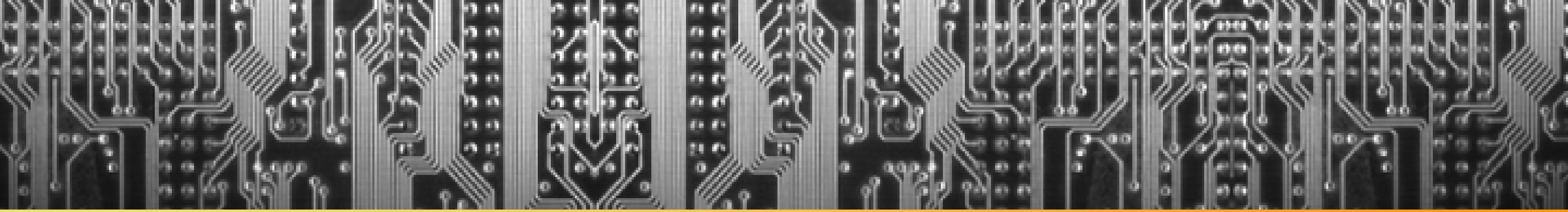


An Improved Shading Cache for Modern GPUs

Pitchaya Sitthi-amorn, Jason Lawrence
University of Virginia

Yang Lei, Pedro V. Sander
Hong Kong University of Science and Technology

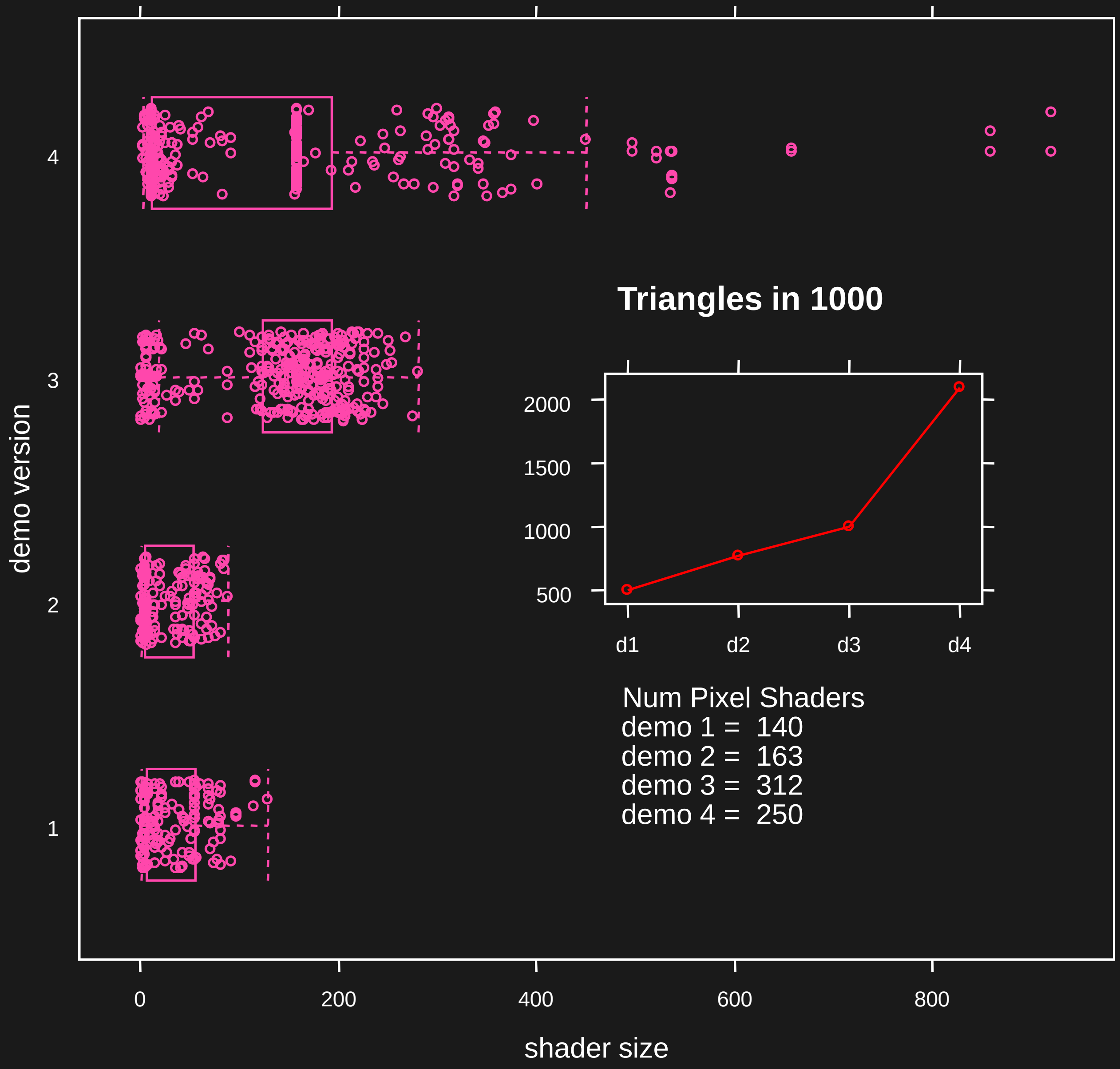
Diego Nehab
Microsoft Research

Motivation

SPEED
25
ATI

Courtesy of AMD/ATI

Shader Complexity of ATI demos



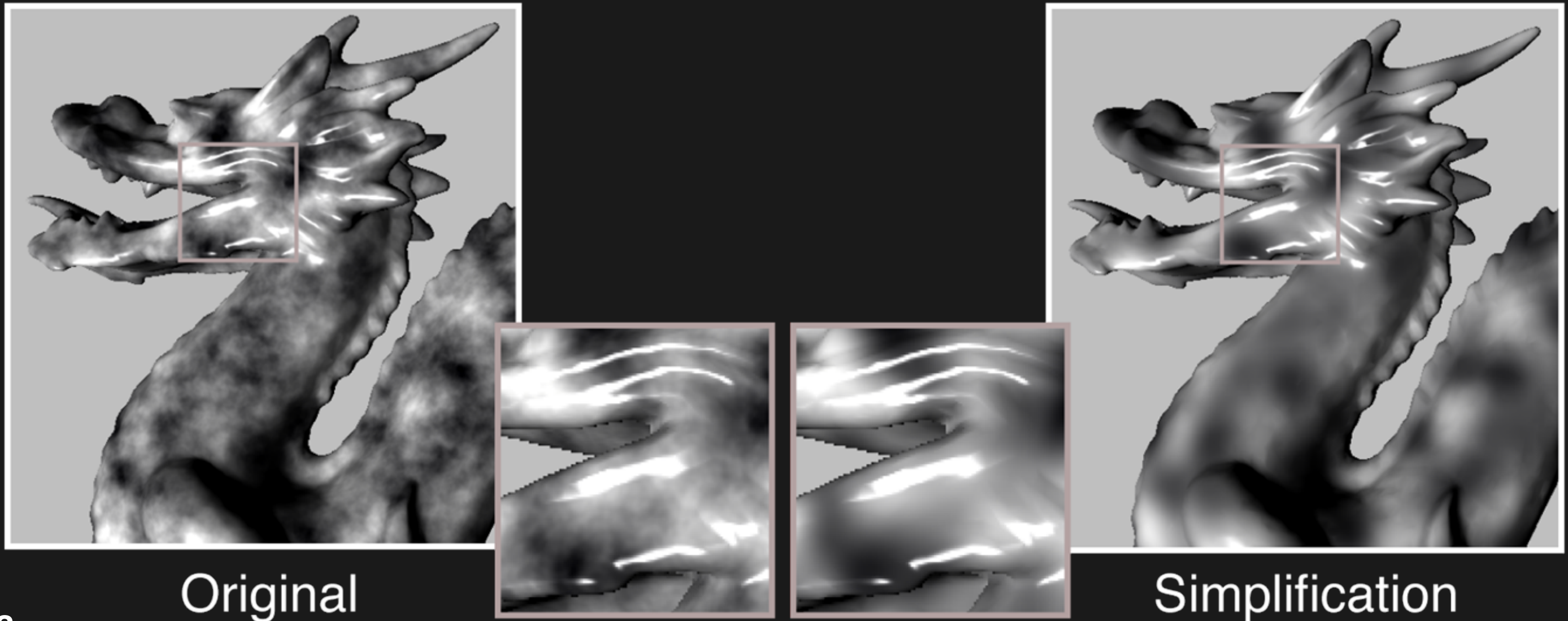
Triangles in 1000

Num Pixel Shaders
demo 1 = 140
demo 2 = 163
demo 3 = 312
demo 4 = 250

Courtesy of Norm Rubin (AMD/ATI)

Related Work: Code Simplification

- Replace subexpressions with constants
- Automatic shader level of detail [Olano et al. 2003]
- User-configurable automatic simplification [Pellacini 2005]

