CGI2022

Dynamic Load Balancing for Real-Time Multiview Path Tracing on Multi-GPU Architectures

Erwan Leria*, Markku Mäkitalo*, Julius Ikkala*, Pekka Jääskeläinen*

**Tampere University, Finland*

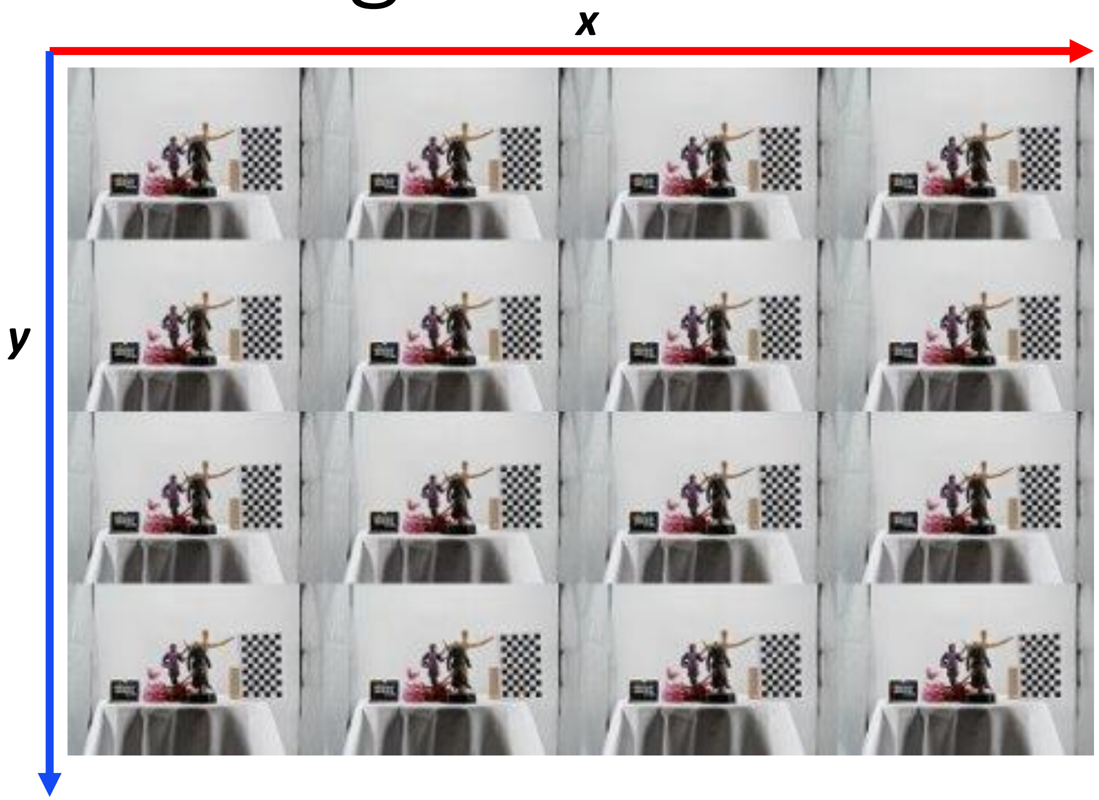
Virtual reality and Graphics Architectures (https://tuni.fi/vga/)

Presenter: **Erwan Leria** (*erwan.leria@tuni.fi*)

Tampere University

Introduction: Multiview technologies

- What has been done:
  - Image-based novel view synthesis
  - Light fields captured with a camera
- What has **not** been done:
  - Real-time photorealistic rendering of multiple views from 3D models

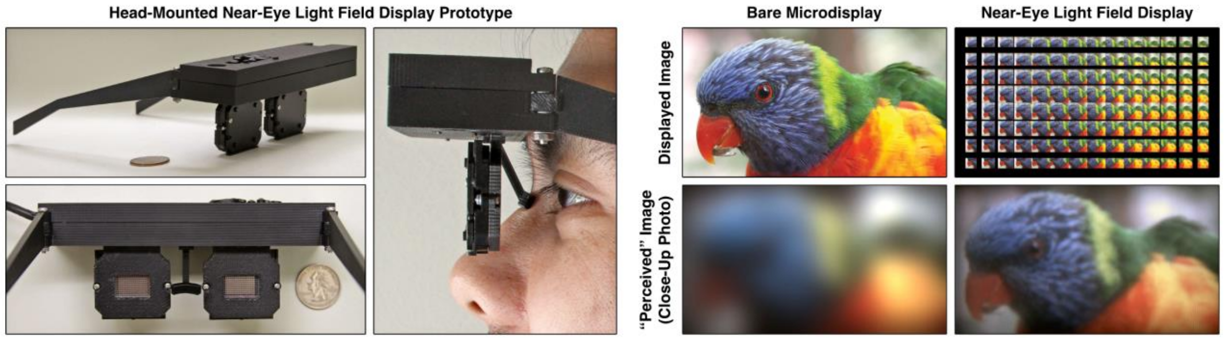


Full parallax 4x4 light field captured with an RGB camera

Applications of Multiview technologies

- Applications
  - Regular light field displays
  - Near-eye light field displays[1] for virtual reality

Viewer moving around a Holovizio display



[1]**Near-Eye Light Field Displays** - D. Lanman & D.Luebke. ACM Trans. Graph. 32, 6, Article 220 (November 2013)

Motivation: Virtual Reality

- Light field displays & Virtual Reality
  - *Is real-time photorealistic rendering of dynamic camera movement feasible for multiview based applications at HD (1280x720) resolution ?*
  - *Real-time: ~90 fps for VR*

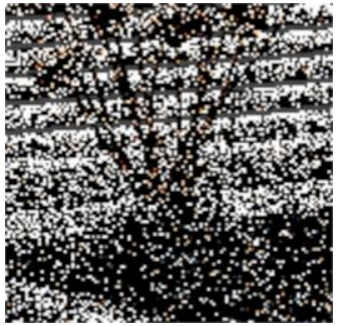
Example of dynamic camera movements to simulate user's head-motion

