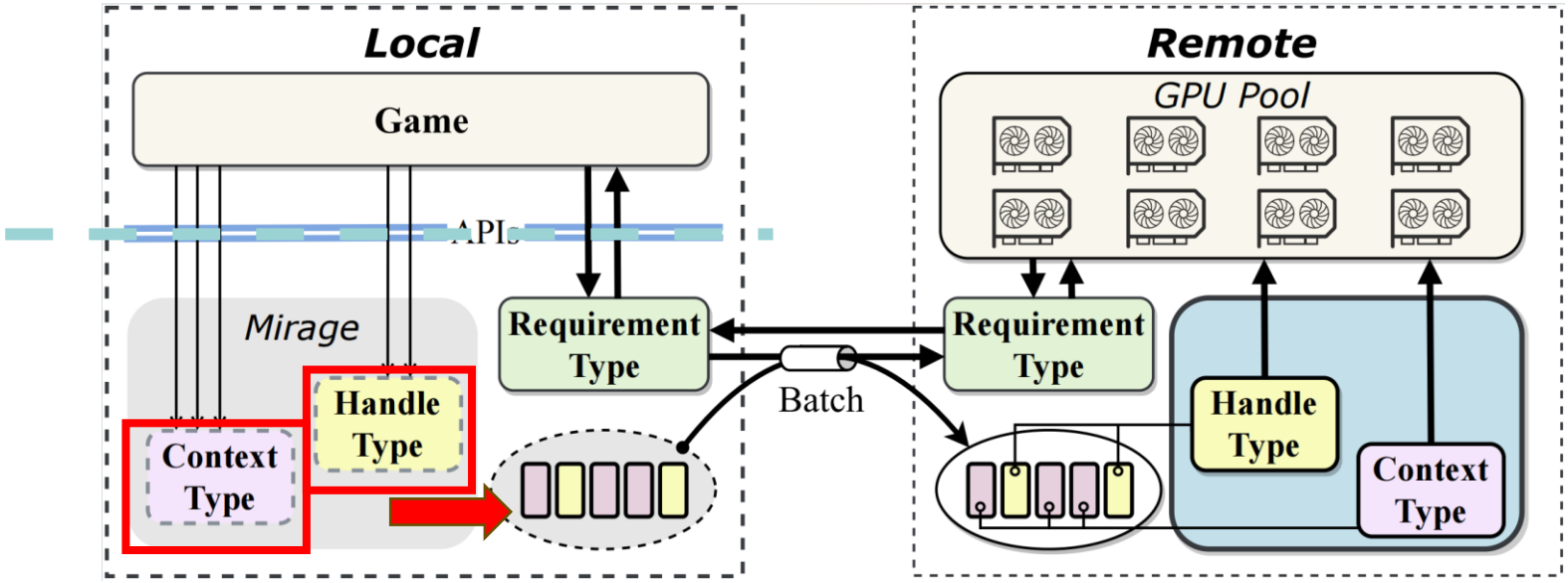
Resource-classified Transparent Forwarding

- Context Type : No need to change any parameters
- Handle Type : Need to change handle to real value
- Requirement Type : Need to return immediately



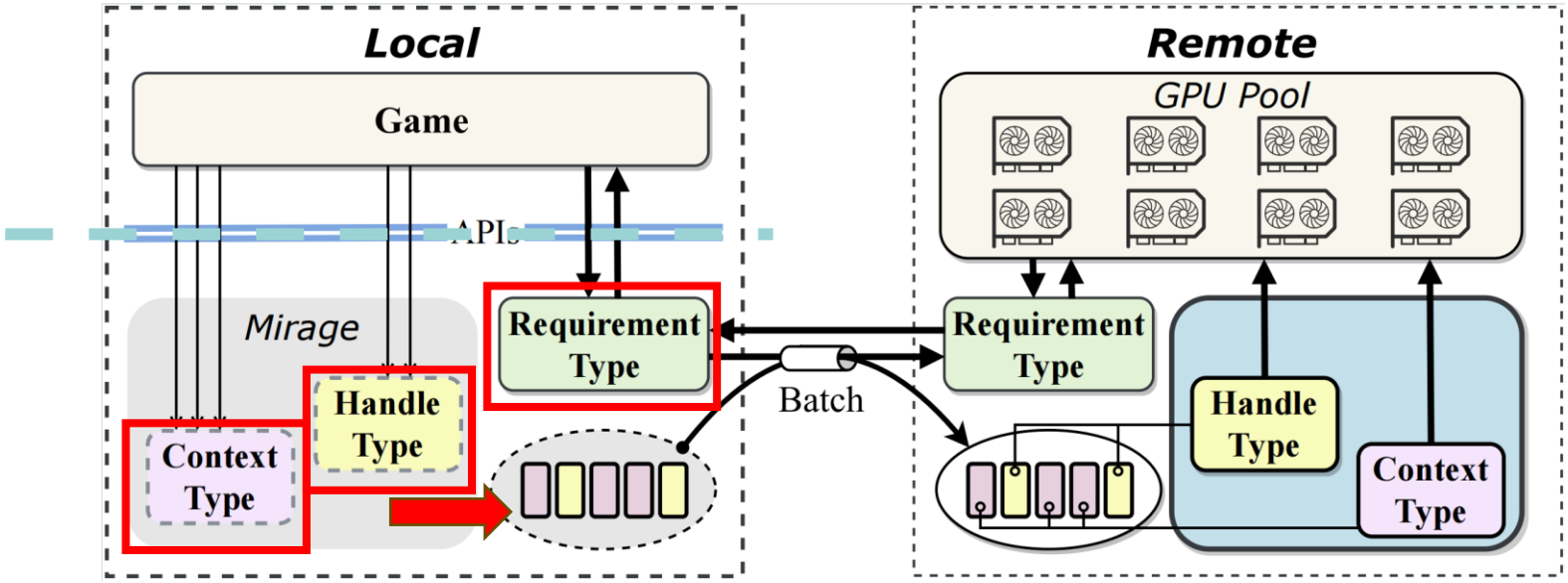
Resource-classified Transparent Forwarding

- Context Type : No need to change any parameters
- Handle Type : Need to change handle to real value
- Requirement Type : Need to return immediately



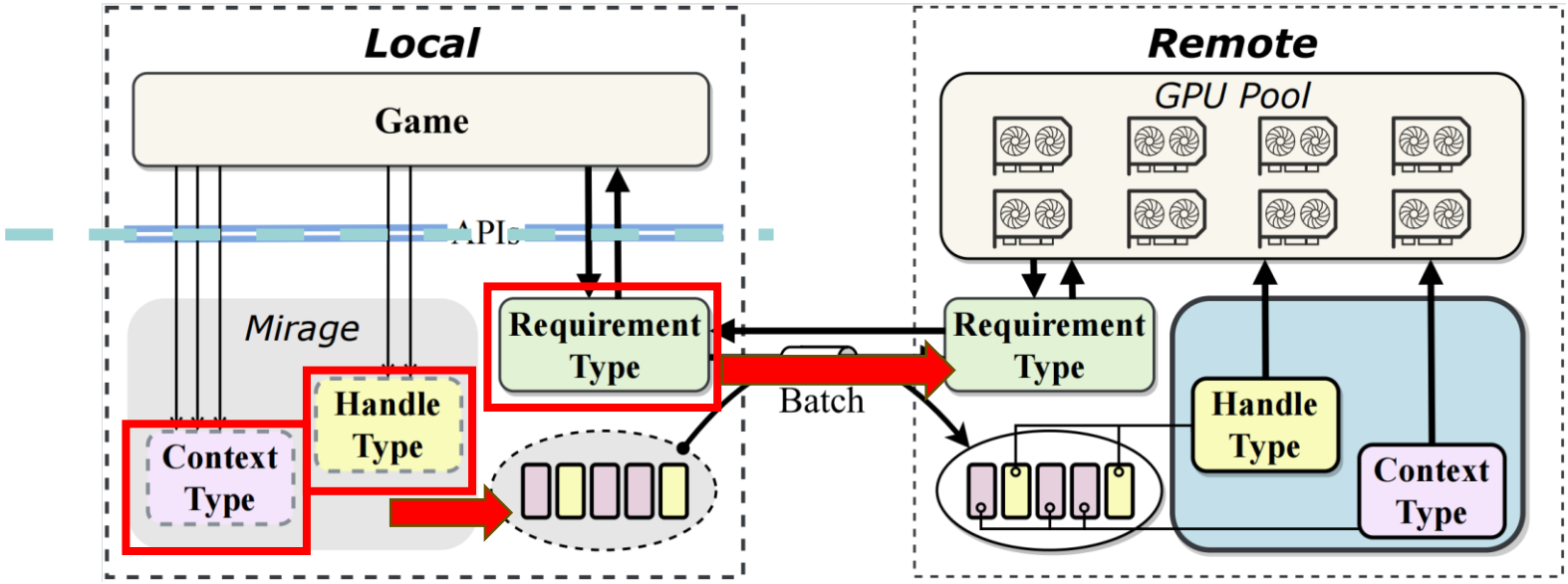
Resource-classified Transparent Forwarding