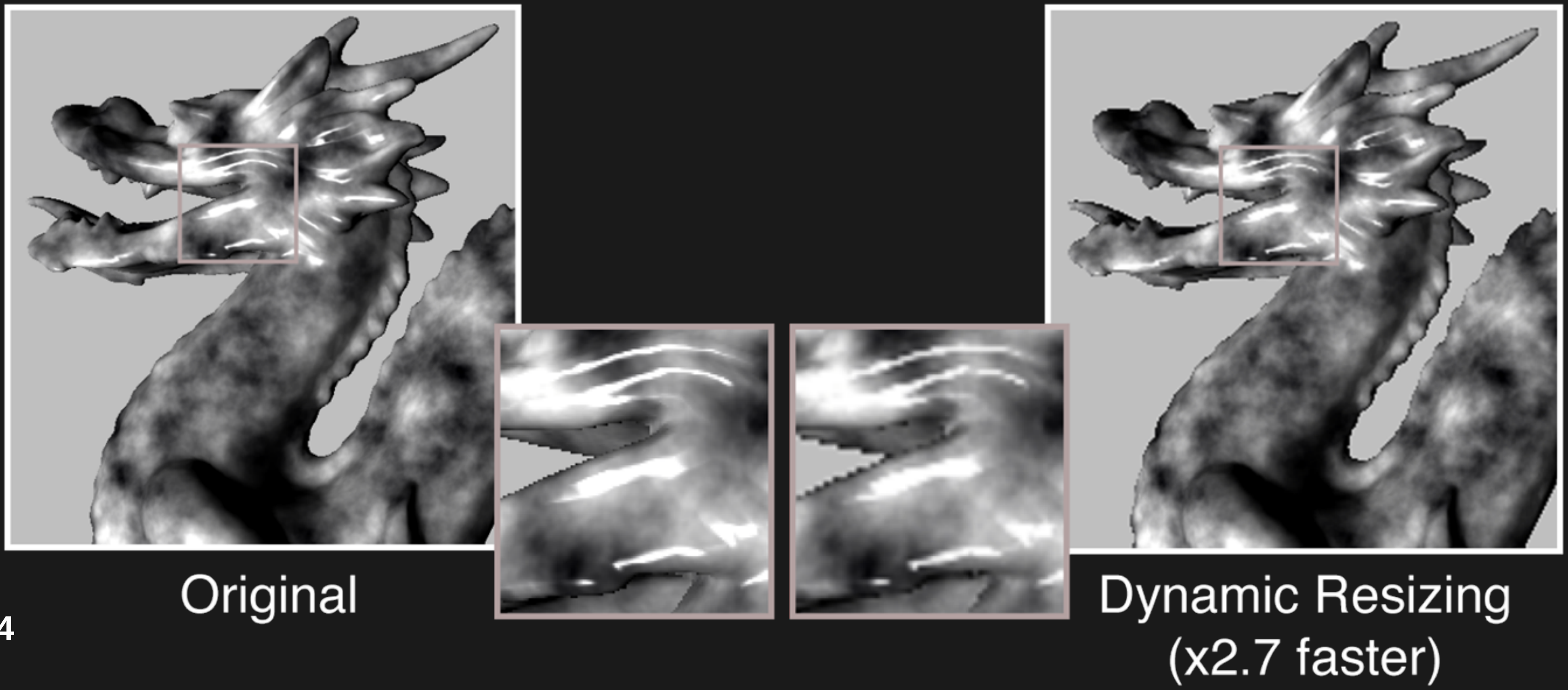


Original

Simplification
(x2.2 faster)

Related Work: Dynamic Resizing

- Render scene to lo-res off-screen buffer and upsample to target resolution
- *InfiniteReality* system [Montrym et al. 1997]
- Geometry-Aware resizing [Yang et al. 2008] (concurrent)

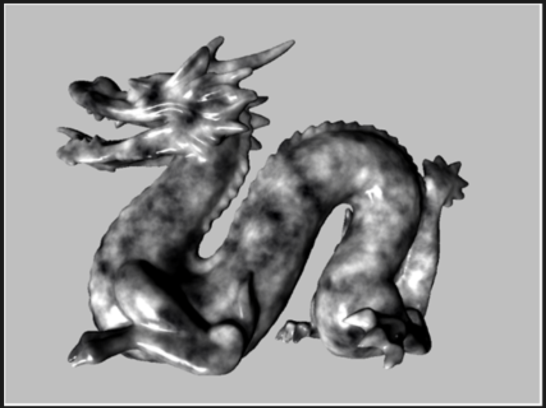


Original

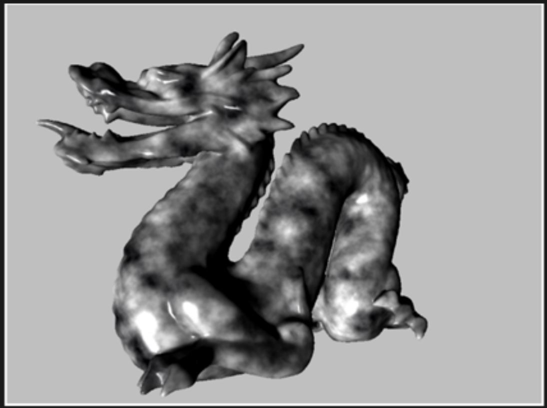
Dynamic Resizing (x2.7 faster)

Related Work: Temporal Reprojection

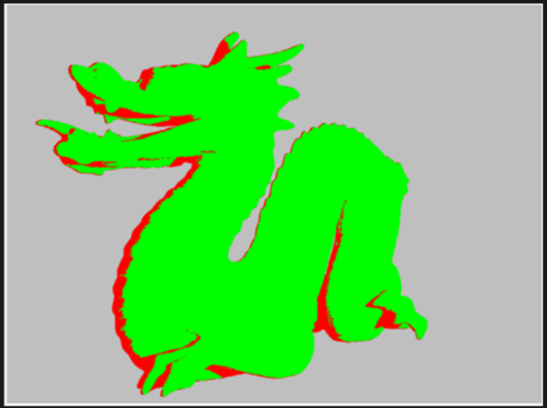
- Reuse partial shading calculations across consecutive frames
- Reverse reprojection cache [Nehab et al. 2007]
- Pixel-correct shadow maps with temporal reprojection and shadow test confidence [Scherzer et al. 2007]
- Multi-view architecture [Hasselgren et al. 2006]