Problem statement

- Photorealistic rendering of 1 single view

✗ Poor visual results

✓Fast



1 sample per pixel (spp)

✓Good visual results

✗ Slow

4096 spp

Problem statement

- Photorealistic rendering of 1 single view

– Correct visual results

✓Fast

1 sample per pixel (spp)
+ denoising

✓Good visual results

✕ Slow

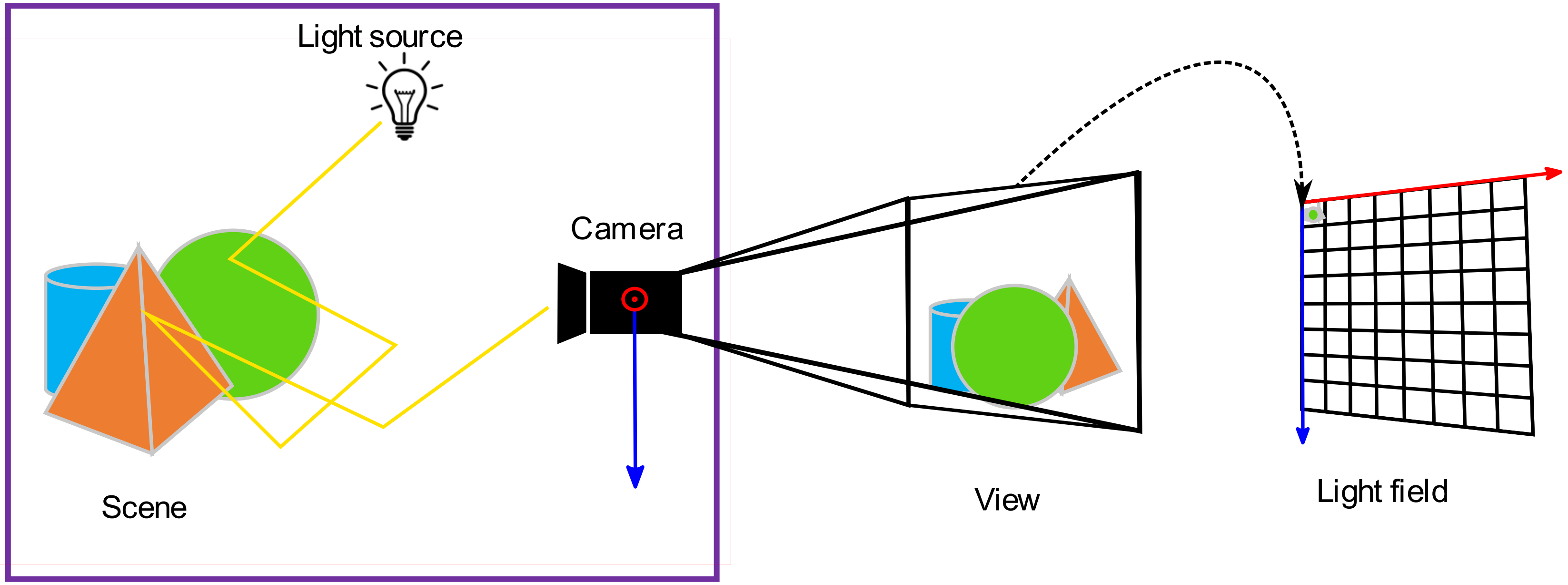
4096 spp

Problem statement