- Context Type : No need to change any parameters
- Handle Type : Need to change handle to real value
- Requirement Type : Need to return immediately



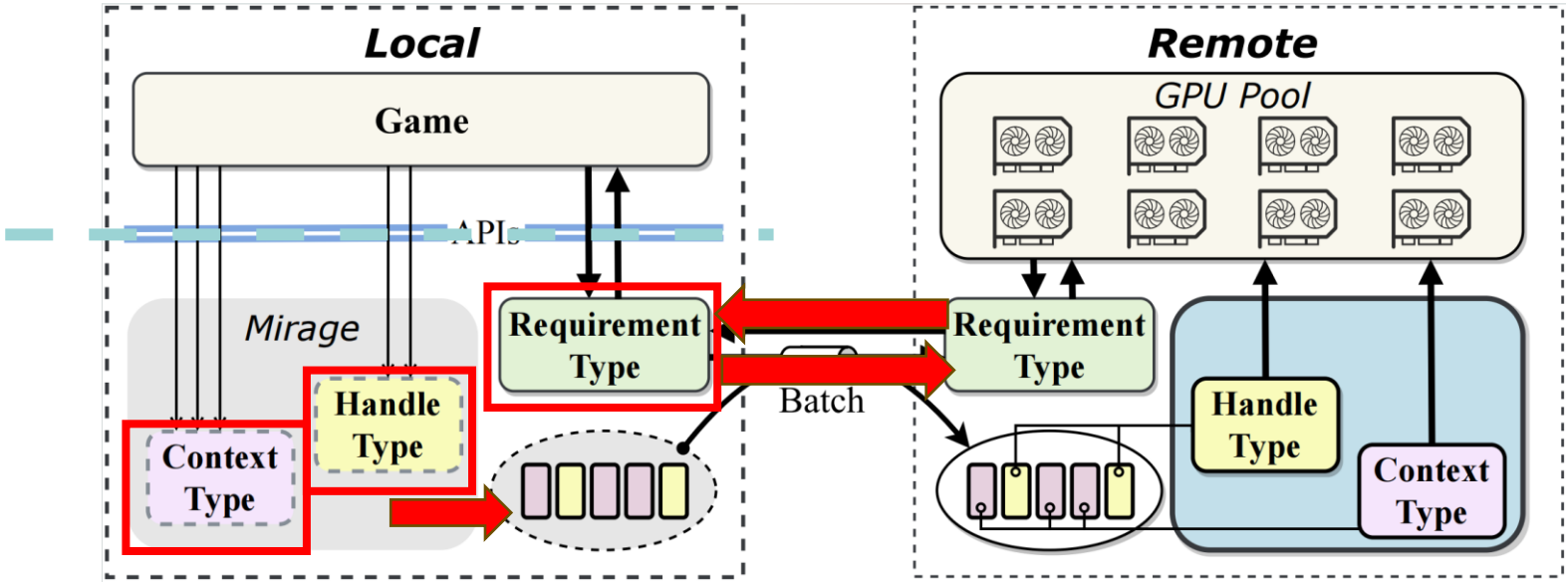
Resource-classified Transparent Forwarding

- Context Type : No need to change any parameters
- Handle Type : Need to change handle to real value
- Requirement Type : Need to return immediately



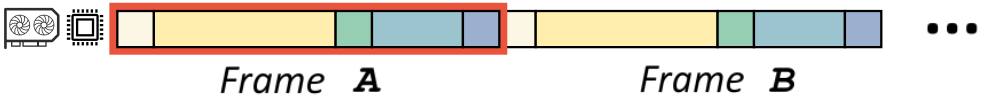
Resource-classified Transparent Forwarding

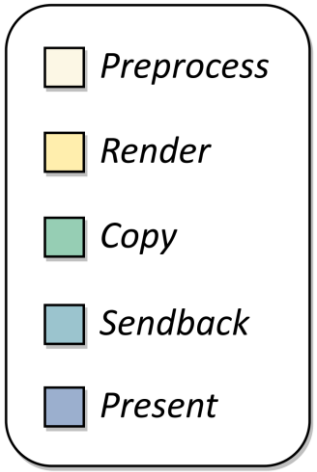
- Context Type : No need to change any parameters
- Handle Type : Need to change handle to real value
- Requirement Type : Need to return immediately



Dependency-decoupled parallel rendering

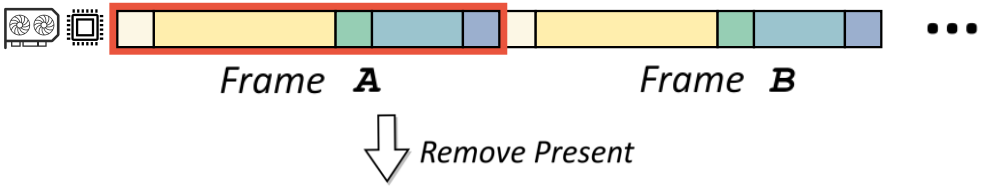
- Decouples the phases in a frame based on dependency relationships.
- Utilizes multi-threading to maximize GPU computation.

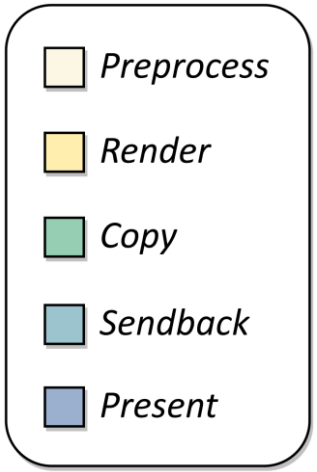




Dependency-decoupled parallel rendering

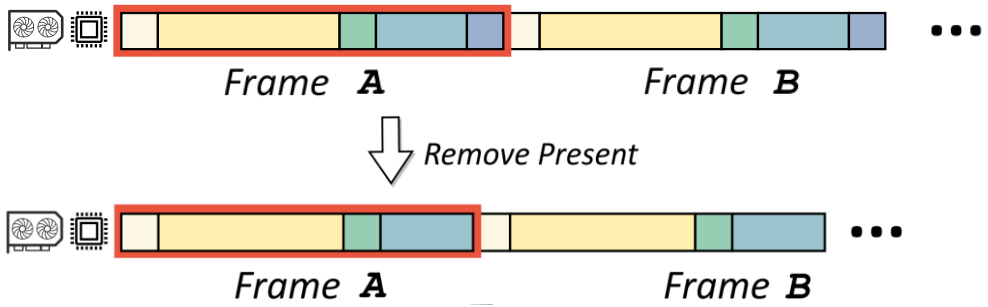
- Decouples the phases in a frame based on dependency relationships.
- Utilizes multi-threading to maximize GPU computation.

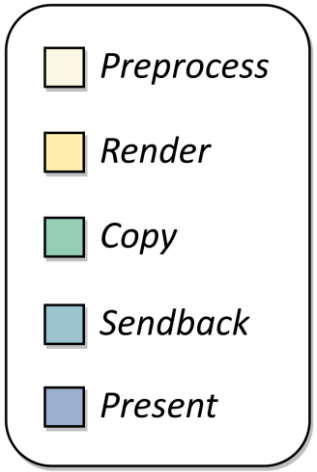




Dependency-decoupled parallel rendering

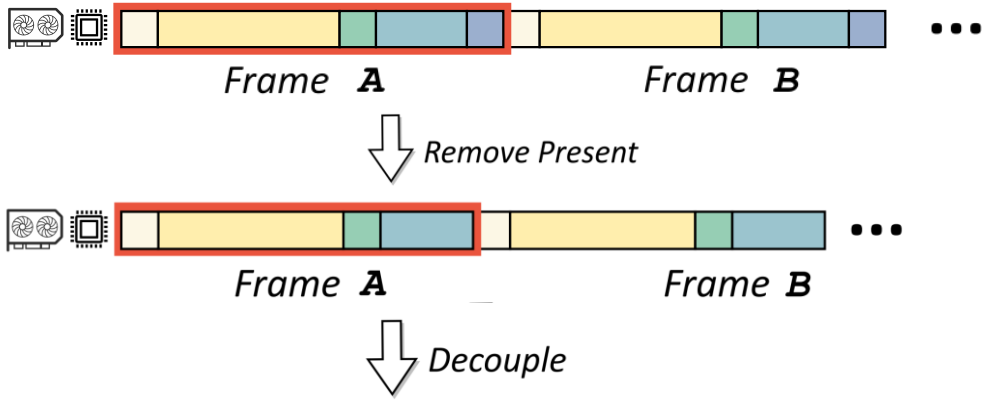
- Decouples the phases in a frame based on dependency relationships.
- Utilizes multi-threading to maximize GPU computation.