frame n-1



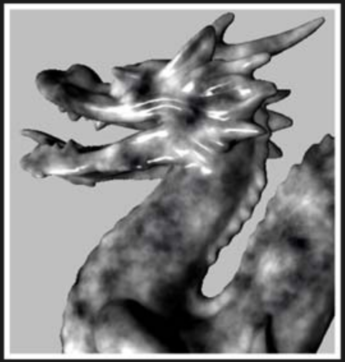
frame n



green = mutually visible
red = occluded

Real Time Shading Cache

Frame n-1

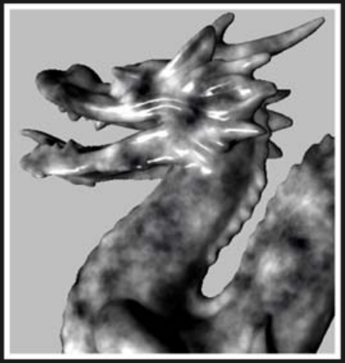


framebuffer

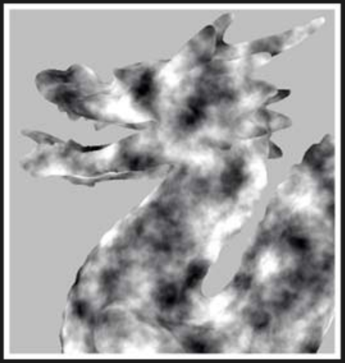
Frame n

Real Time Shading Cache

Frame n-1



framebuffer

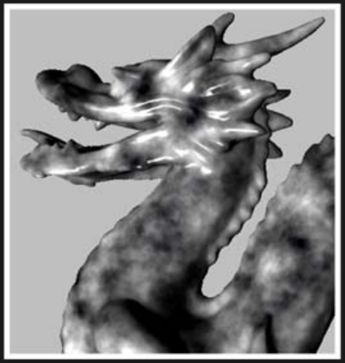


payload

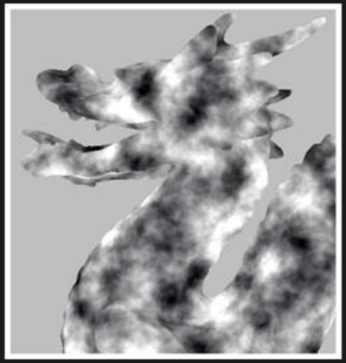
Frame n

Real Time Shading Cache

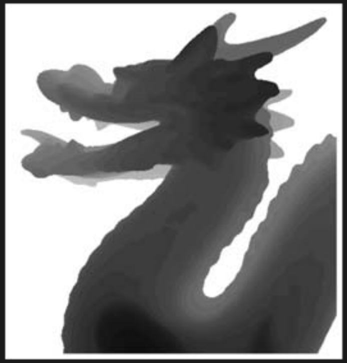
Frame n-1



framebuffer



payload

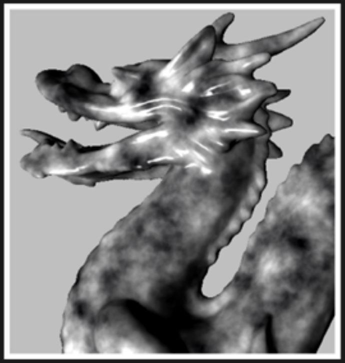


depth

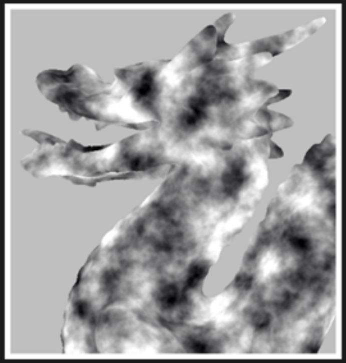
Frame n

Real Time Shading Cache

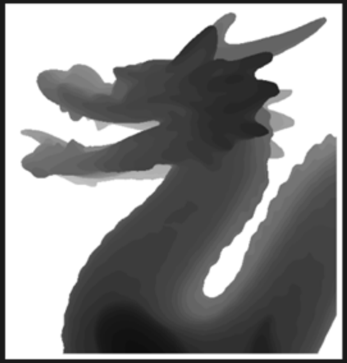
Frame n-1



framebuffer



payload



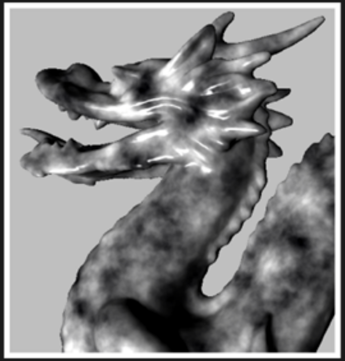
depth

Shading Cache

Frame n

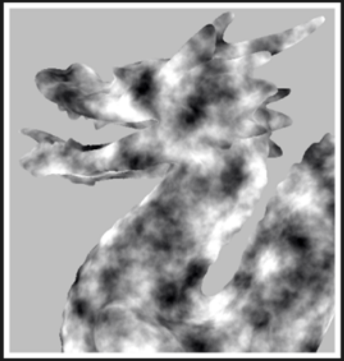
Real Time Shading Cache

Frame n-1

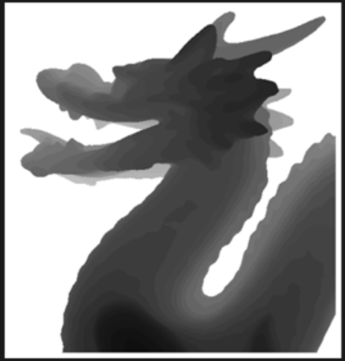


framebuffer

Shading Cache

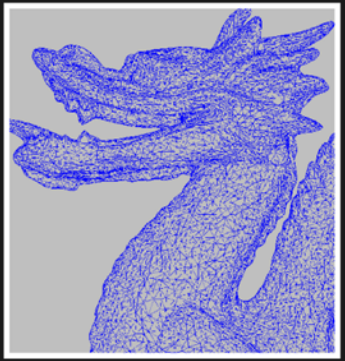


payload



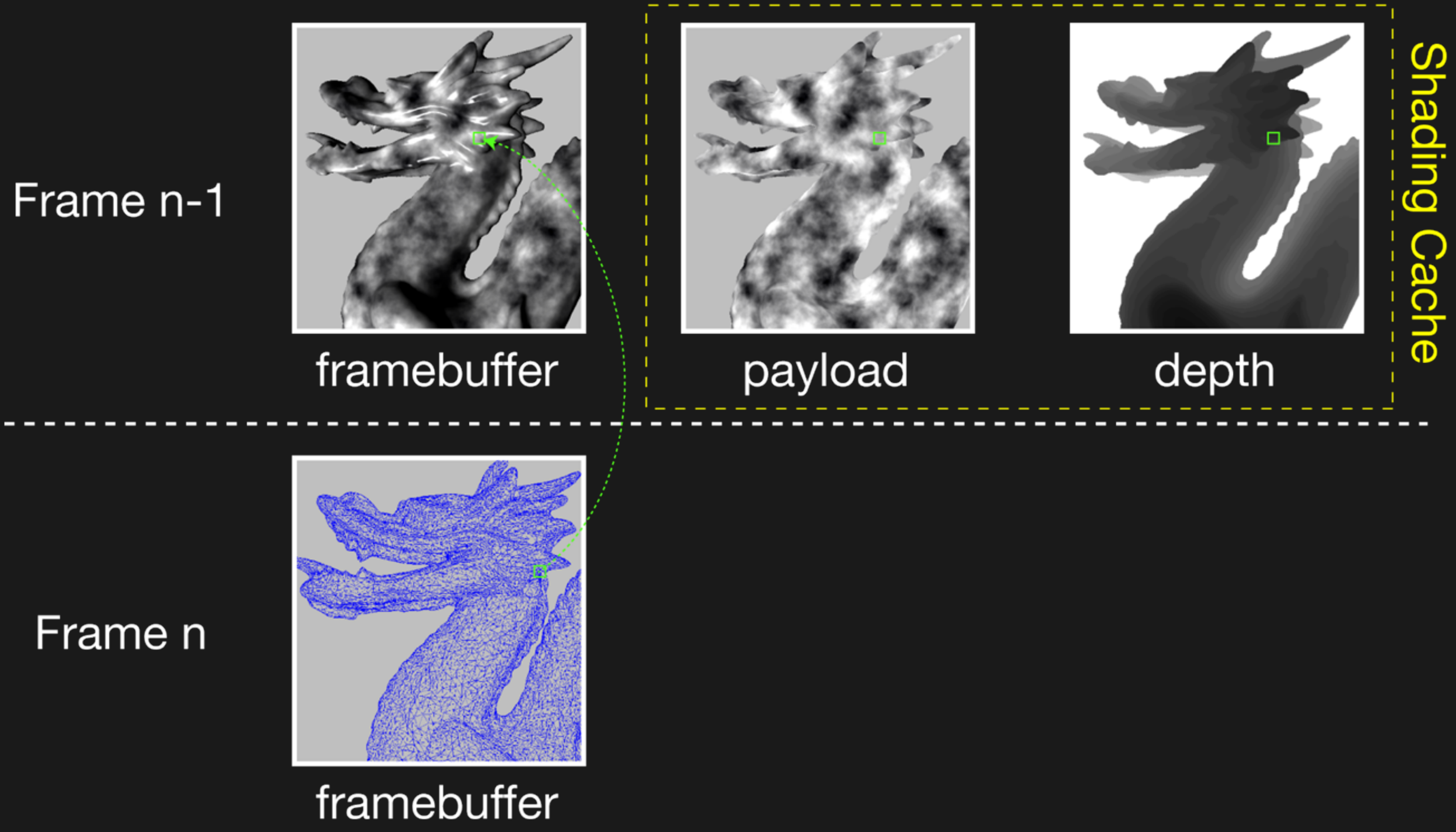
depth

Frame n



framebuffer

Real Time Shading Cache



Frame n-1

framebuffer

payload