- Multiview path tracing

❌ **Too slow**

Single computer

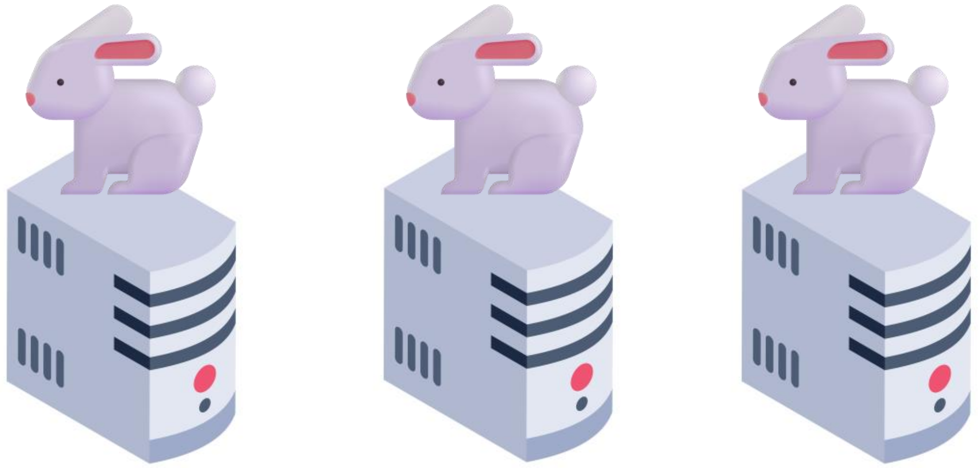


Path tracing

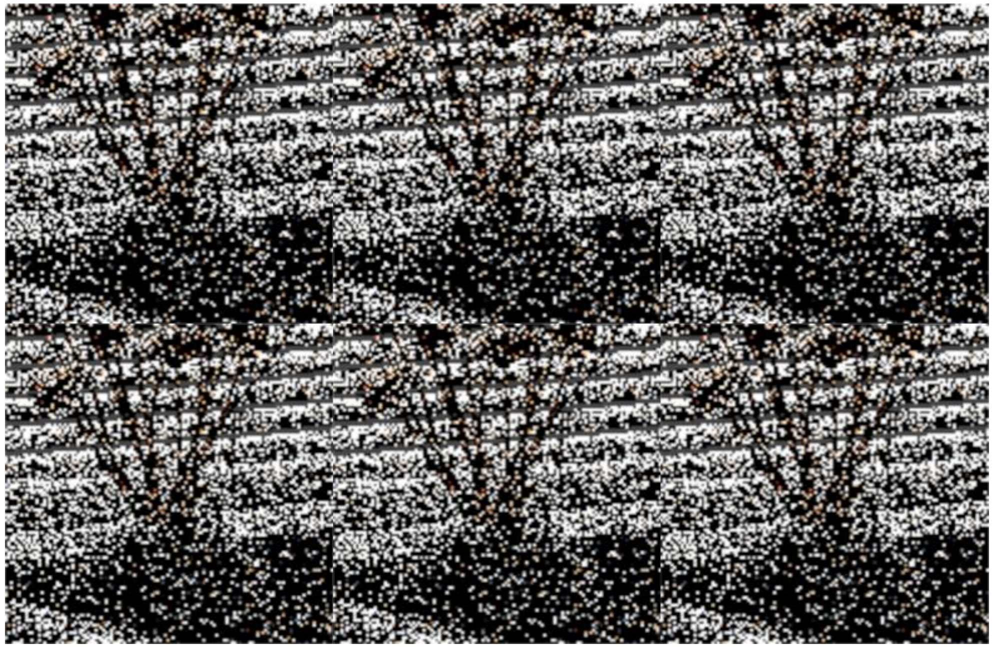
Output light field

Problem statement

- Multiview path tracing workload



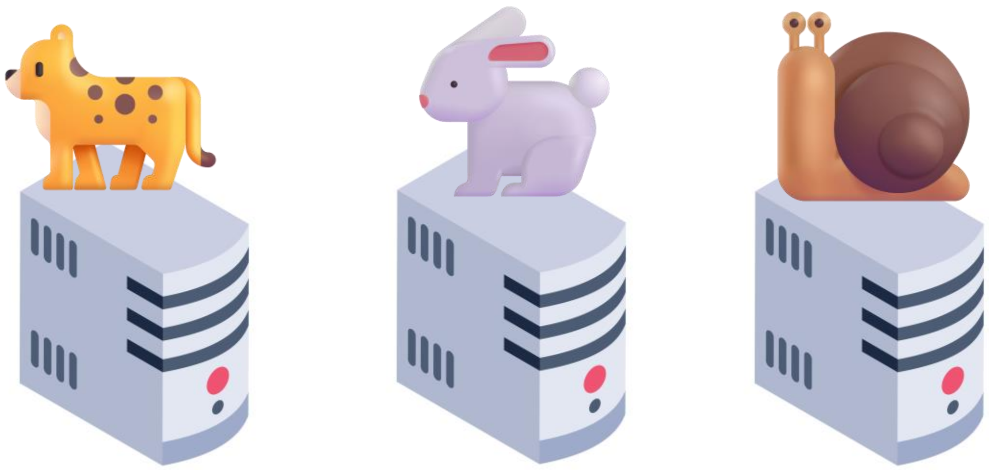
Homogeneous compute platform



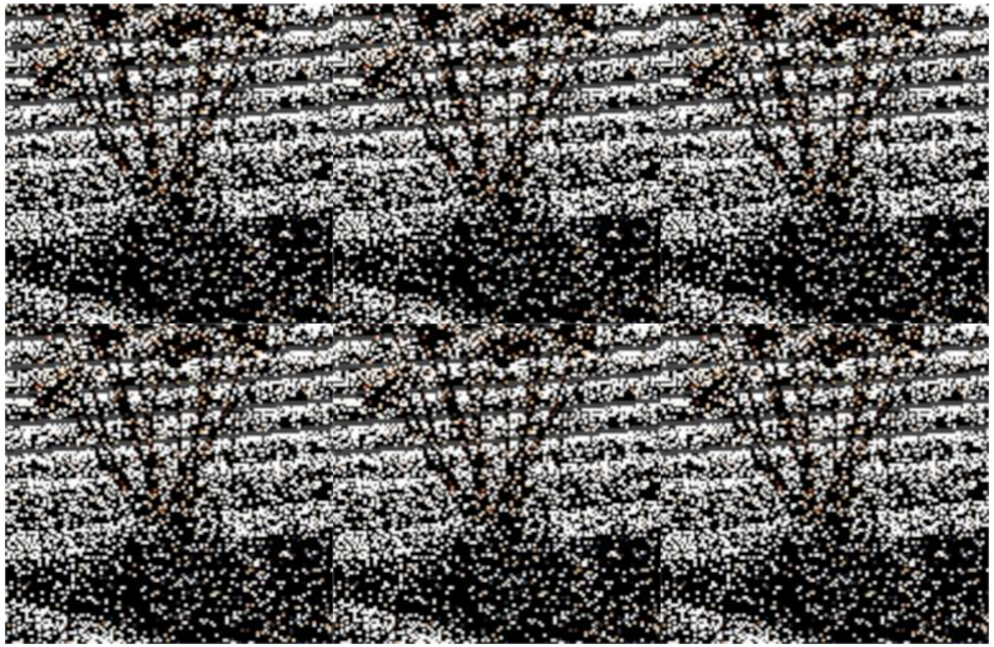
3x2 multiview grid rendered with 1 sample per pixel