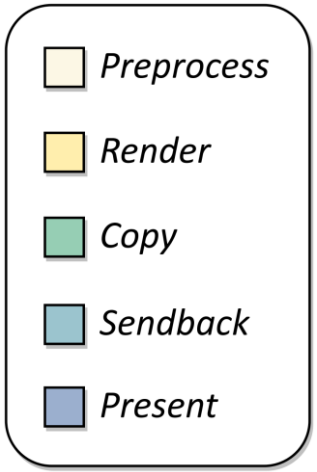




Dependency-decoupled parallel rendering

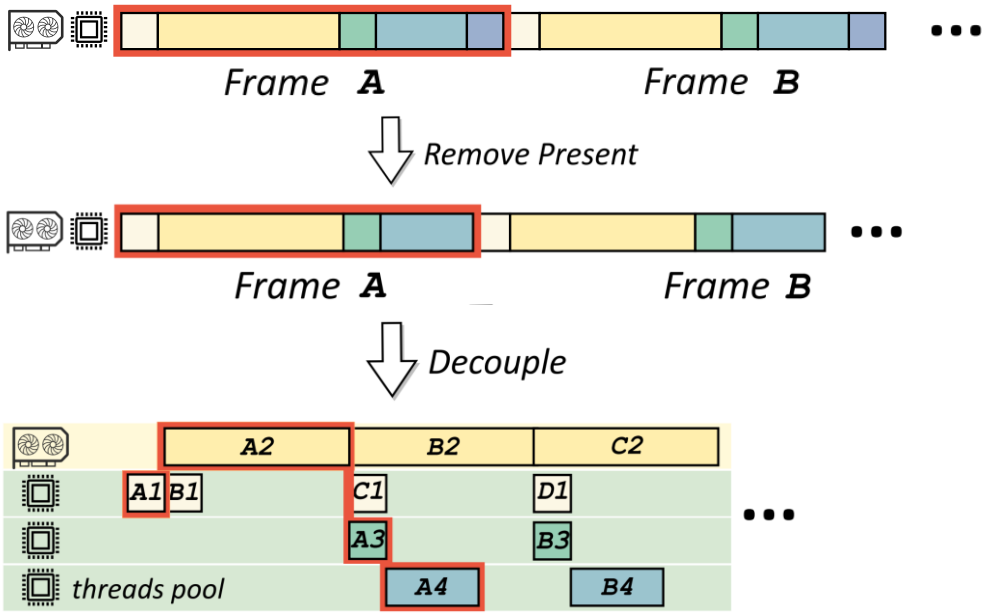
- Decouples the phases in a frame based on dependency relationships.
- Utilizes multi-threading to maximize GPU computation.

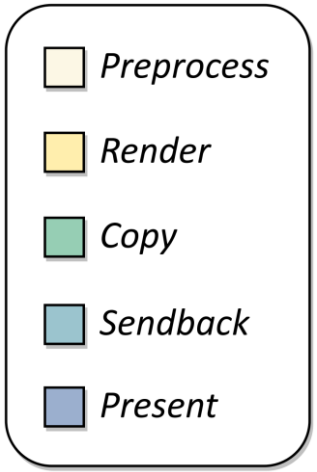




Dependency-decoupled parallel rendering

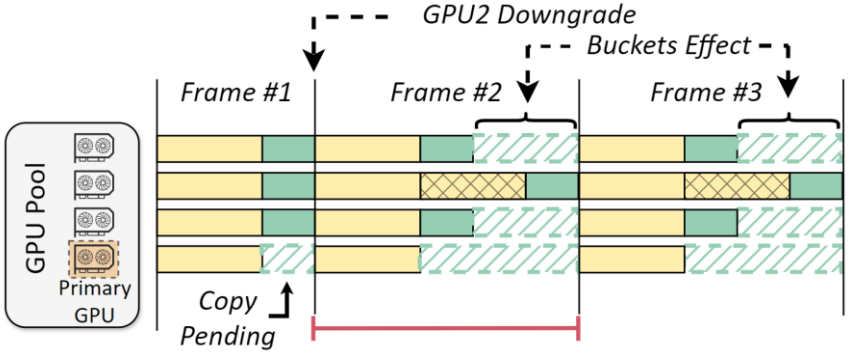
- Decouples the phases in a frame based on dependency relationships.
- Utilizes multi-threading to maximize GPU computation.

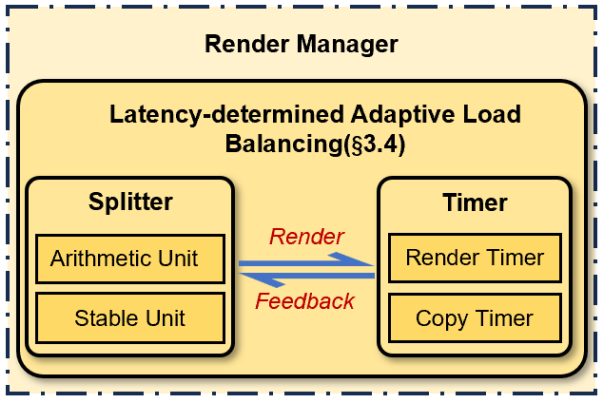


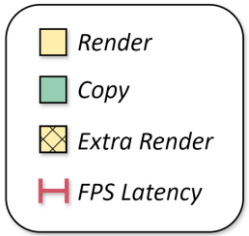


Latency-determined adaptive load balancing

- Obtaining the rendering and copying latency of GPUs through the Timer.
- Calculating the optimal allocation scheme based on the algorithm.
- Reducing Jitter with stable unit.

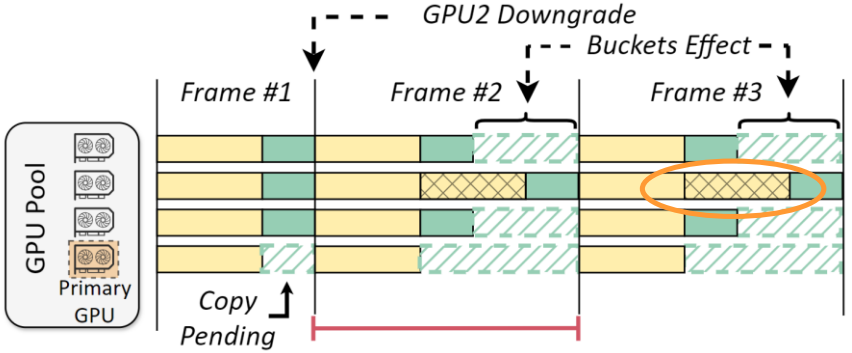


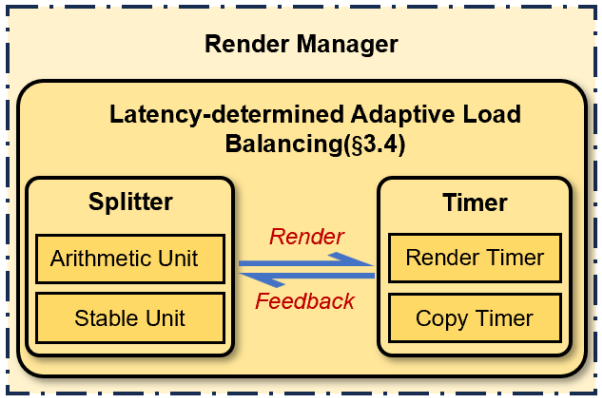


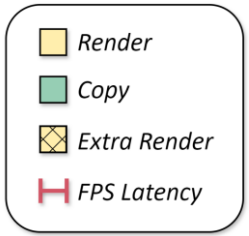


Latency-determined adaptive load balancing

- Obtaining the rendering and copying latency of GPUs through the Timer.
- Calculating the optimal allocation scheme based on the algorithm.
- Reducing Jitter with stable unit.