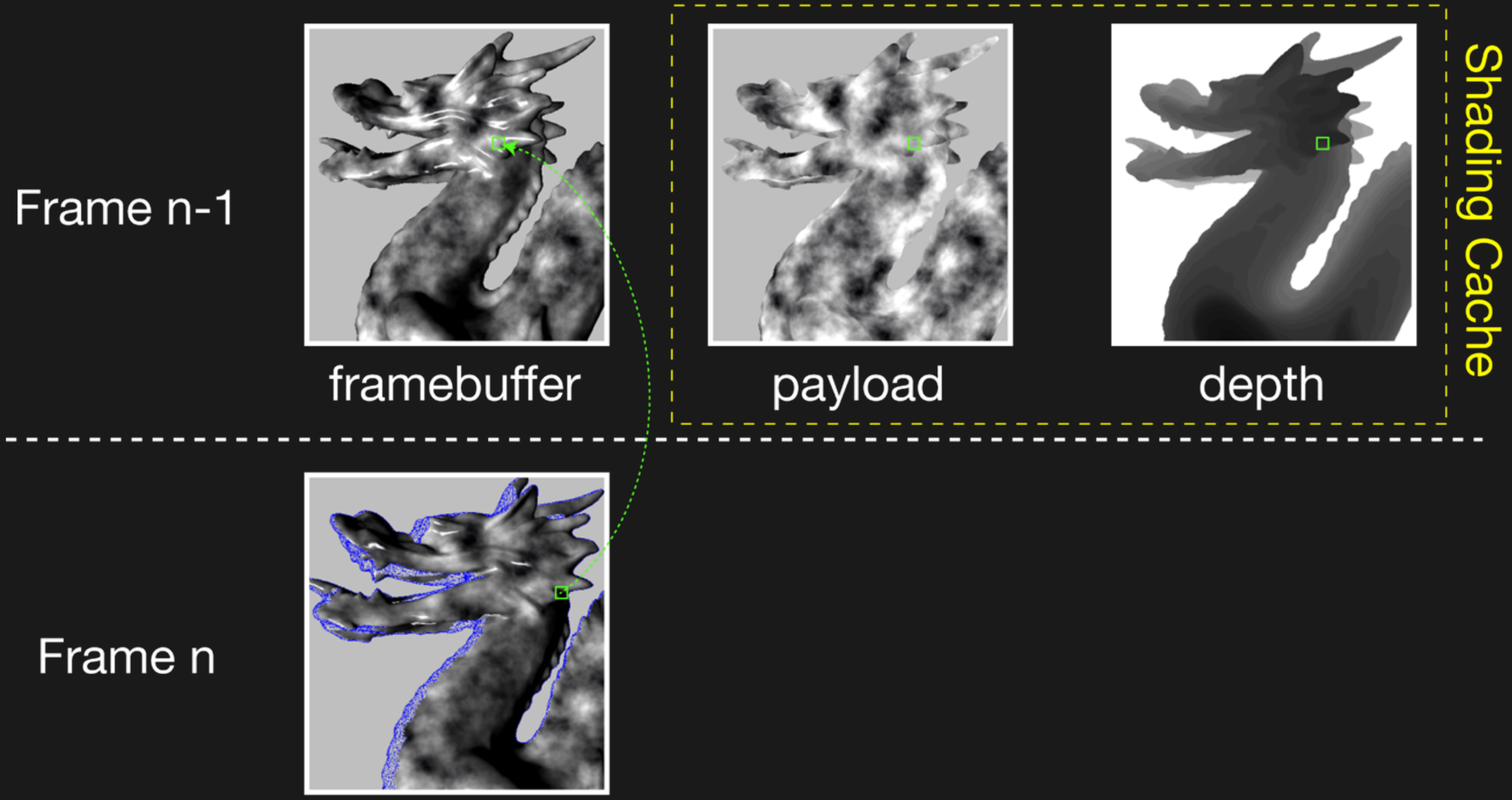
depth

Shading Cache

Frame n

framebuffer

Real Time Shading Cache



Frame n-1

framebuffer

payload

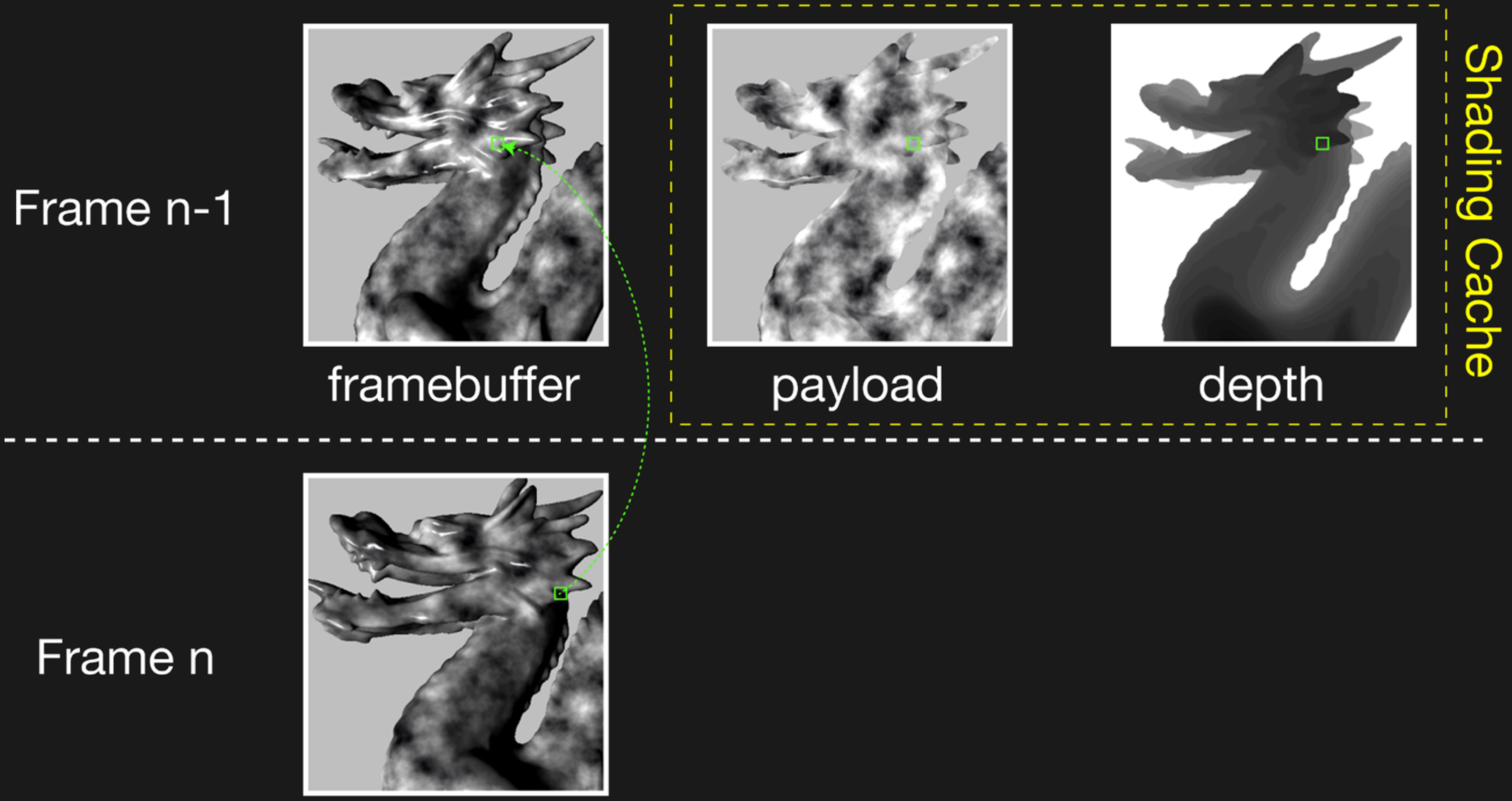
depth

Shading Cache

Frame n

framebuffer

Real Time Shading Cache



Frame n-1

framebuffer

payload

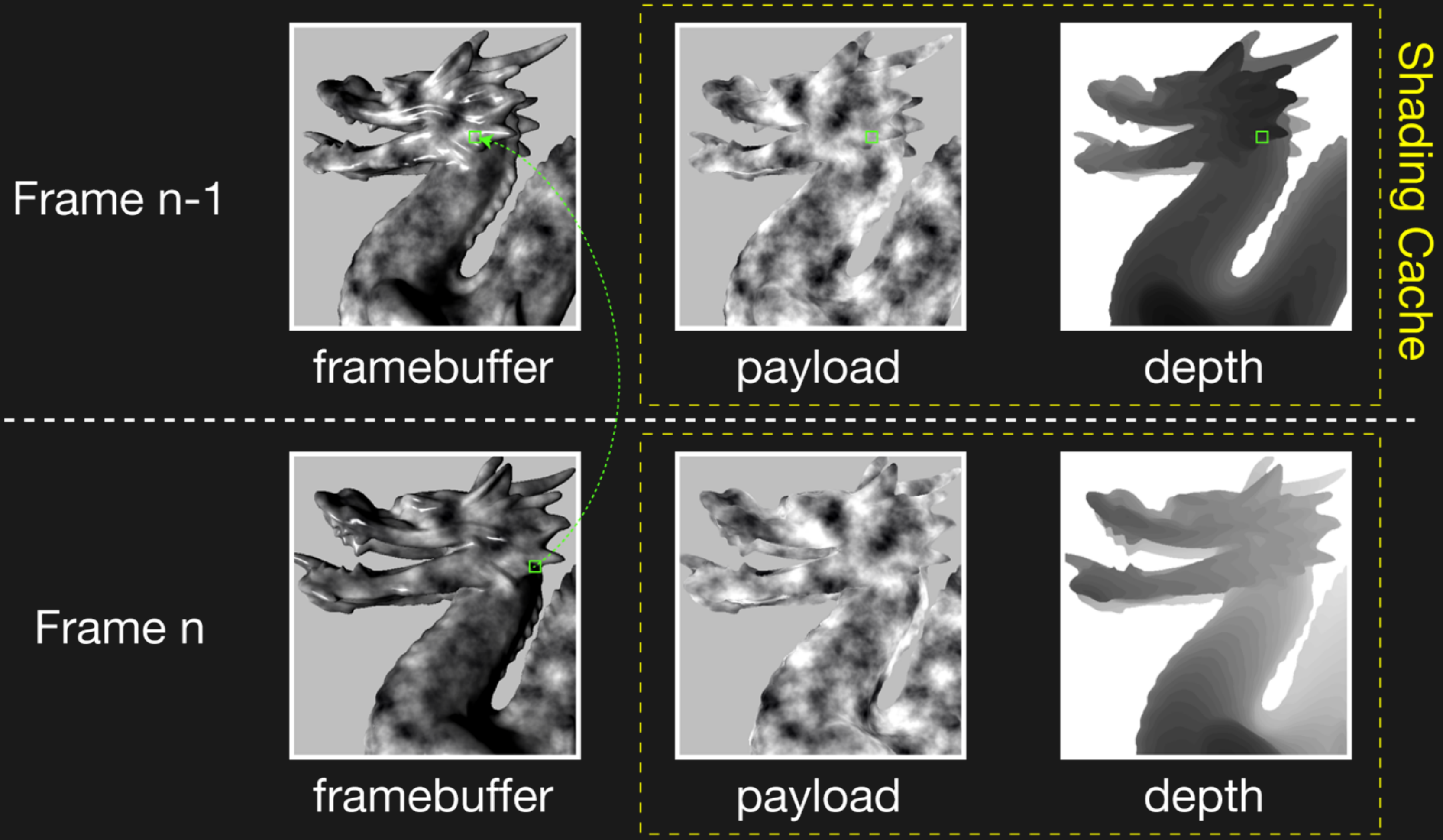
depth

Shading Cache

Frame n

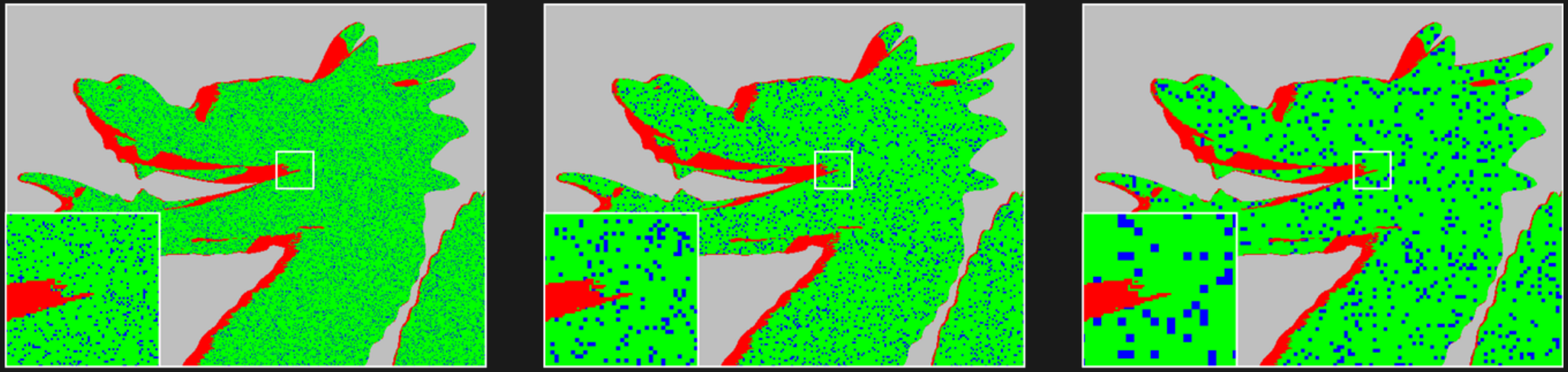
framebuffer

Real Time Shading Cache



Cache Refresh

- Scene points may remain visible over many frames
- Cached entries will become stale due to changes in shader inputs and from resampling error
- Explicitly refresh cached entries within a user-set refresh period Δn by forcing misses within k x k blocks of pixels



$k = 1$

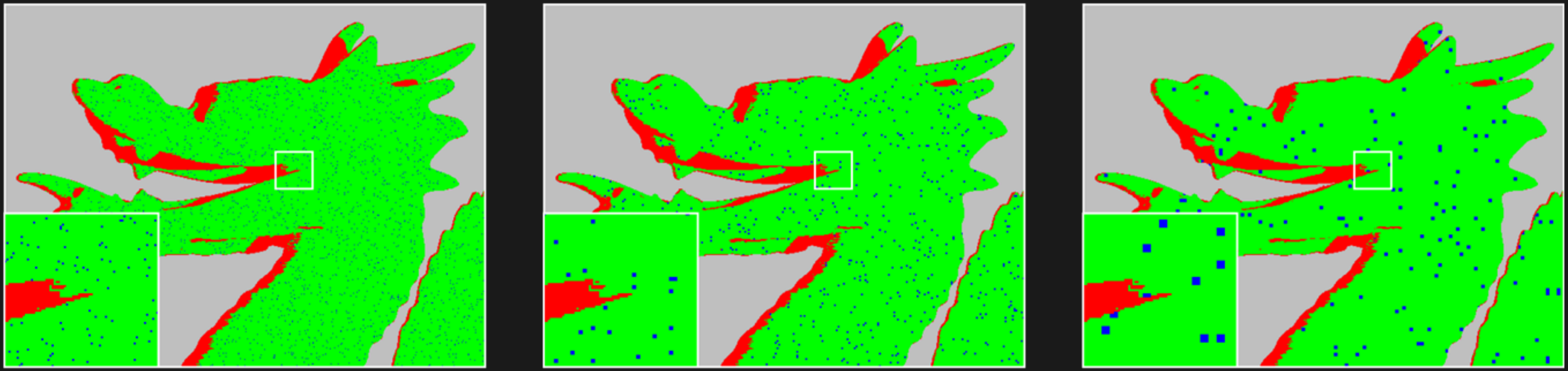
$k = 2$

$k = 4$

Δn = 10

Cache Refresh

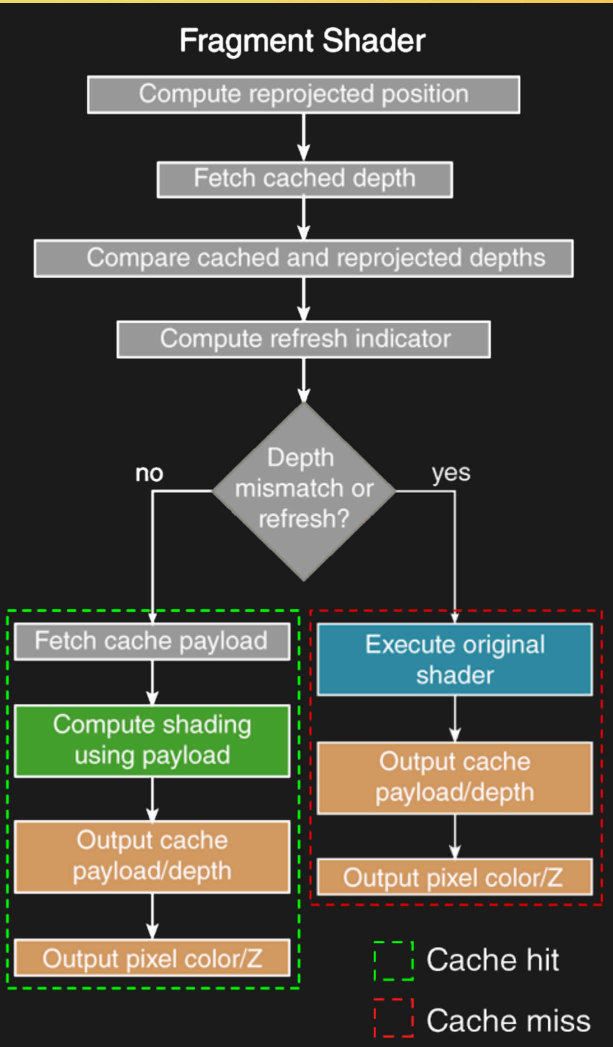
- Scene points may remain visible over many frames
- Cached entries will become stale due to changes in shader inputs and from resampling error
- Explicitly refresh cached entries within a user-set refresh period Δn by forcing misses within k x k blocks of pixels



