Efficient Spatial Binning on the GPU

Parallel Computing for Graphics: Beyond Programmable Shading

Christopher Oat | SIGGRAPH ASIA 2008

AMD
The future is fusion

Introduction

- Sorting random point data into bins/buckets
  - Unsorted input
  - Points binned in sorted order

- This is a key operation in spatial data structure construction

Motivation

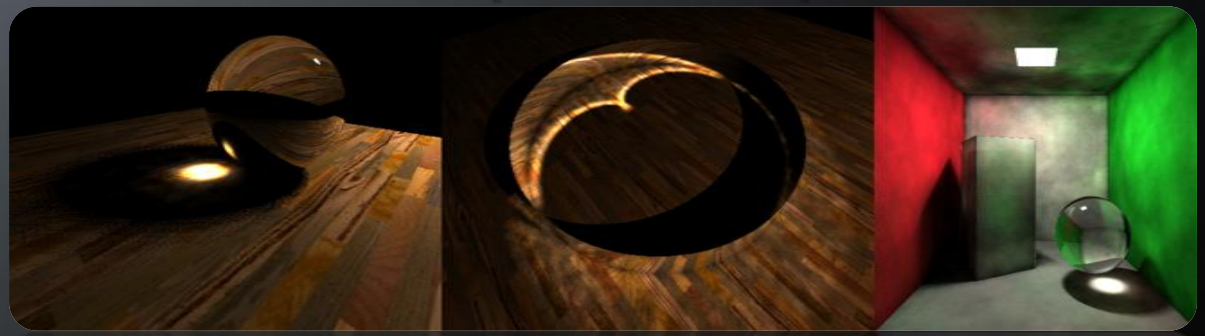
Many GPU applications need items sorted into bins/buckets:

Motivation

Many GPU applications need items sorted into bins/buckets:

- **Photon Mapping**

[Purcell *et al.* 2003]

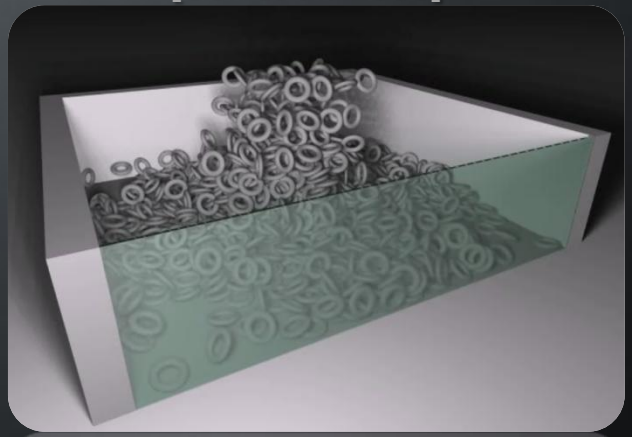


Motivation

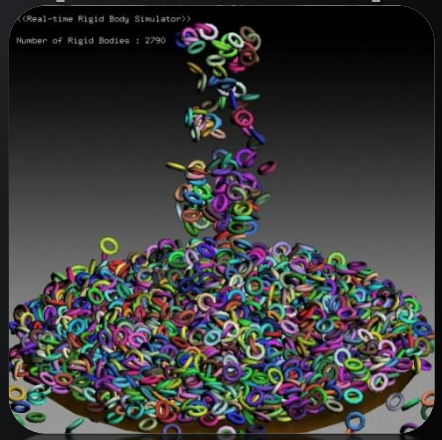
Many GPU applications need items sorted into bins/buckets:

- Photon Mapping
- **Rigid body simulation**

[Bell *et al.* 2005]



[Harada *et al.* 2007]



Motivation

Many GPU applications need items sorted into bins/buckets:

- Photon Mapping
- Rigid body simulation
- **Path finding**

[Shopf *et al.* 2008]

Motivation

Many GPU applications need items sorted into bins/buckets:

- Photon Mapping
- Rigid body simulation
- Path finding
- **Particle simulation**