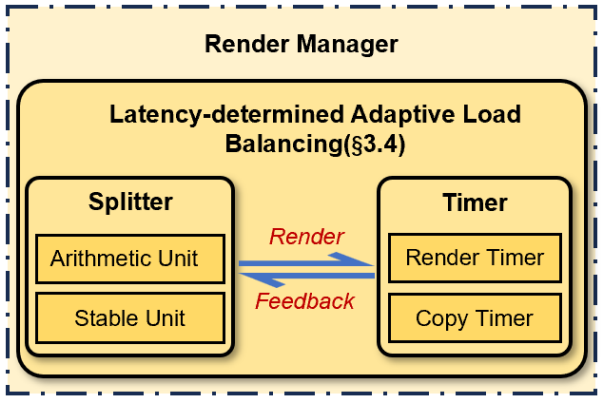


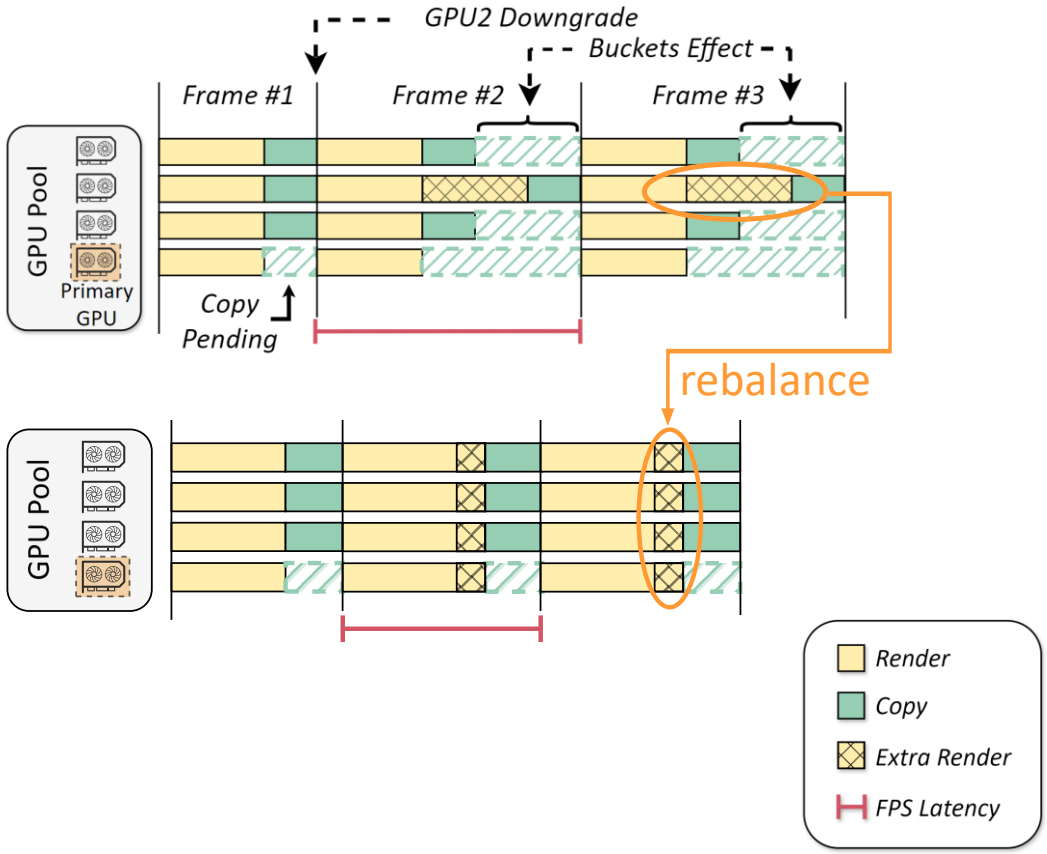




Latency-determined adaptive load balancing

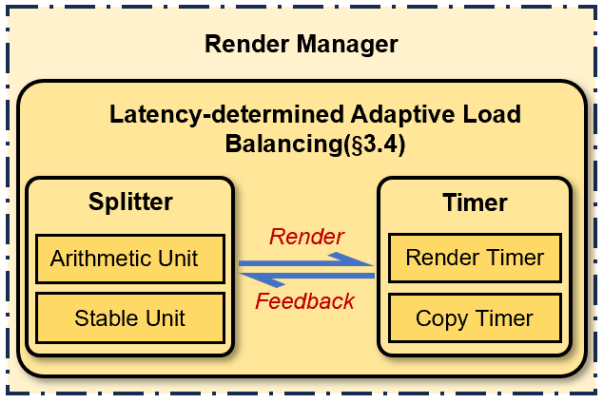
- Obtaining the rendering and copying latency of GPUs through the Timer.
- Calculating the optimal allocation scheme based on the algorithm.
- Reducing Jitter with stable unit.

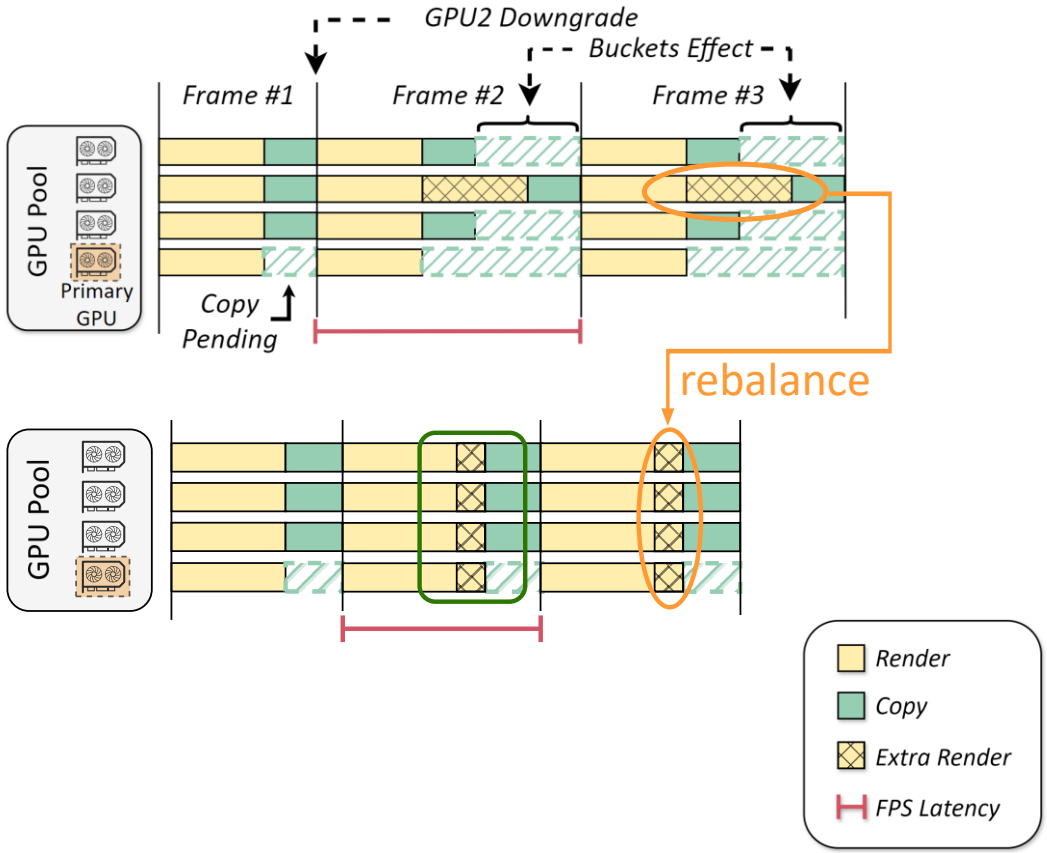




Latency-determined adaptive load balancing

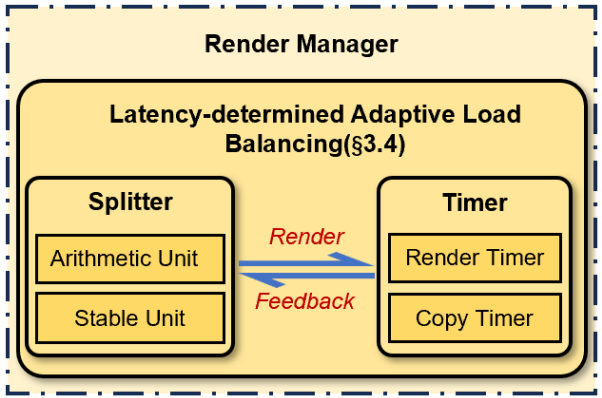
- Obtaining the rendering and copying latency of GPUs through the Timer.
- Calculating the optimal allocation scheme based on the algorithm.
- Reducing Jitter with stable unit.