$k = 1$ $k = 2$ $k = 4$

$\Delta n = 50$

1-Pass Algorithm [Nehab et al. 2007]



Fragment Shader

Compute reprojected position

Fetch cached depth

Compare cached and reprojected depths

Compute refresh indicator

Depth mismatch or refresh?

no

yes

Fetch cache payload

Compute shading using payload

Output cache payload/depth

Output pixel color/Z

Execute original shader

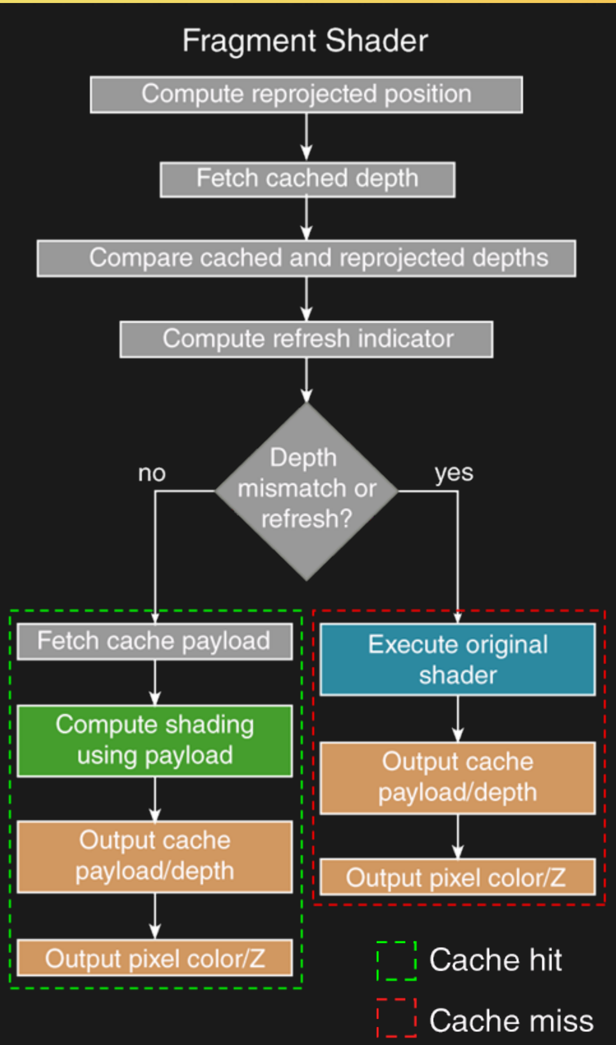
Output cache payload/depth

Output pixel color/Z

Cache hit

Cache miss

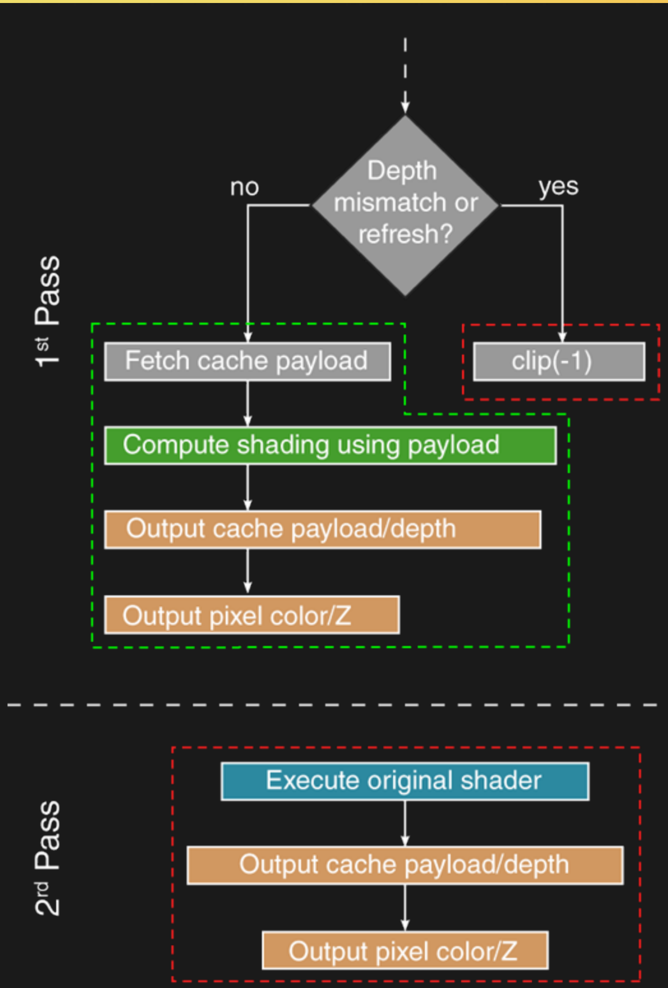
1-Pass Algorithm [Nehab et al. 2007]



Fragment Shader
Compute reprojected position
Fetch cached depth
Compare cached and reprojected depths
Compute refresh indicator
Depth mismatch or refresh?
no
yes
Fetch cache payload
Compute shading using payload
Output cache payload/depth
Output pixel color/Z
Execute original shader
Output cache payload/depth
Output pixel color/Z
Cache hit
Cache miss

- Branch efficiency of underlying hardware
- Relative cost of processing hit and miss
- Use of multiple render targets (MRTs)

2-Pass Algorithm [Nehab et al. 2007]

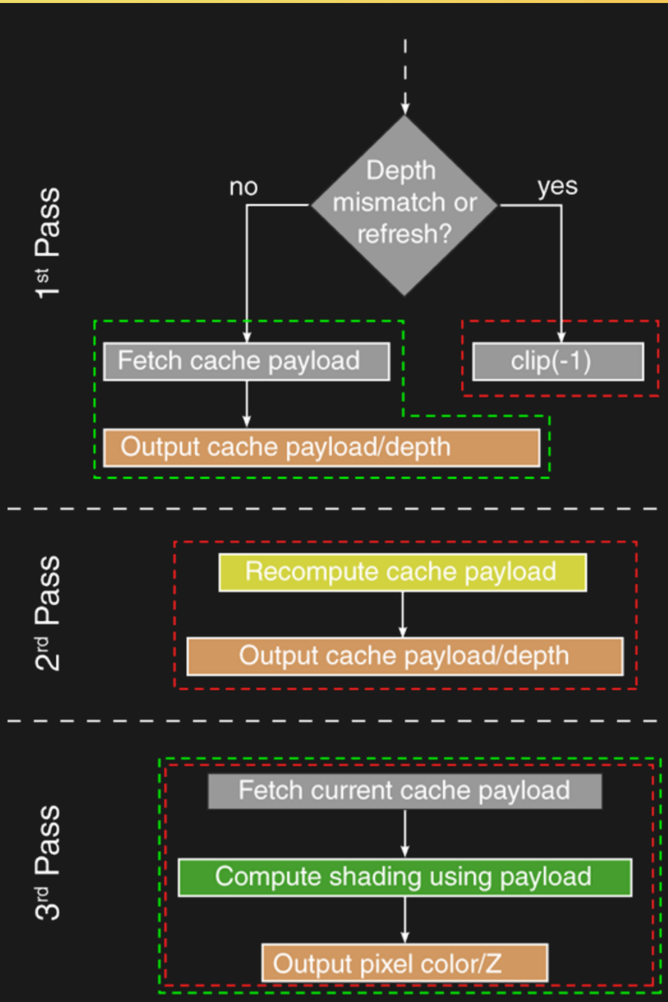


- Still depends on branch efficiency; however, difference in cost of paths is reduced when hit << miss
- Still requires MRTs

⬚ Cache hit

⬚ Cache miss

3-Pass Algorithm (Our approach)

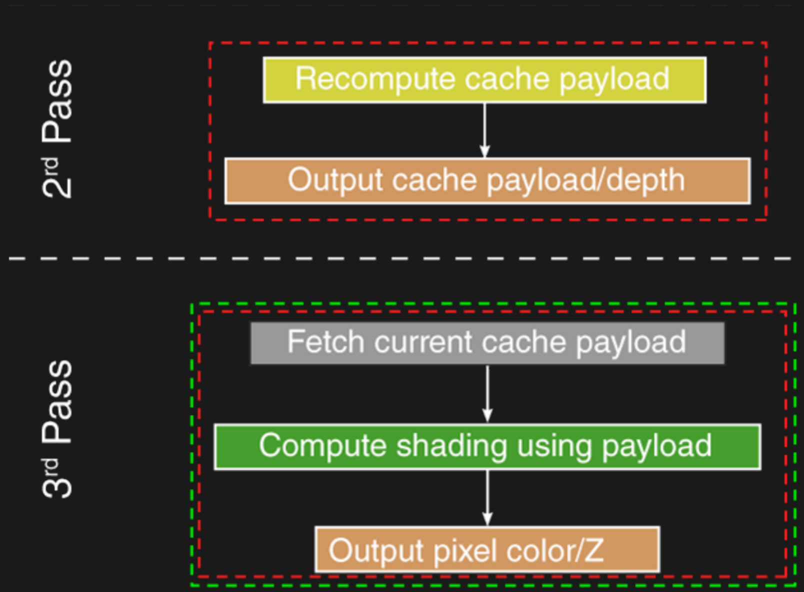

1st Pass
Depth mismatch or refresh?
no
yes
Fetch cache payload
clip(-1)
Output cache payload/depth
2nd Pass
Recompute cache payload
Output cache payload/depth
3rd Pass
Fetch current cache payload
Compute shading using payload
Output pixel color/Z

- Execution paths in the first pass are independent of what is being cached
- Not require MRTs
- Drawback – three rendering passes