Problem statement

- Multiview path tracing workload



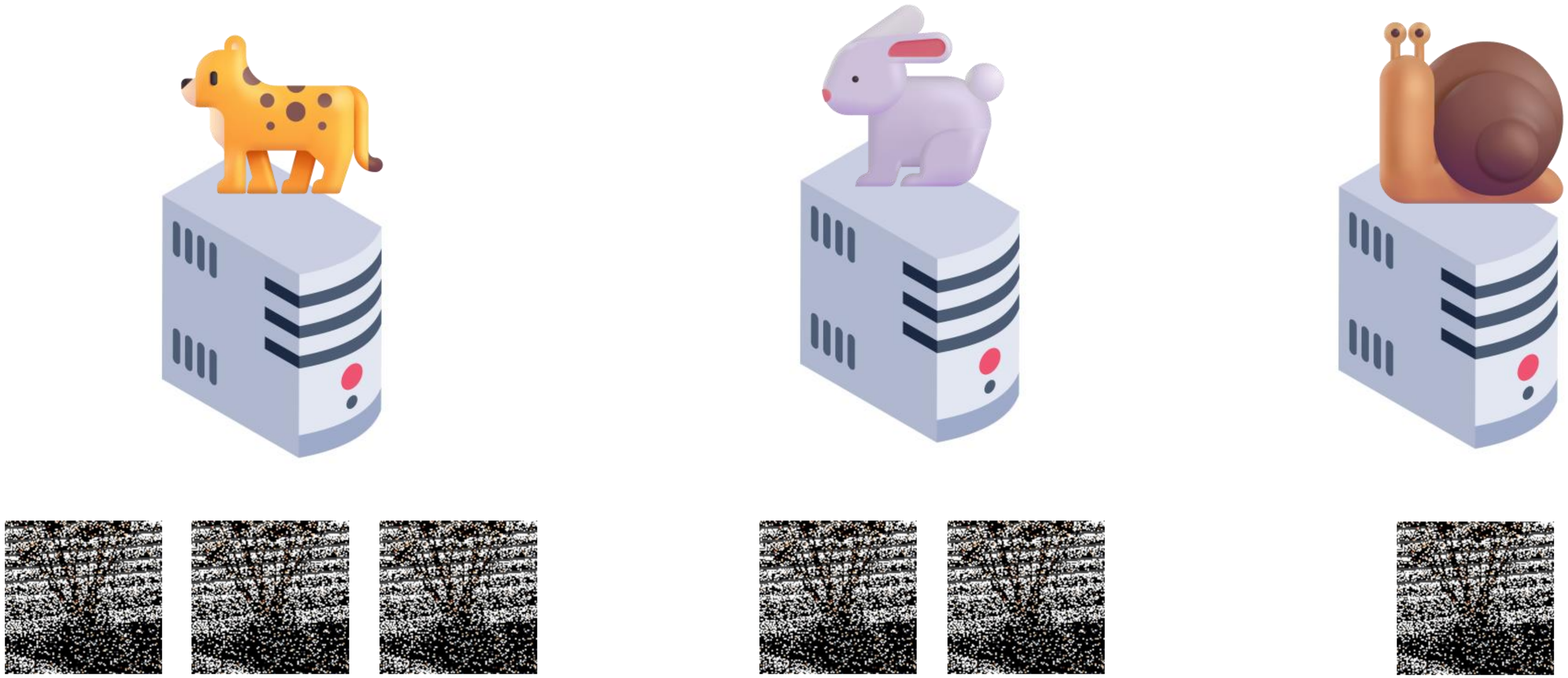
Heterogeneous compute platform



3x2 multiview grid rendered with 1 sample per pixel

Problem statement

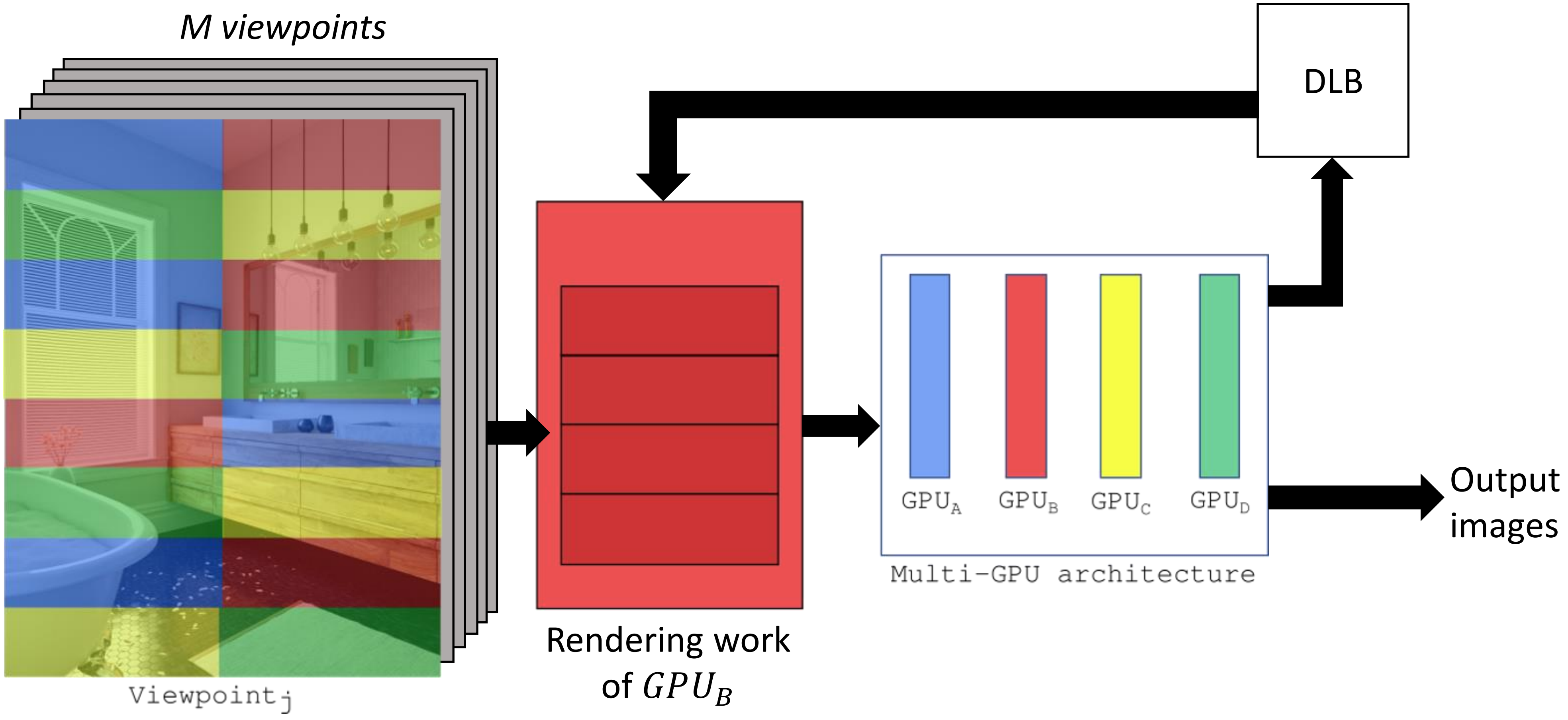
- How to map multiview path tracing workload to multiple compute devices?