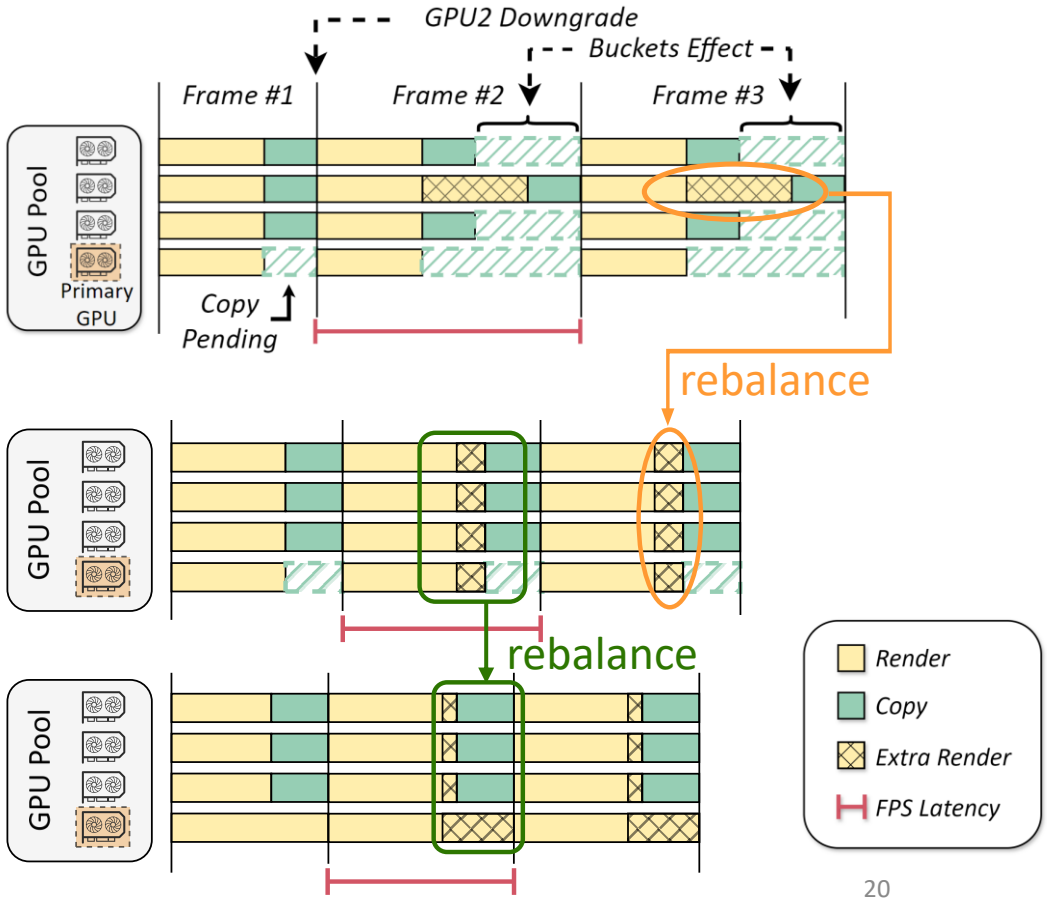




Latency-determined adaptive load balancing

- Obtaining the rendering and copying latency of GPUs through the Timer.
- Calculating the optimal allocation scheme based on the algorithm.
- Reducing Jitter with stable unit.





Ecosystem compatibility： improvement in FPS, image quality

Intra-GPU： performance of gVulkan

Inter-GPU: dynamic self-rebalancing

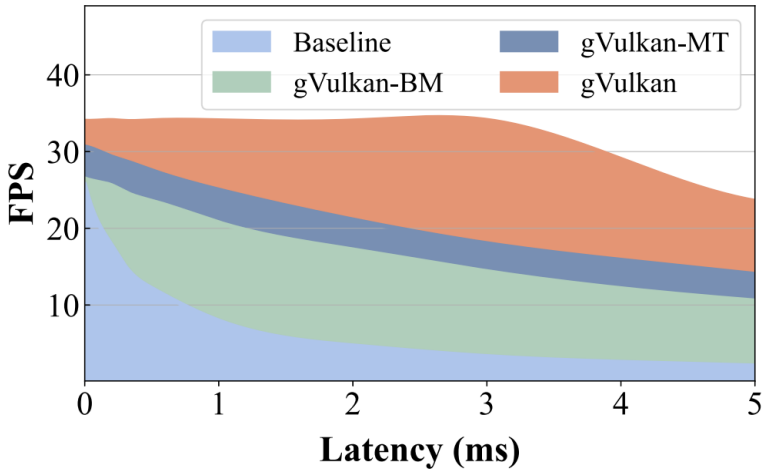
Evaluation

Ecosystem compatibility

- gVulkan reduces the impact of network latency on FPS.
- gVulkan can improve the rendering rate while maintaining image quality.

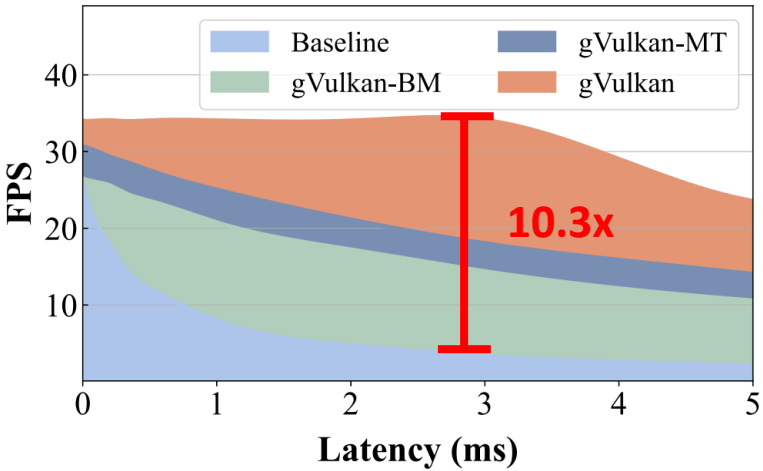
Ecosystem compatibility

- gVulkan reduces the impact of network latency on FPS.
- gVulkan can improve the rendering rate while maintaining image quality.



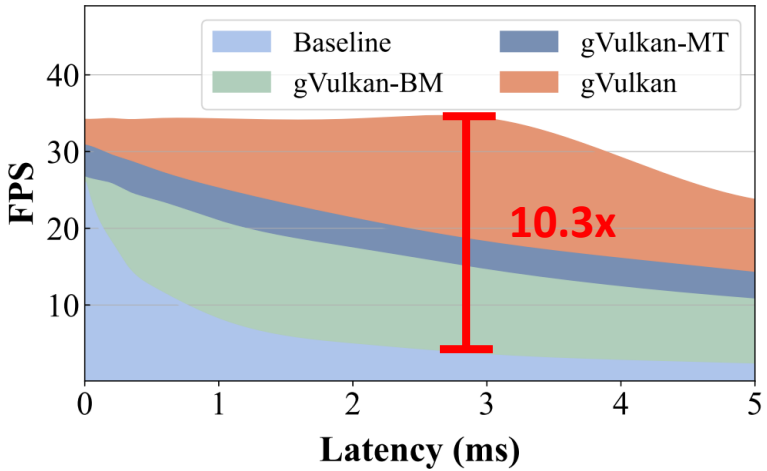
Ecosystem compatibility

- gVulkan reduces the impact of network latency on FPS.
- gVulkan can improve the rendering rate while maintaining image quality.



Ecosystem compatibility

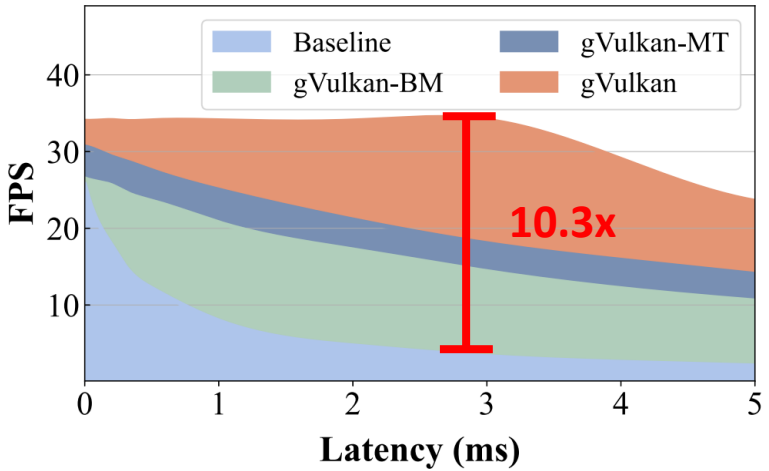
- gVulkan reduces the impact of network latency on FPS.
- gVulkan can improve the rendering rate while maintaining image quality.



	360P	480P	720P	1080P
Outdoor-Simple	54.60	54.63	54.68	54.68
Outdoor-Lucy	41.64	41.70	42.06	42.07
Cornell-Simple	30.25	30.25	31.52	38.78
Cornell-Lucy	37.11	37.14	38.41	38.26

Ecosystem compatibility

- gVulkan reduces the impact of network latency on FPS.
- gVulkan can improve the rendering rate while maintaining image quality.

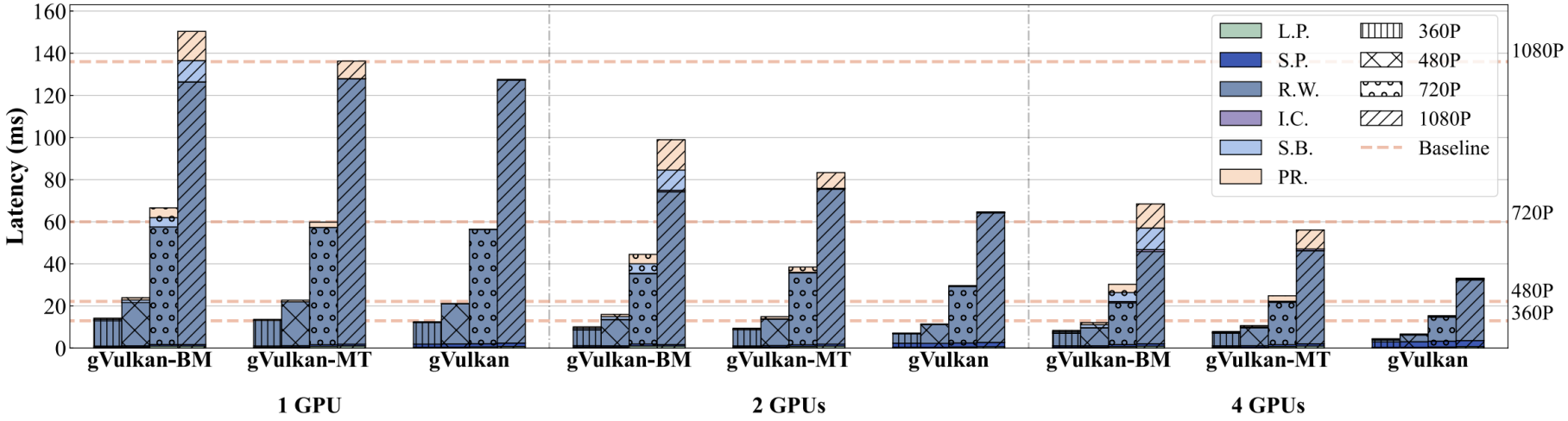


	360P	480P	720P	1080P
Outdoor-Simple	54.60	54.63	54.68	54.68
Outdoor-Lucy	41.64	41.70	42.06	42.07
Cornell-Simple	30.25	30.25	31.52	38.78
Cornell-Lucy	37.11	37.14	38.41	38.26