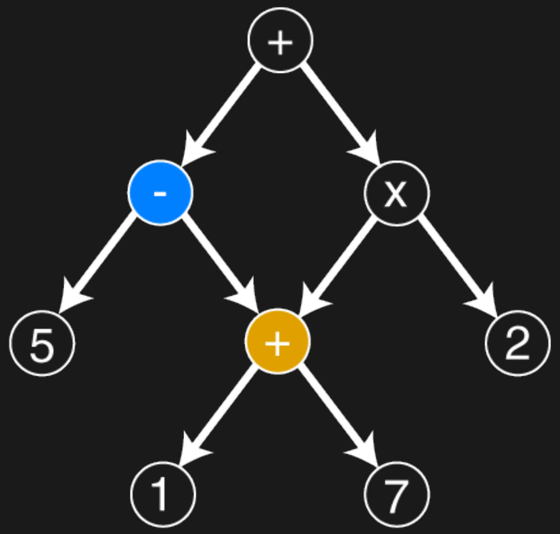
🟩 Cache hit

🟥 Cache miss

Computation Overlap Problem

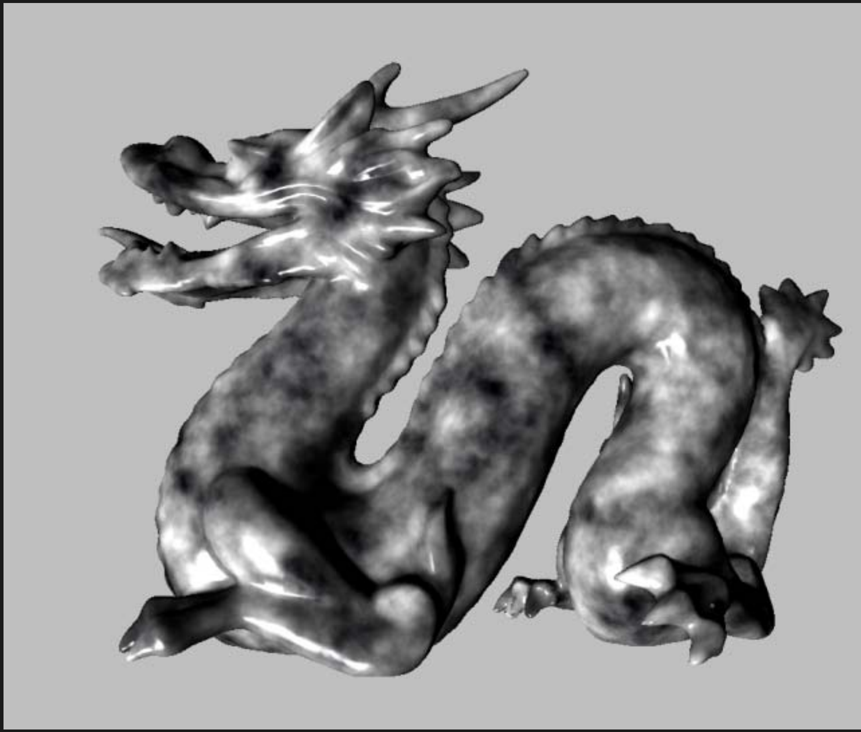

2rd Pass
Recompute cache payload
Output cache payload/depth
3rd Pass
Fetch current cache payload
Compute shading using payload
Output pixel color/Z

+80 +4 . : ,,. +5 × +4 . : ,,



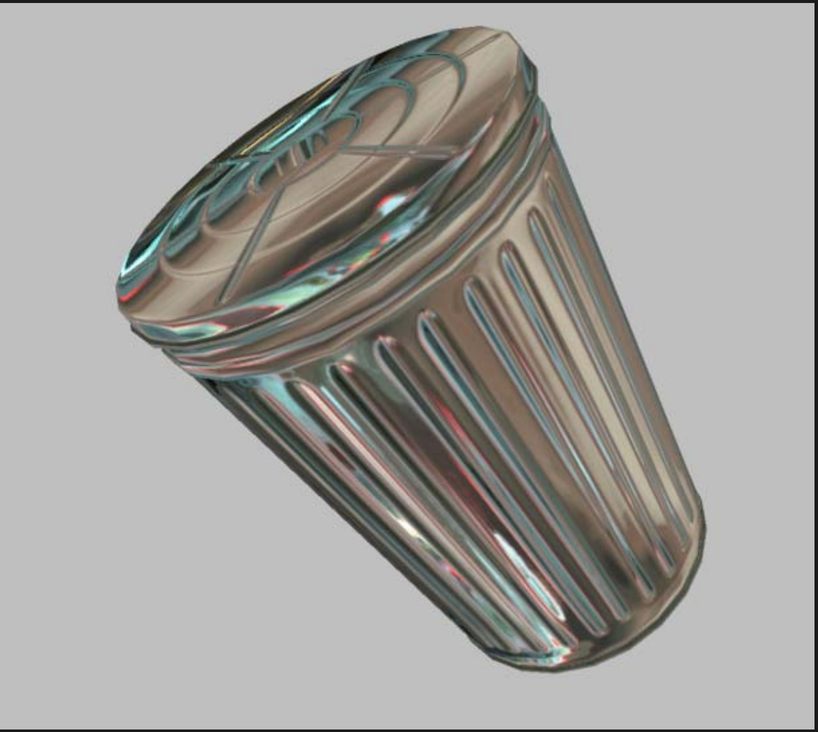
Test Scenes

Dragon shader



procedural noise with
Blinn-Phong specular layer
(75K triangles)

Trashcan shader

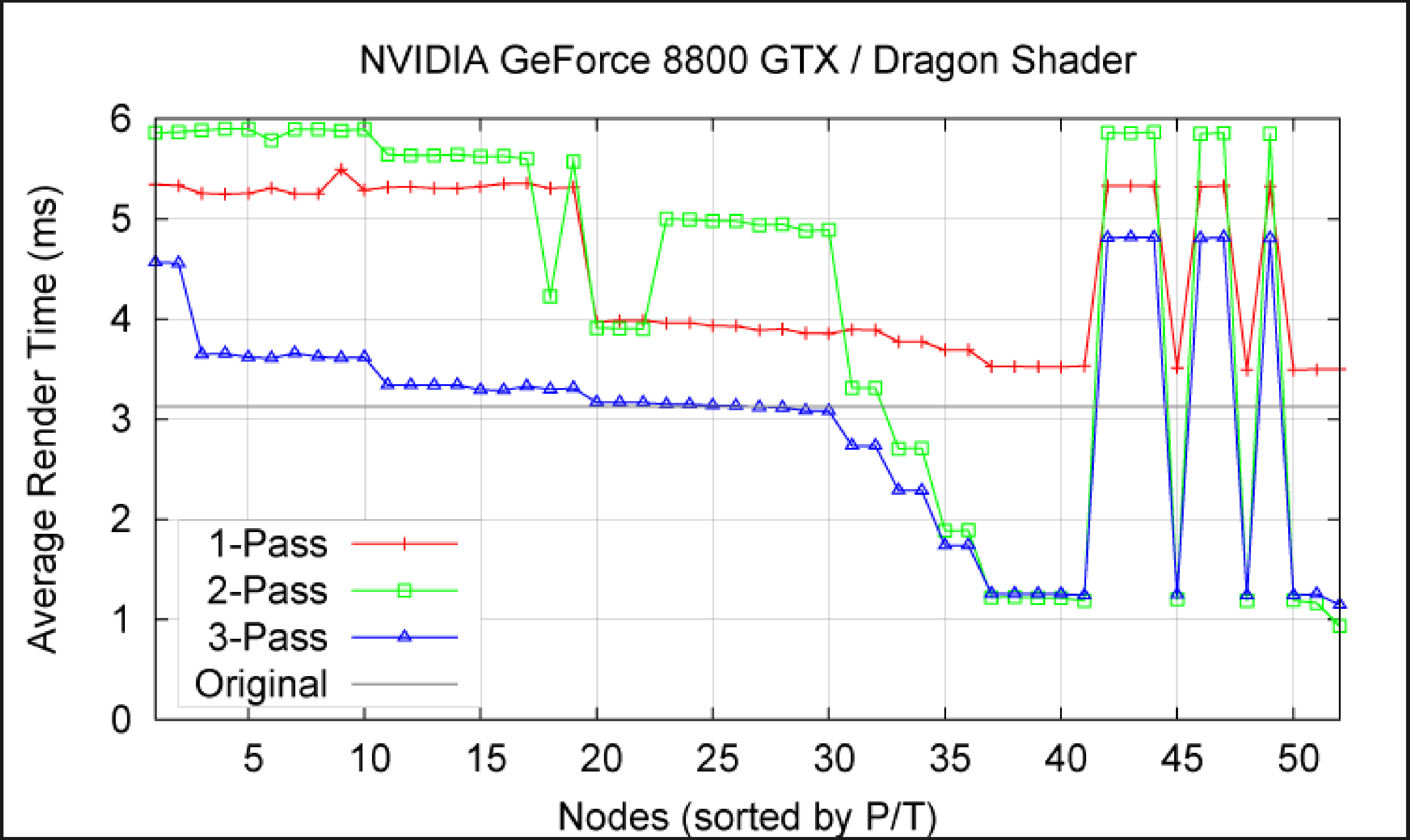


supersampled (25)
environment map
(15K triangles)