Our contributions

- A formal model to map path tracing workload from multiple viewpoints to multiple compute devices
- A dynamic load balancing (DLB) algorithm can be integrated within our model to target real-time multiview path tracing
- Our implementation of the DLB scales across multiple light field resolutions with a constant performance gain over time

Method overview

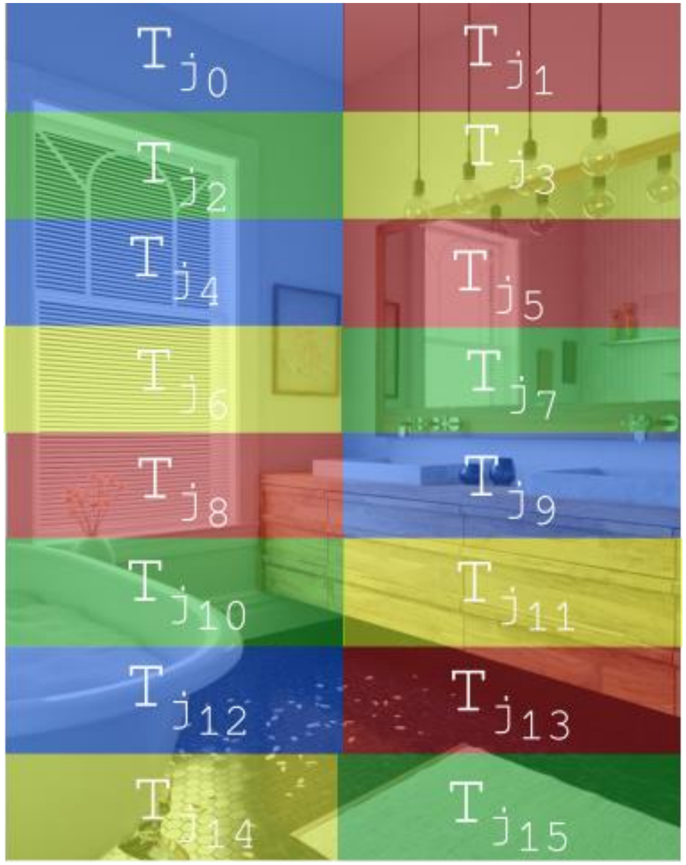


Scaling multiview path tracing

- We define T_j a set that contains tasks T_{j_k}.
- Each of these sets is associated to a viewpoint j.