All PSNR >30

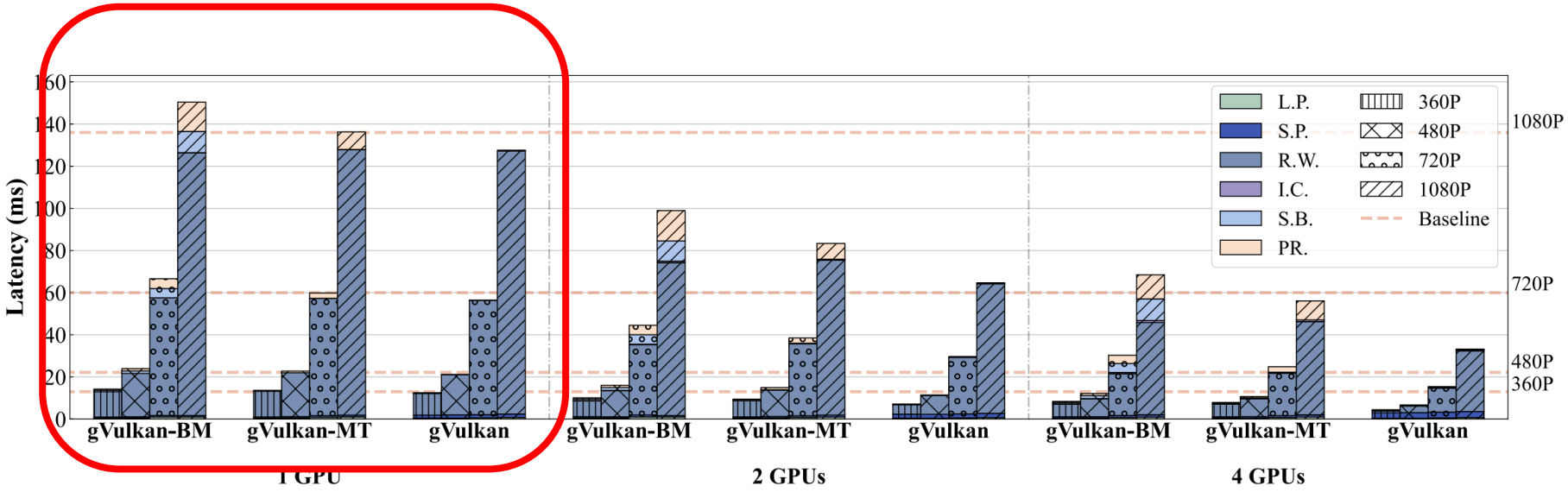
Performance of gVulkan

- While the local application fails to reach the 30 FPS threshold, gVulkan successfully achieved the QoS guarantee of 60 FPS with 4 GPUs.
- The average speedup for the four scenes under the gVulkan with 4 GPUs can reach 3.81.



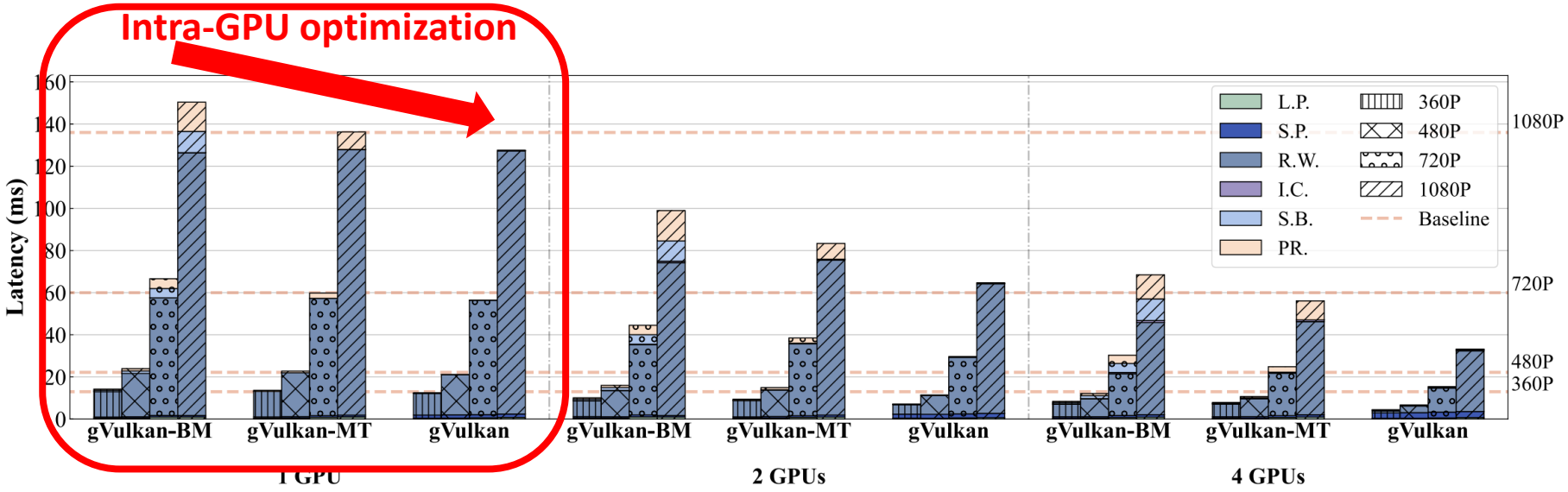
Performance of gVulkan

- While the local application fails to reach the 30 FPS threshold, gVulkan successfully achieved the QoS guarantee of 60 FPS with 4 GPUs.
- The average speedup for the four scenes under the gVulkan with 4 GPUs can reach 3.81.



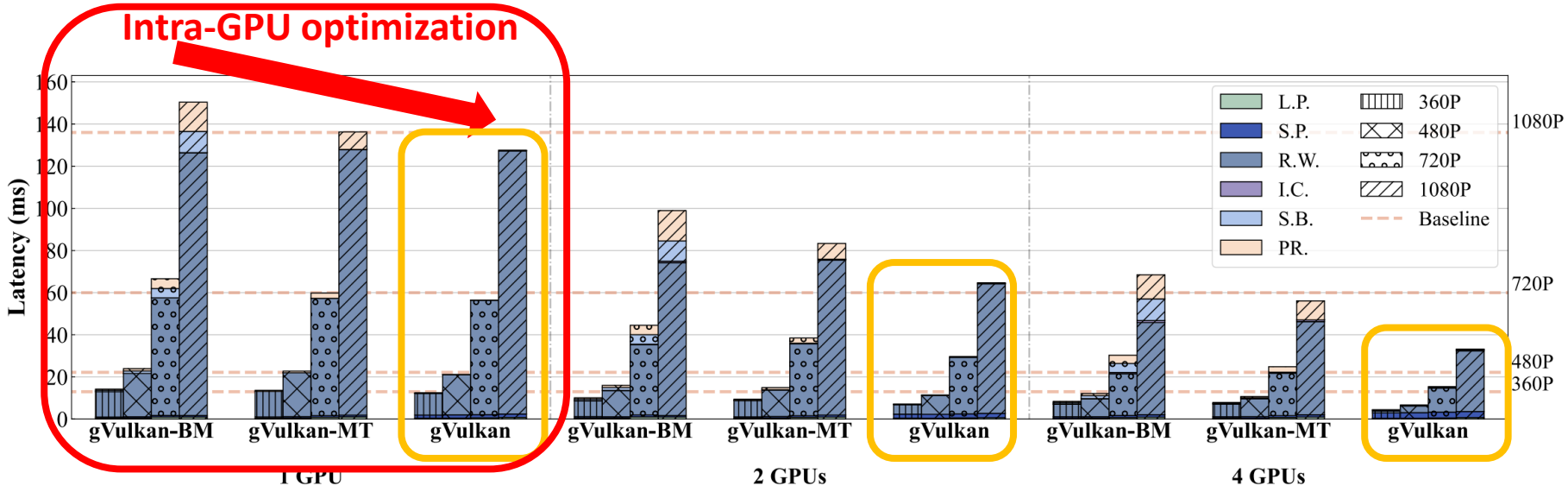
Performance of gVulkan

- While the local application fails to reach the 30 FPS threshold, gVulkan successfully achieved the QoS guarantee of 60 FPS with 4 GPUs.
- The average speedup for the four scenes under the gVulkan with 4 GPUs can reach 3.81.