Experiment #1

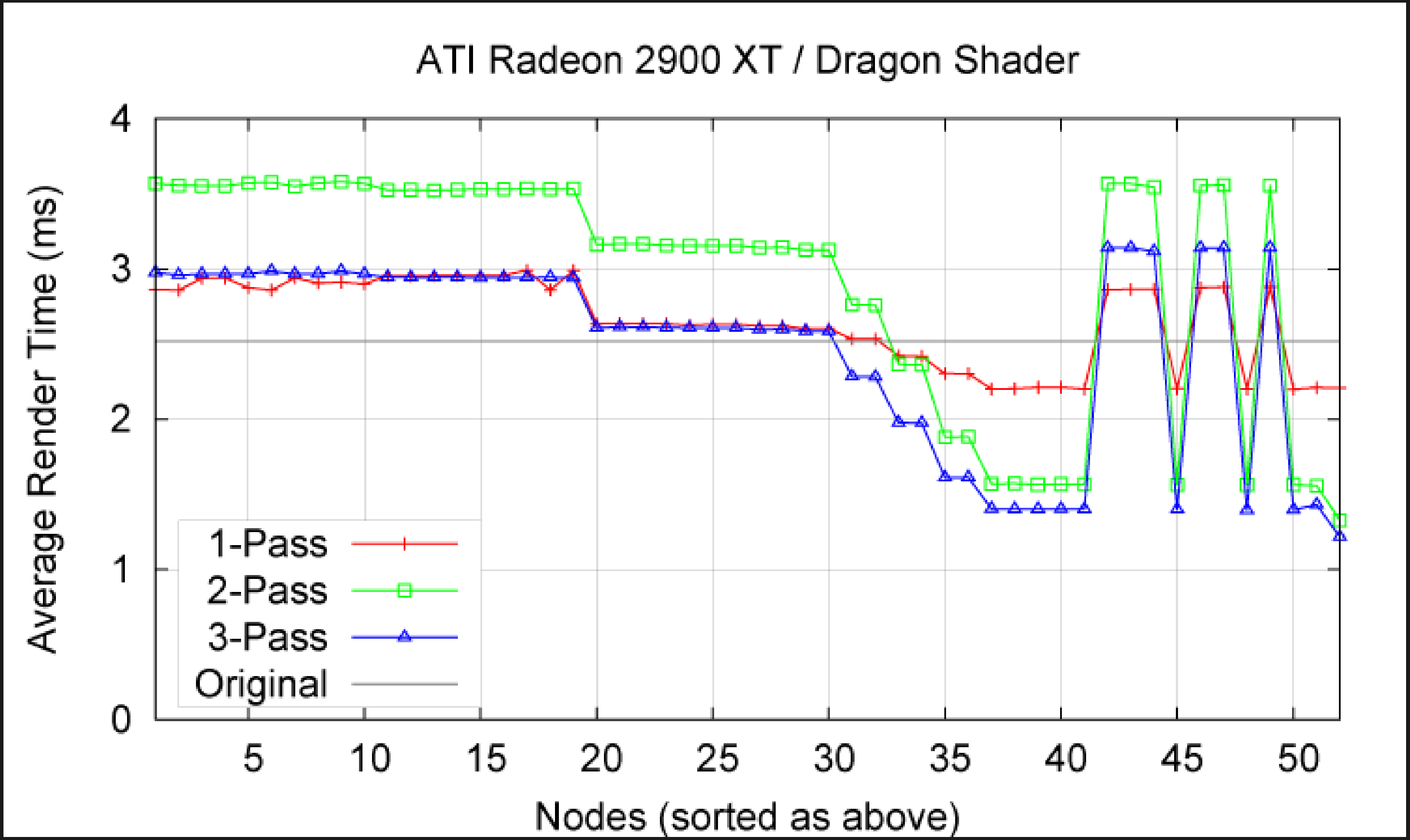
- Generated versions of the shader that caches every intermediate calculation
- Compute cost of evaluating payload (P)
- Compute cost of evaluating full shader (T)
- Fixed refresh period of 32 and 4 x 4 block size
- Compare performance of three different algorithms on NVIDIA Geforce 8800GTX and ATI Radeon 2900TX