$$T_j = \{T_{j_0}, \dots, T_{j_k}, \dots, T_{j_{K-1}}\} \Longleftrightarrow$$

Example with K = 16

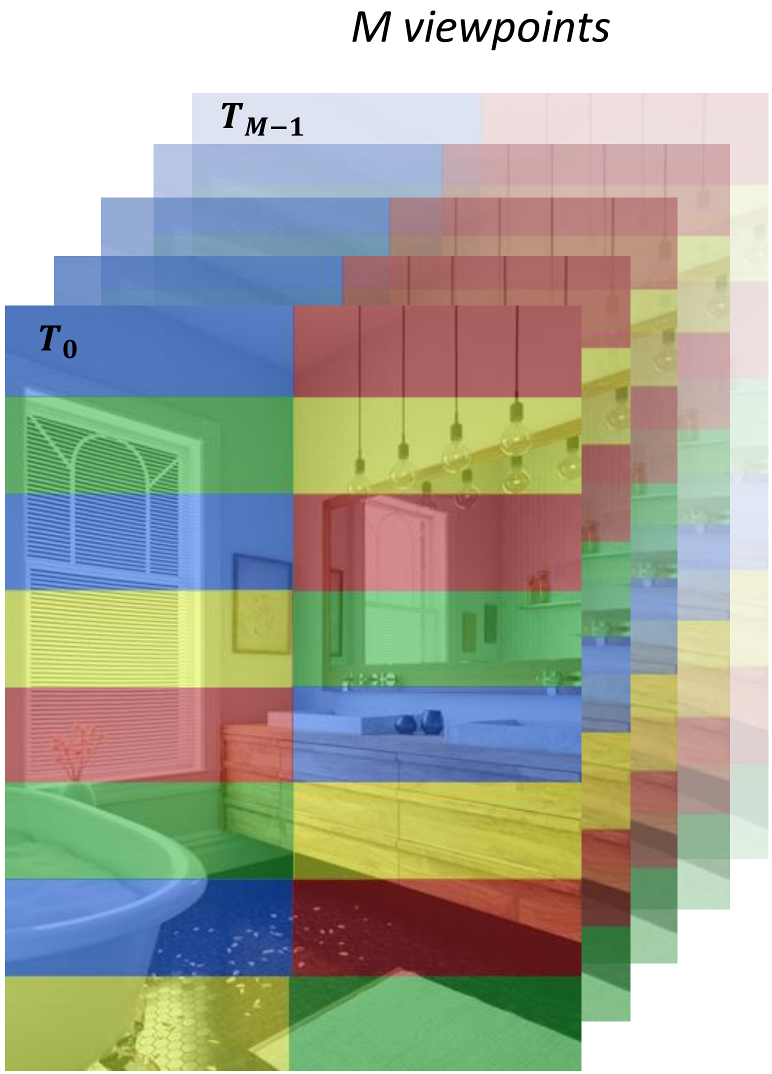


Viewpoint_j

Scaling multiview path tracing

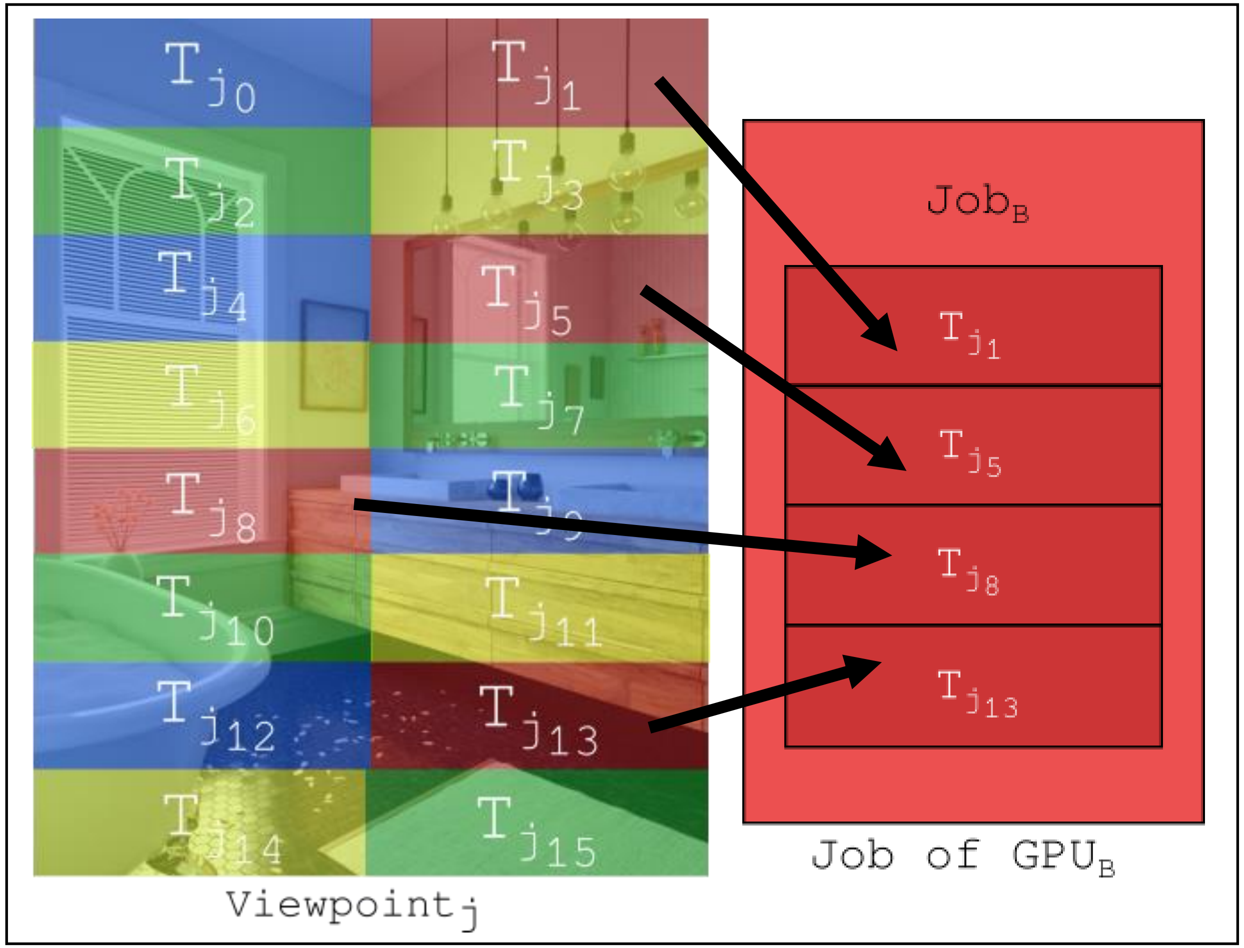
- All the sets T_j **form the global set of tasks** T.

$$T = \{T_0, \dots, T_j, \dots, T_{M-1}\} \iff$$



Scaling multiview path tracing

- Each GPU is assigned a set of tasks. We call this set of tasks a **job**.
  - A job contains screen regions from different viewpoints and is sent to the GPU to be processed.



Example for 1 viewpoint

Scaling multiview path tracing

- We avoid data structure management for tasks by expressing the workload job_i as a quantity of tasks.

$$|job_i| = w_i \cdot |T|$$

- w_i is the ratio of tasks to be processed by GPU i

Scaling multiview path tracing

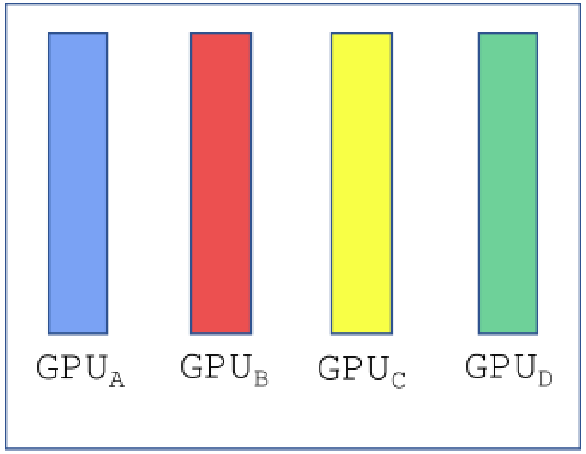
- Since T is a set of subsets T_j and has a size M, then we can reformulate the previous equation as follows:

$$|job_i| = w_i \cdot |T| \iff |job_i| = \sum_{j=0}^{M-1} w_i \cdot |T_j|$$

Scaling multiview path tracing

- At the **initialization** of the rendering application, we assign a uniform number of tasks to each GPU.