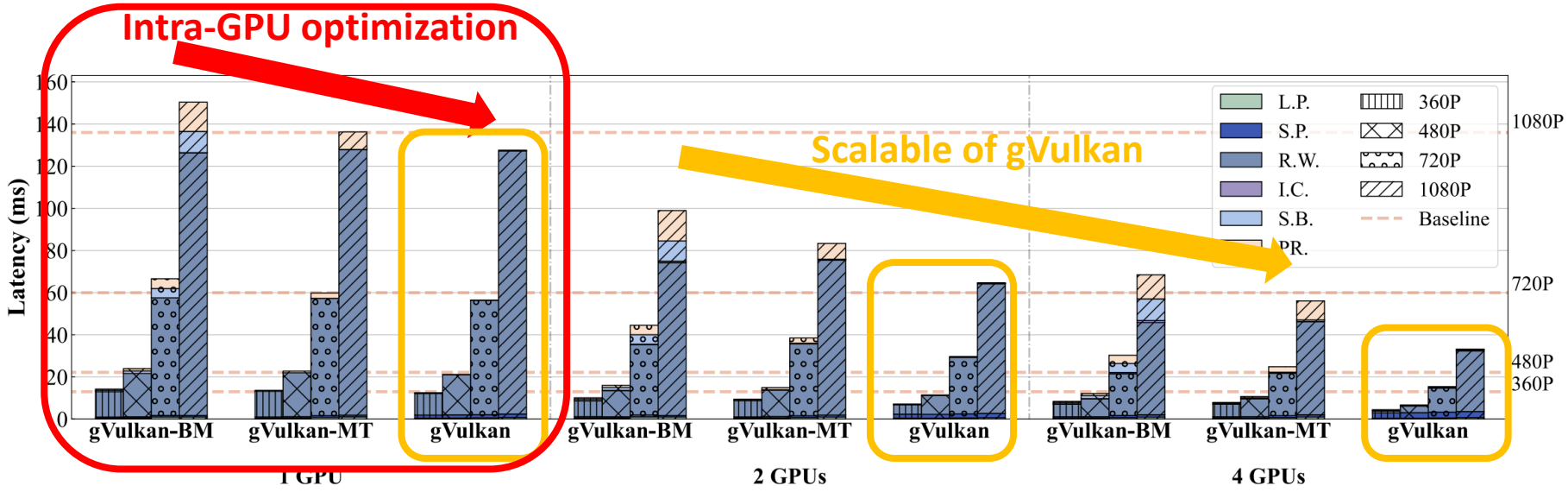
Performance of gVulkan

- While the local application fails to reach the 30 FPS threshold, gVulkan successfully achieved the QoS guarantee of 60 FPS with 4 GPUs.
- The average speedup for the four scenes under the gVulkan with 4 GPUs can reach 3.81.



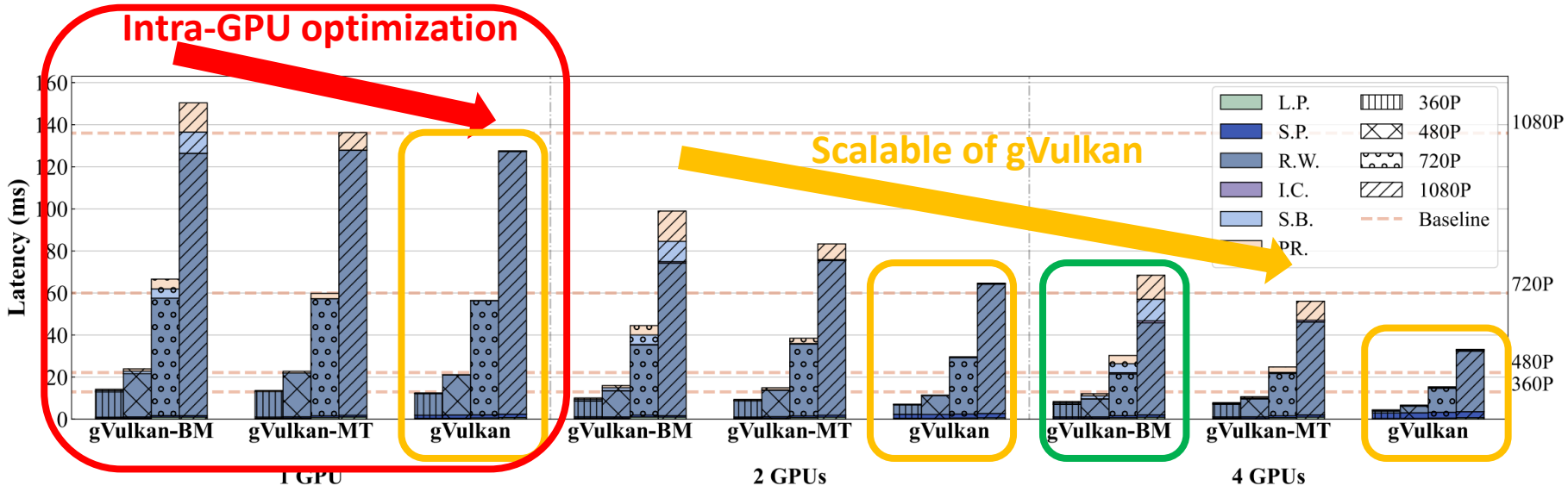
Performance of gVulkan

- While the local application fails to reach the 30 FPS threshold, gVulkan successfully achieved the QoS guarantee of 60 FPS with 4 GPUs.
- The average speedup for the four scenes under the gVulkan with 4 GPUs can reach 3.81.



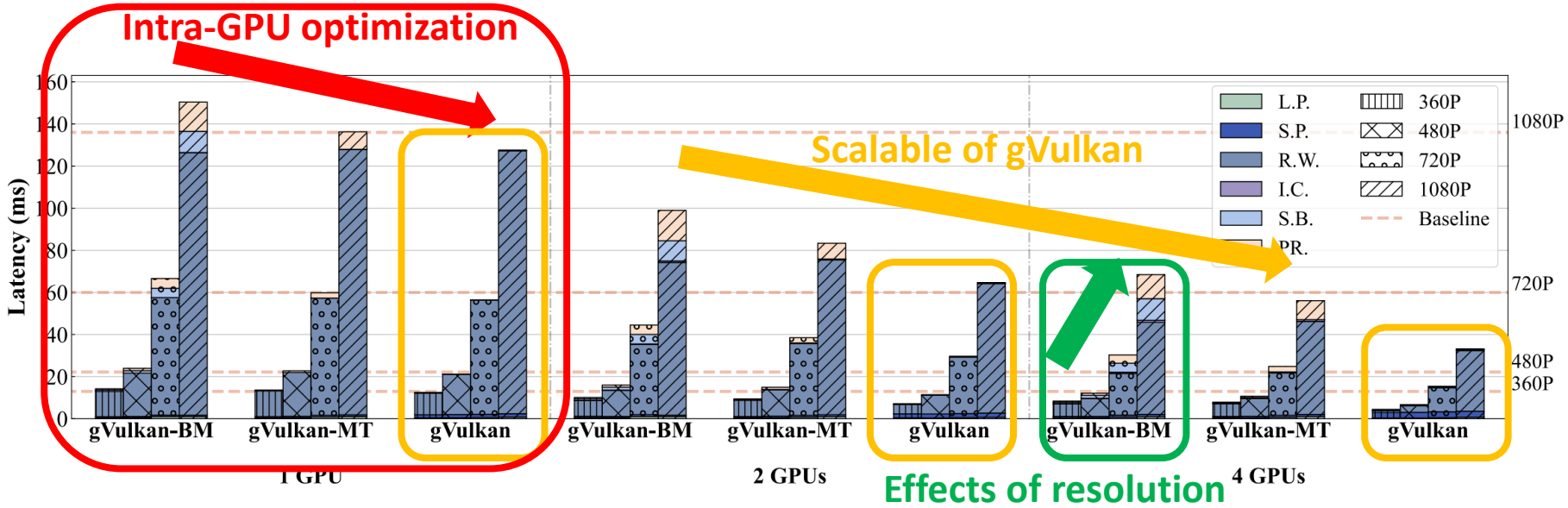
Performance of gVulkan

- While the local application fails to reach the 30 FPS threshold, gVulkan successfully achieved the QoS guarantee of 60 FPS with 4 GPUs.
- The average speedup for the four scenes under the gVulkan with 4 GPUs can reach 3.81.