Experiment #1: Dragon / NVIDIA



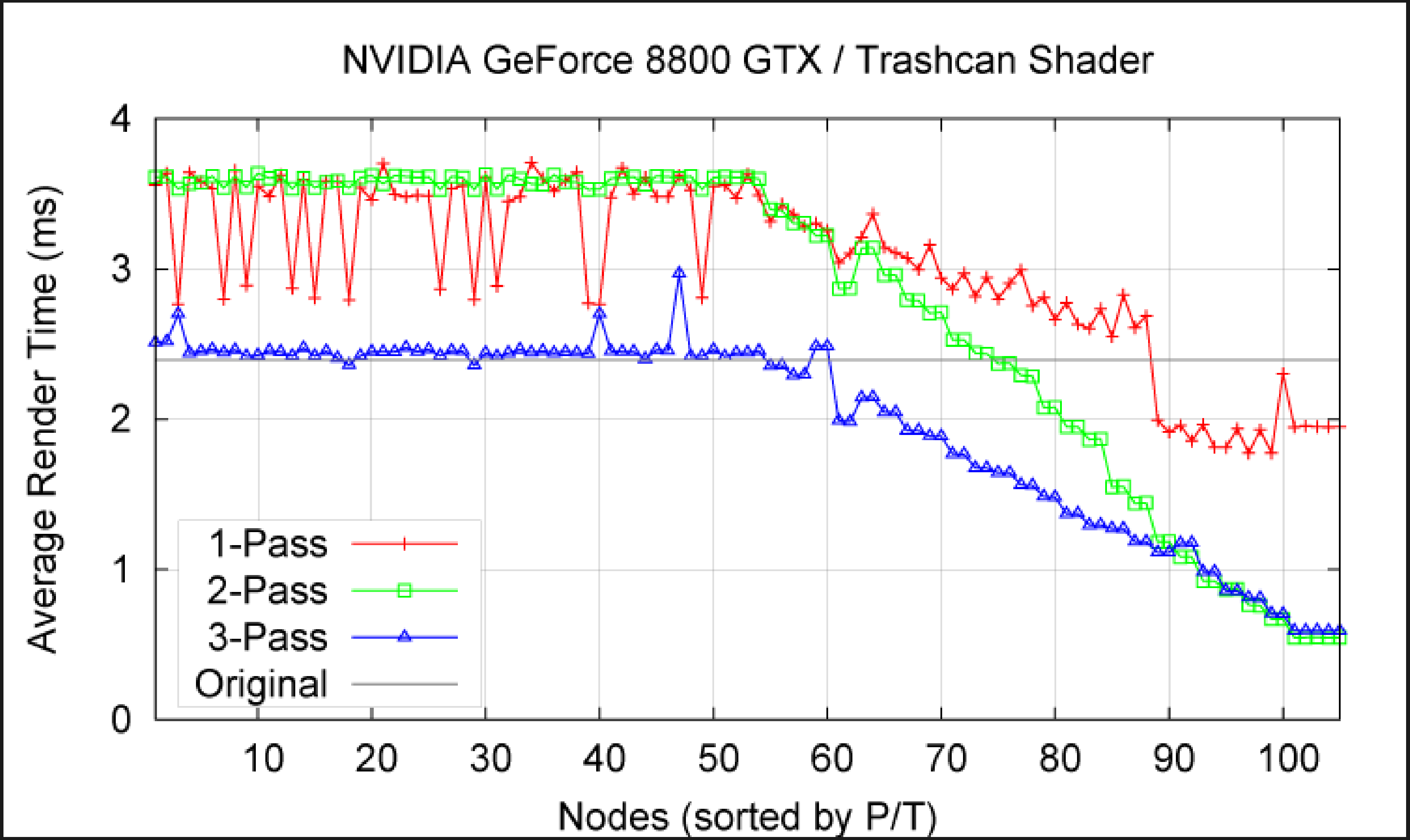
P/T increasing

Experiment #1: Dragon / ATI



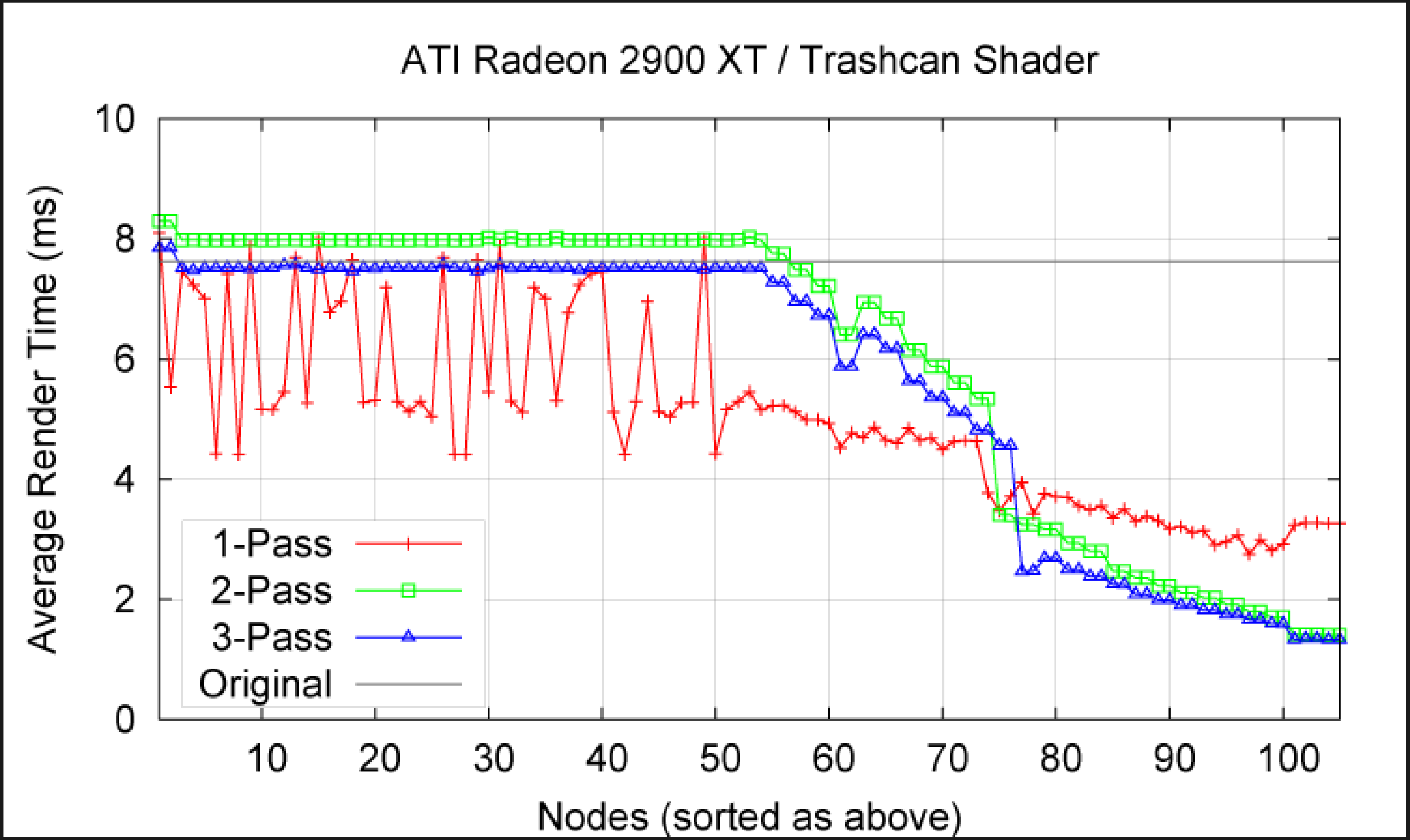
P/T increasing

Experiment #1: Trashcan / NVIDIA



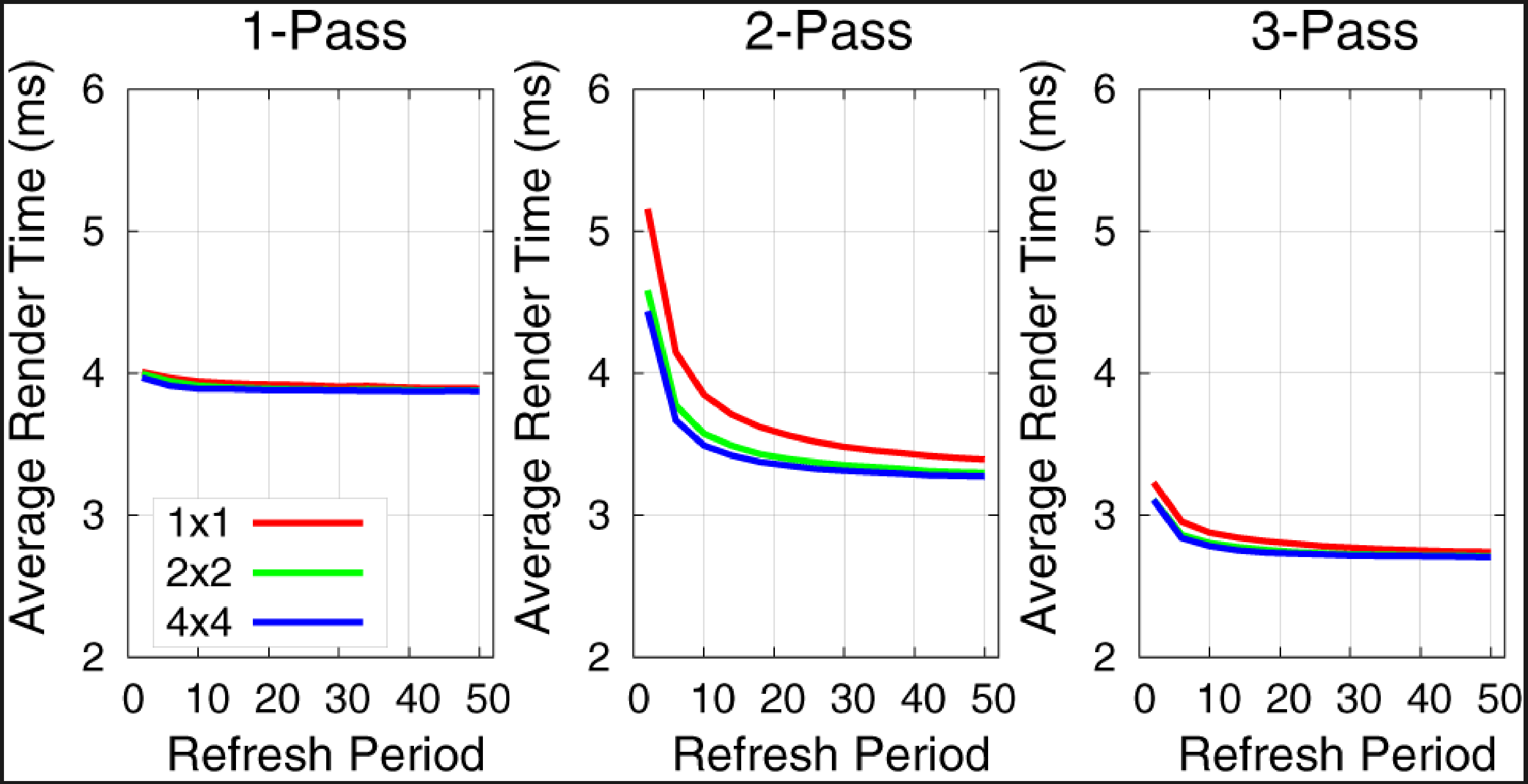
P/T increasing

Experiment #1: Trashcan / ATI



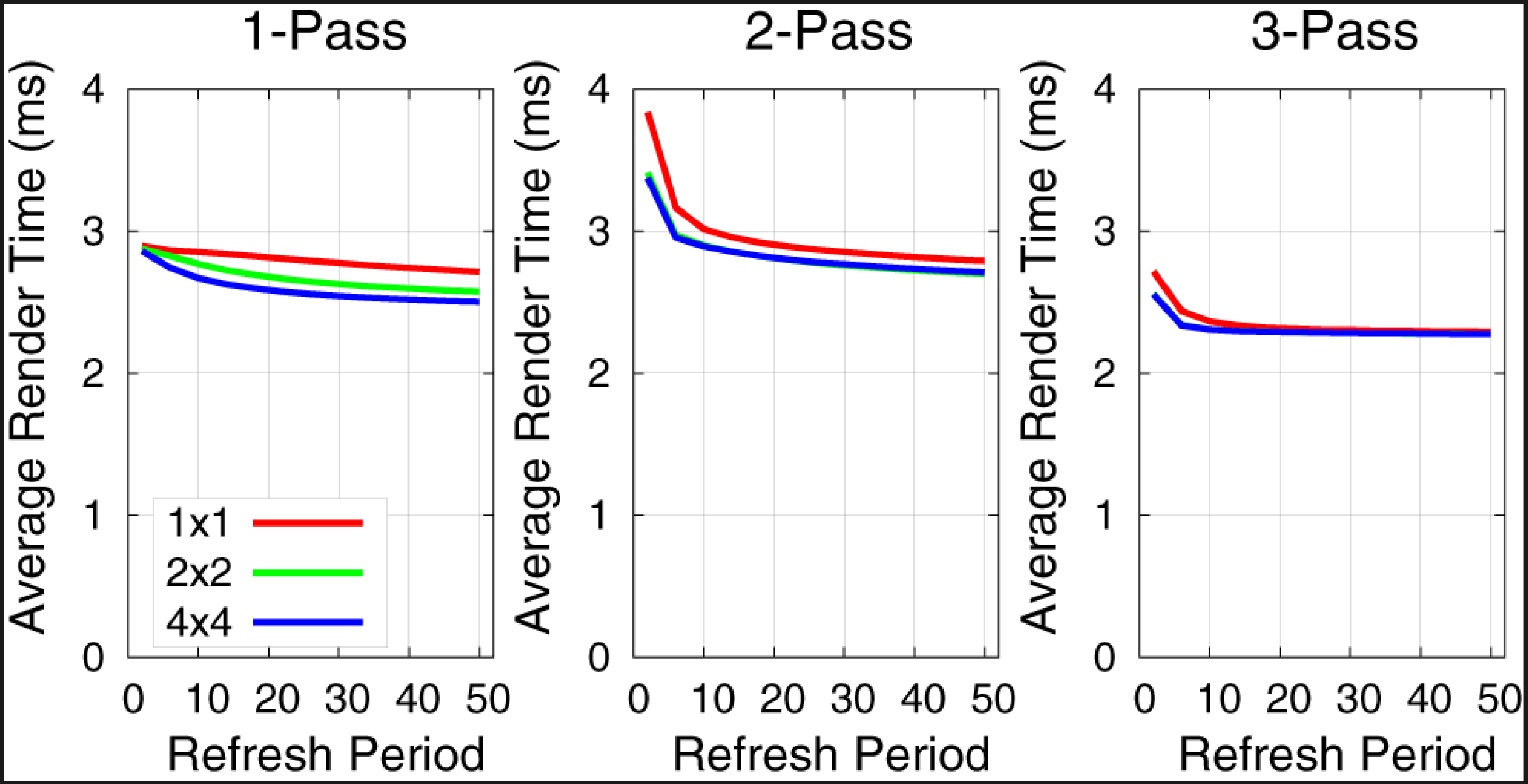
P/T increasing

Experiment #2: Refresh parameters



Dragon Shader: NVIDIA Geforce 8800GTX

Experiment #2: Refresh parameters



Dragon Shader: ATI 2900TX

Conclusion

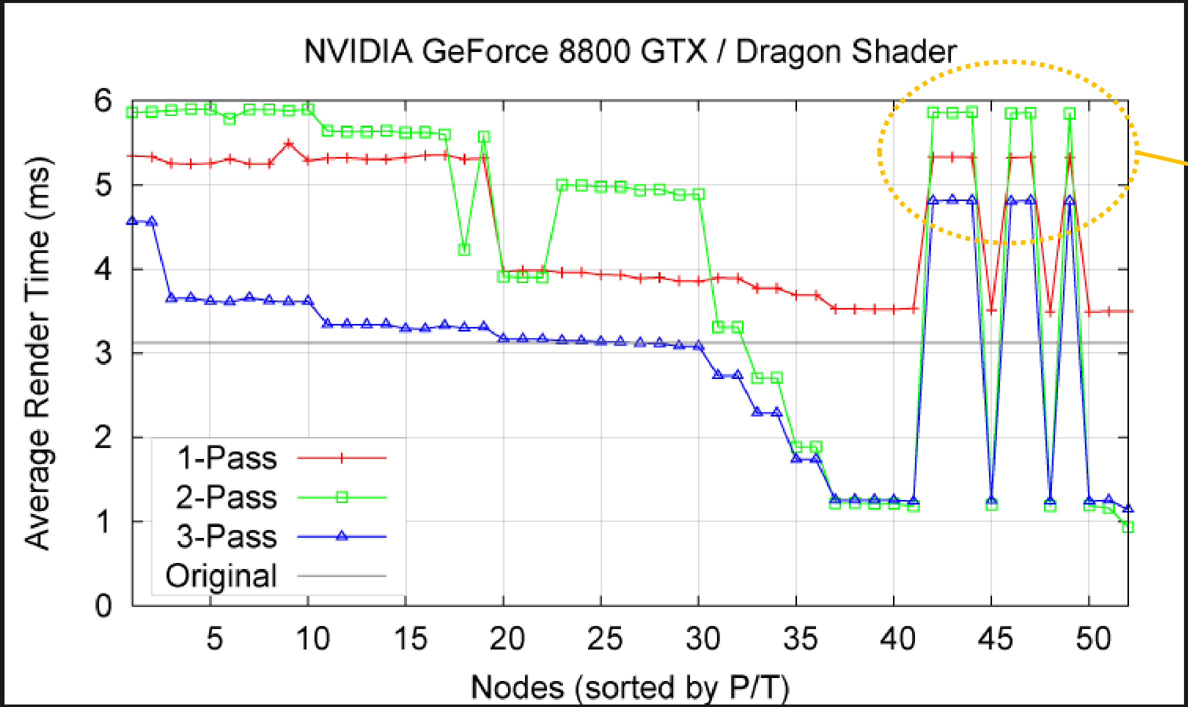
- Introduced an improved implementation of a shading reprojection cache
- Require single target and limits reliance on efficient branching in hardware
- More consistent performance across a wide range of cache loads on modern NVIDIA and ATI hardware

Future Work

- Explore the possibility of combing existing acceleration techniques
- Automatic cache allocation
- Alternative cache parameterization

Thank You

Elaboration on Experiment #1 Results



αnoise()+(1-α)noise()

Imagine caching αnoise() subexpression, noise() would need to be called in both hit and miss paths.