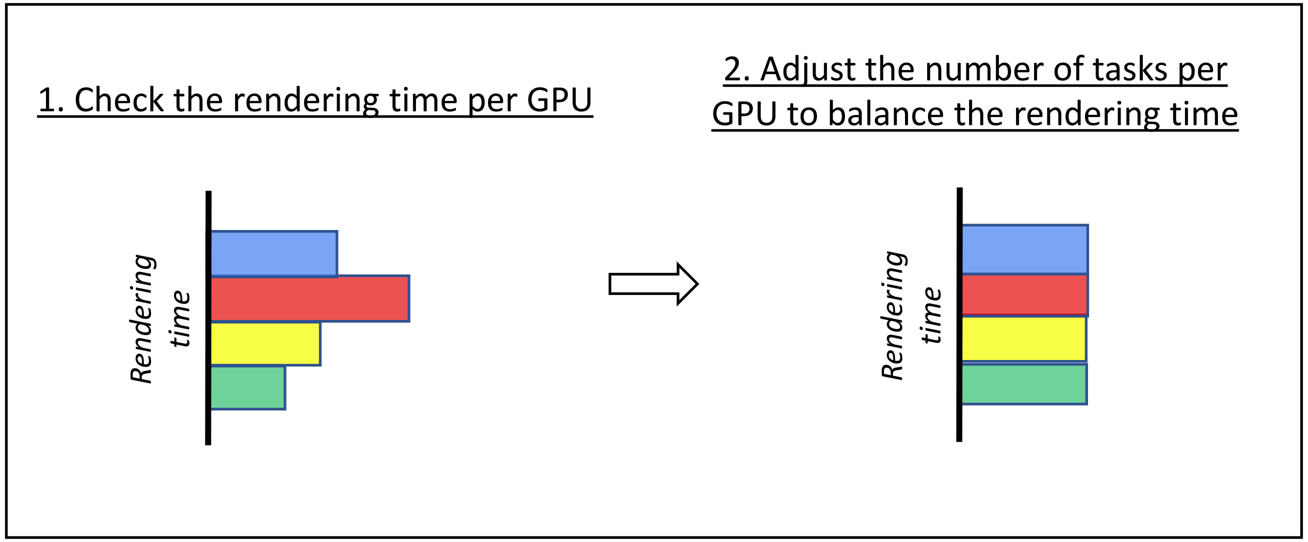
$$w_i = \frac{1}{N} \Longleftrightarrow$$



Multi-GPU architecture

Dynamic load balancing of screen space tasks

- Overview



Dynamic load balancing of screen space tasks

- For each GPU i, we need to compute their individual workload ratio w_i
  - We compute the rendering speed of each GPU i: $r_i = \frac{|job_i|}{\Delta_i}$
  - Then, we normalize the rendering speed for each GPU i: $w_i = \frac{r_i}{\sum_{\iota=0}^{N-1} r_\iota}$

Implementation

- Vulkan path tracer implemented in C++
- The screen space decomposition we used is the *shuffled strips*[1]

[1]**A Simple Load-Balancing Scheme with High Scaling Efficiency** - Antwerpen, D.v., Seibert, D., Keller, A. Ray Tracing Gems (2019).

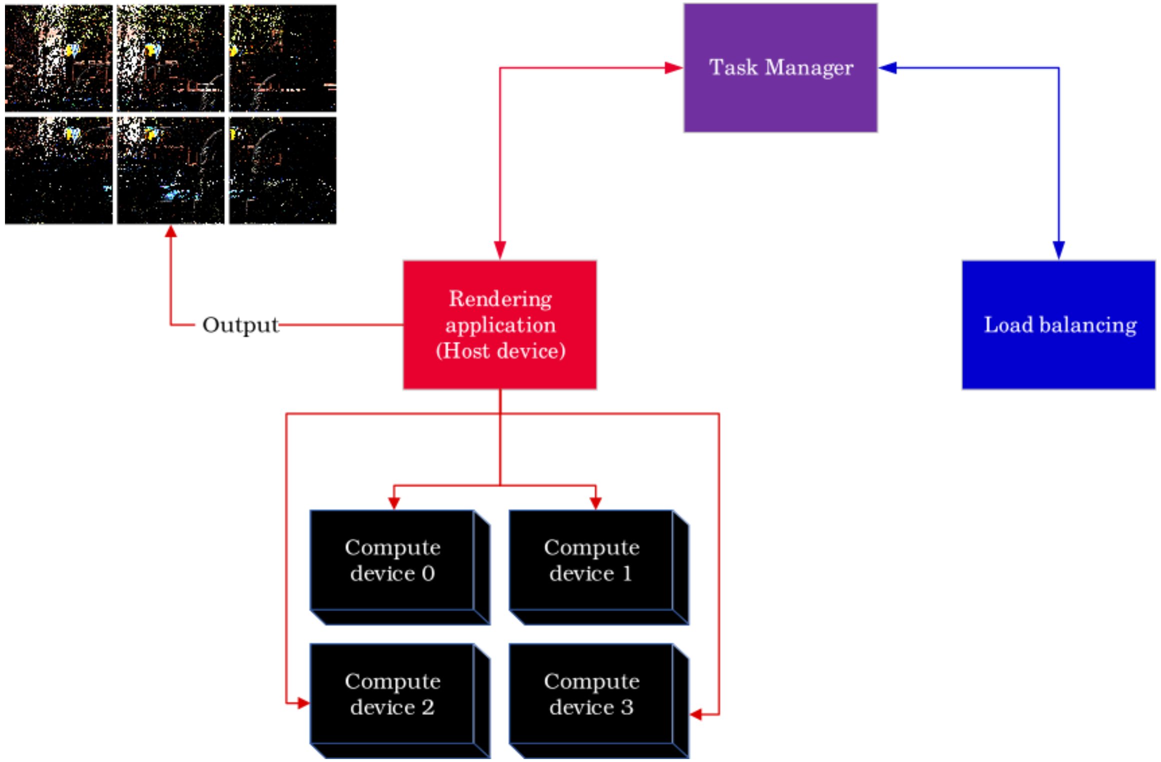
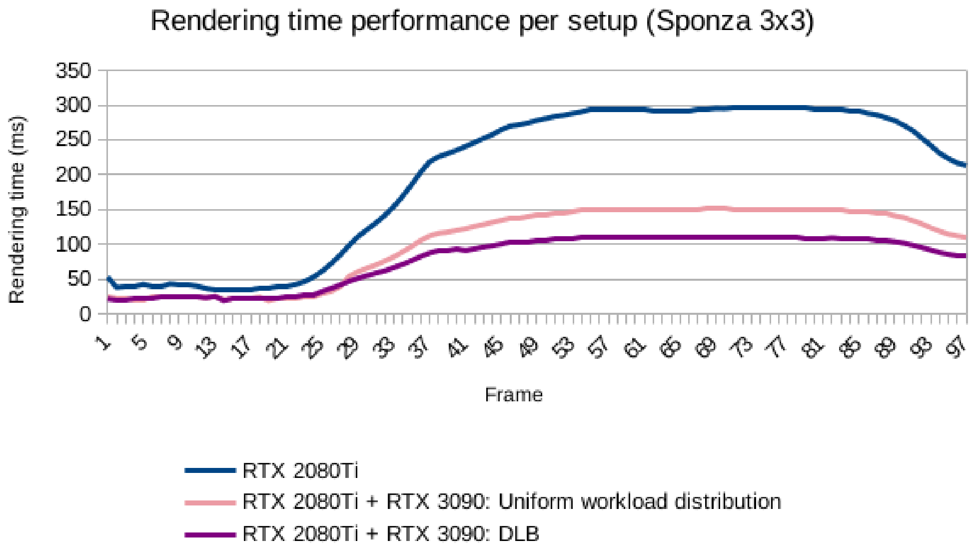