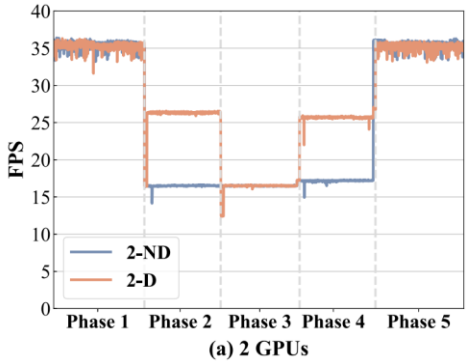
Performance of gVulkan

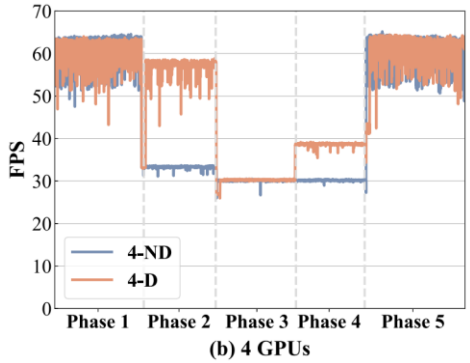
- While the local application fails to reach the 30 FPS threshold, gVulkan successfully achieved the QoS guarantee of 60 FPS with 4 GPUs.
- The average speedup for the four scenes under the gVulkan with 4 GPUs can reach 3.81.

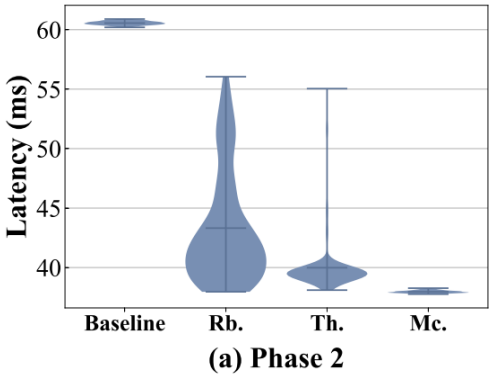


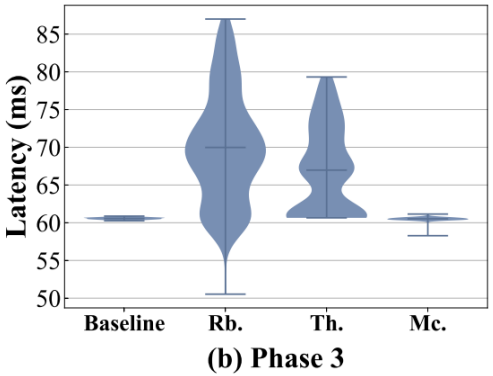
Dynamic self-rebalancing

- Dynamic rebalancing significantly reduces maximum latency when the GPU's rendering power is not evenly distributed
- Stable units can effectively reduce fluctuations and latency.



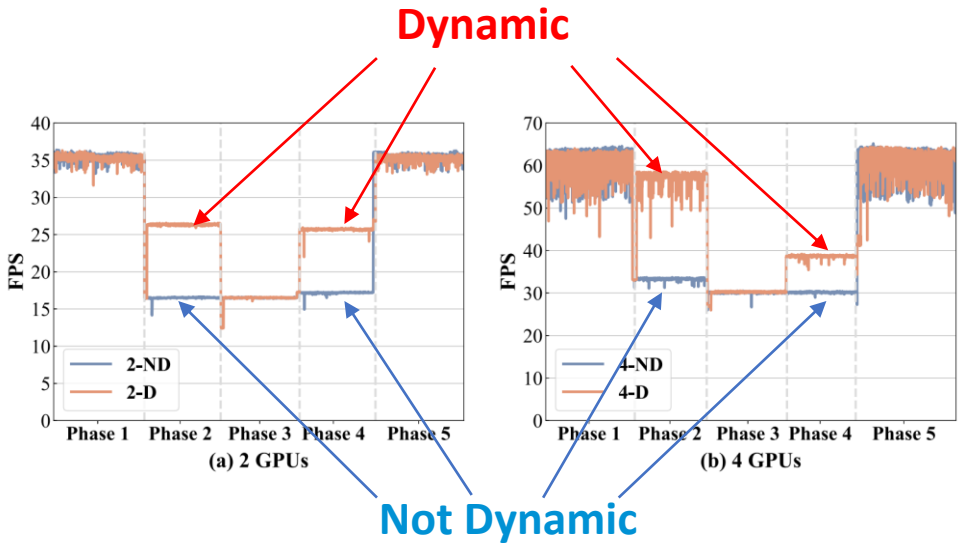


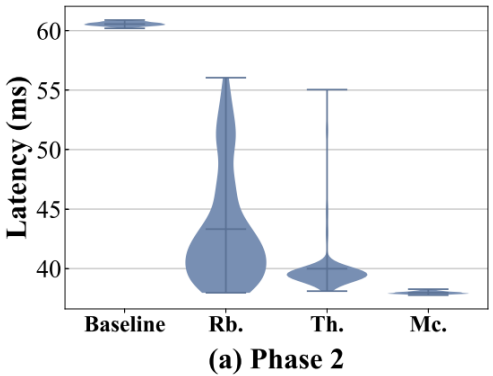


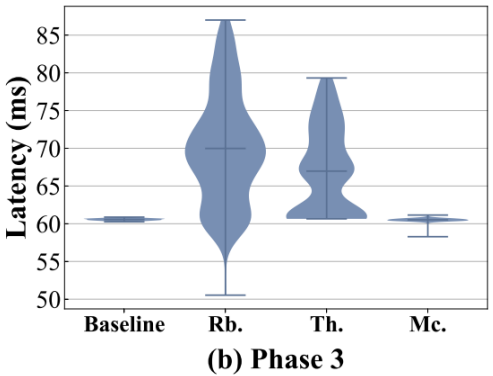


Dynamic self-rebalancing

- Dynamic rebalancing significantly reduces maximum latency when the GPU's rendering power is not evenly distributed
- Stable units can effectively reduce fluctuations and latency.

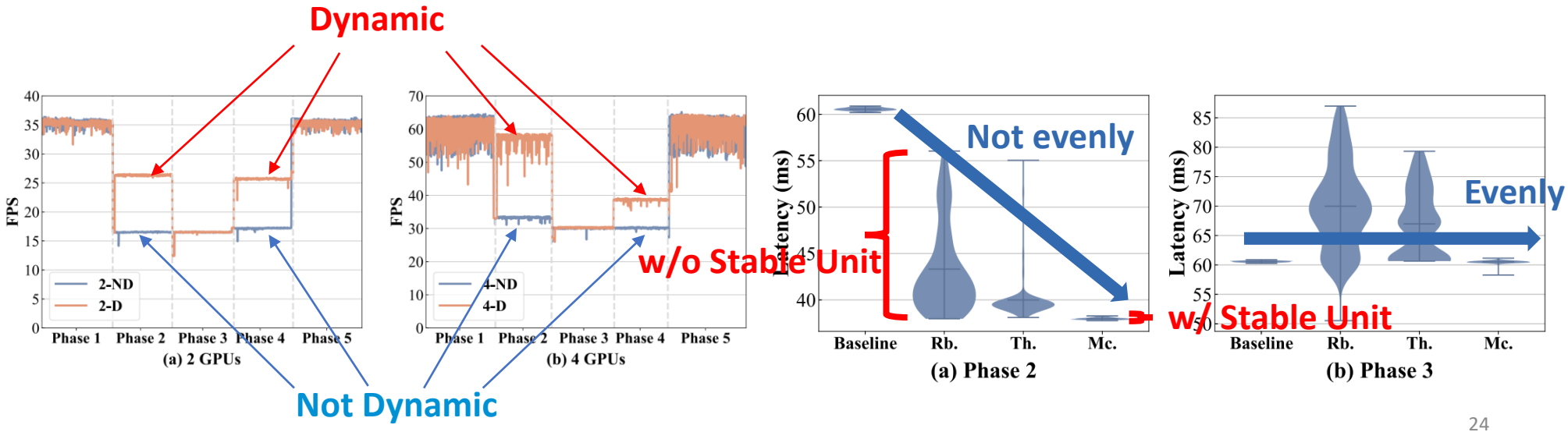






Dynamic self-rebalancing

- Dynamic rebalancing significantly reduces maximum latency when the GPU's rendering power is not evenly distributed
- Stable units can effectively reduce fluctuations and latency.



Conclusion

gVulkan is the first transparent multi-GPU acceleration rendering solution for Vulkan-based ray tracing rendering.

- A resource-classified transparent forwarding scheme.
- A latency-determined adaptive load balancing mechanism.
- A dependency-decoupled parallel rendering approach.

- gVulkan achieves good linearity with 3.81× speedup across 4 GPUs on average.

https://github.com/funnygyc/gVulkan-artifact

THANK YOU

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Q&A

USENIX ATC '24

Yicheng Gu, Yun Wang, Yunfan Sun, Yuxin Xiang,
Yufan Jiang, Xuyan Hu, Zhengwei Qi, Haibing Guan