Diagram of our implementation

Experiments and results

- Relative gain: a single GPU against dual-GPU setups



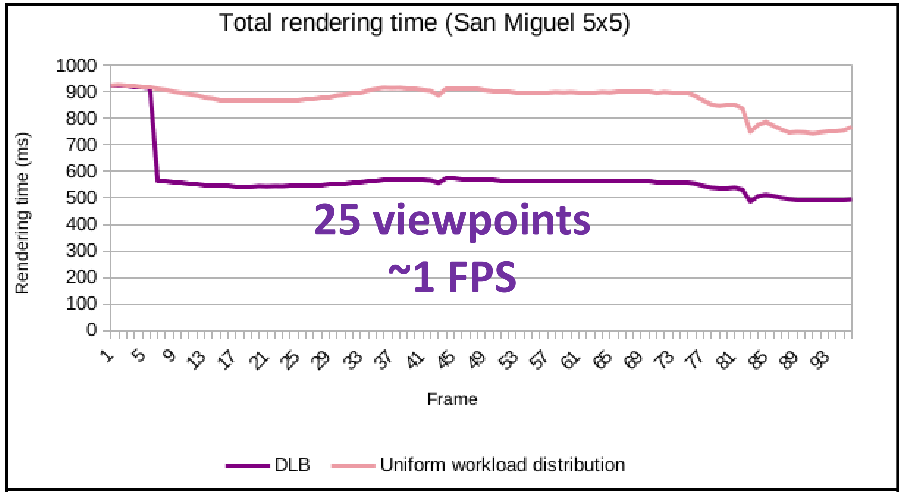
Single view animation for the Sponza scene (1spp)

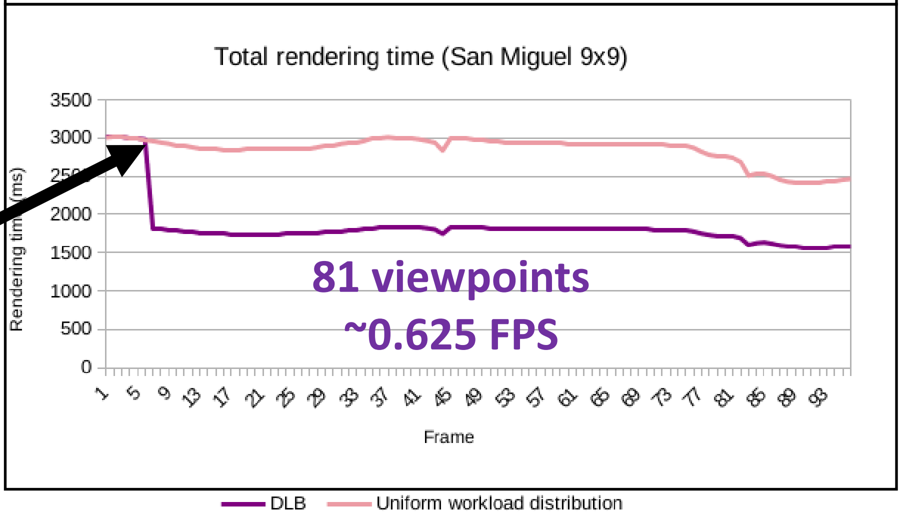
Experiments and results

- Constant performance gain for heterogeneous dual-GPU
  - Uniform workload distribution
  - **Dynamic Load Balancing**

Single view San Miguel scene (top left reference image)

Load balancing





Experiments and results

- Spatial reprojection: 81 viewpoints
  - 1 path traced viewpoint
  - Below the primary GPU is indicated by (0)

	Uniform workload distribution		DLB	
	RTX 2080Ti (0)	RTX 2080Ti (1)	RTX 2080Ti (0)	RTX 2080Ti (1)
Path tracing (ms)	11.35	34.19	18.37	24.14
G-buffer transfers from host - for 80 viewpoints - (ms)	31.08	-	31.09	-
Spatial reprojection (ms)	7.41	-	7.40	-

Single view *Abandoned warehouse + Chinese Dragon* scene (reference image)

Limitations and future work

- Non-primary devices are idle during post-processing stages (including spatial reprojection)
  → Modeling a fully parallel multiview path tracing pipeline with parallel post-processing stages

- I/O communication overhead occurring on the host after the spatial reprojection stage delays the transfers of buffers from non-primary GPUs
  →Direct Memory Access, Locality aware multiview path tracing

- Rendering a large number of light fields per time frame is memory consuming
  → Light field compression

Conclusions

- We proposed a **formal model** to map screen space tasks on multi compute device platforms for **multiview path tracing**.
- We implemented our mapping model inside a **real-time scalable multiview path tracing pipeline**. We showed that our model is suitable for dynamic load balancing of multiview path tracing workloads.
- Our DLB algorithm shows a **constant performance gain** on a heterogeneous dual-GPU setup **between 30% and 50%** with our test scenes.

CGI2022

Thank you for your attention