Motivation

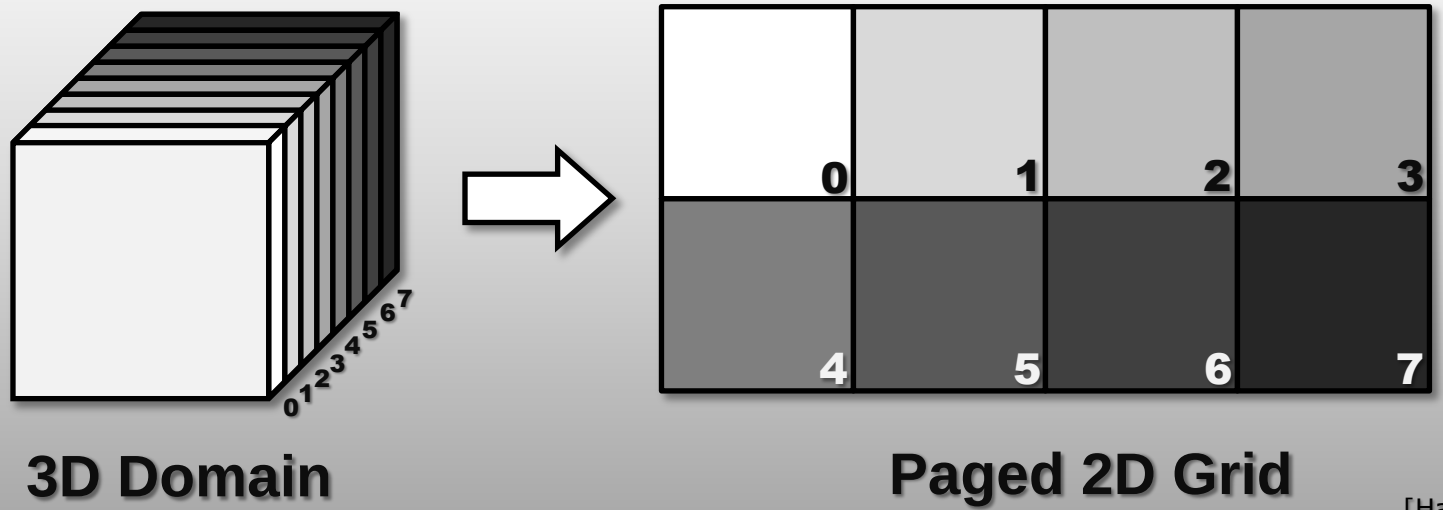
Many GPU applications need items sorted into bins/buckets:

- Photon Mapping
- Rigid body simulation
- Path finding
- Particle simulation

Previous approaches:

- CPU/GPU Hybrid
- Stencil Routing
- Others… require special compute API

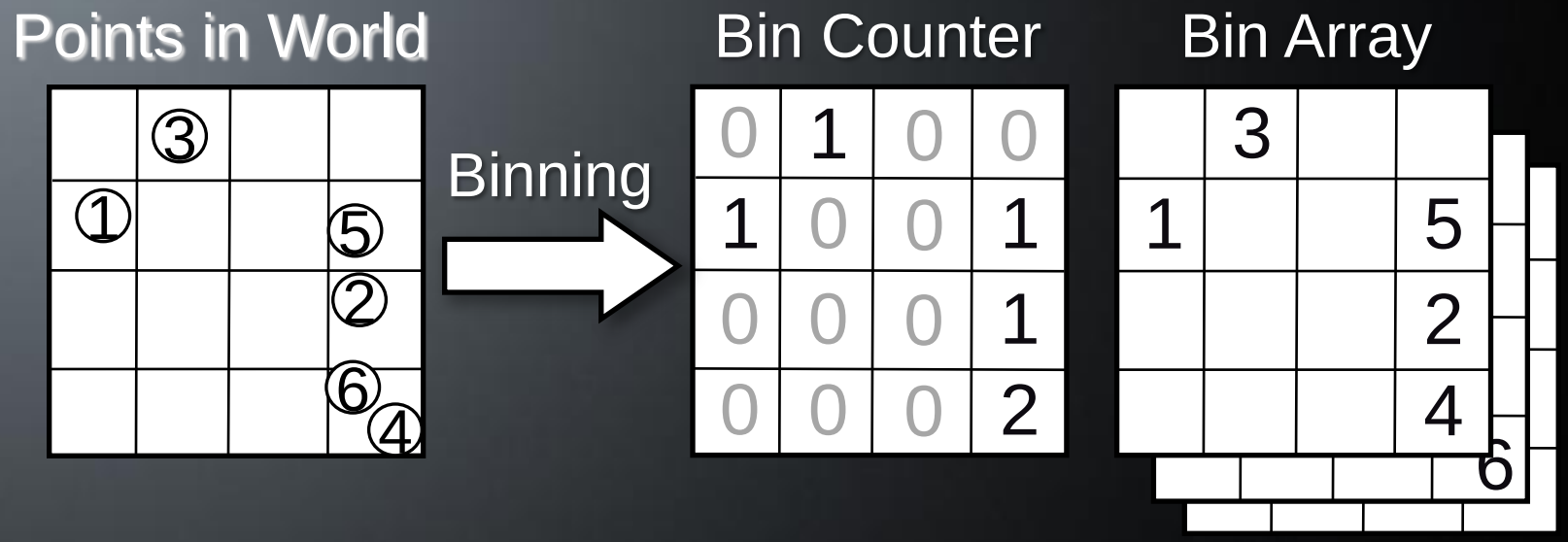
Grid-Based Spatial Data Structure



[Harris *et al.* 2003]

- Conceptually, very straightforward
- 1D, 2D & 3D domains all map to paged 2D grid
- Good for uniformly distributed data
  - Can be wasteful otherwise
  - Use a spatial hash tailored to your expected distribution
- How to fill grid without using atomics?

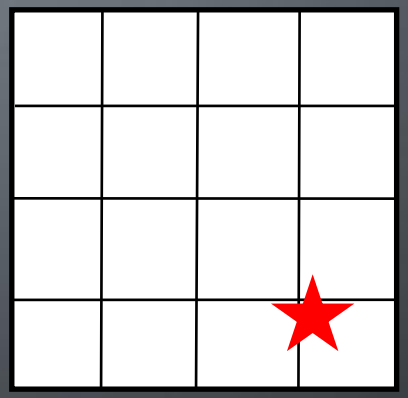
Bins on the GPU



- World-space position mapped to 2D grid index
- *Bin Counter* = color buffer, tracks bin load
- *Bin Array* = depth texture array, binned item IDs

Spatial Queries

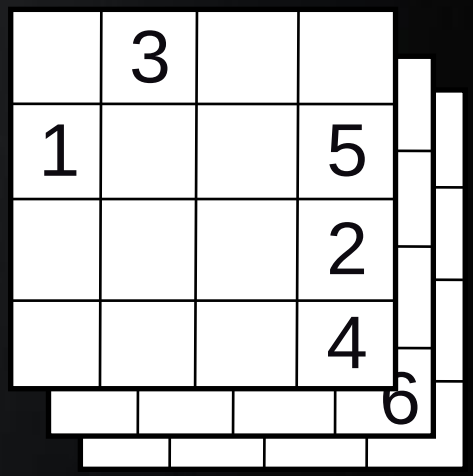
Query Point



Bin Counter

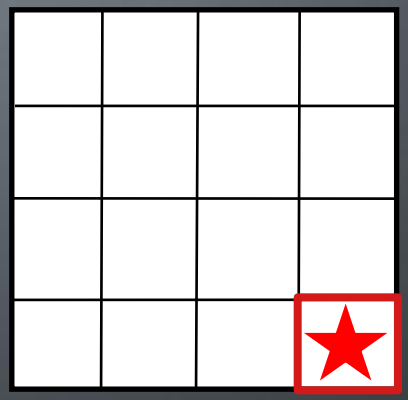
0	1	0	0
1	0	0	1
0	0	0	1
0	0	0	2

Bin Array



Spatial Queries

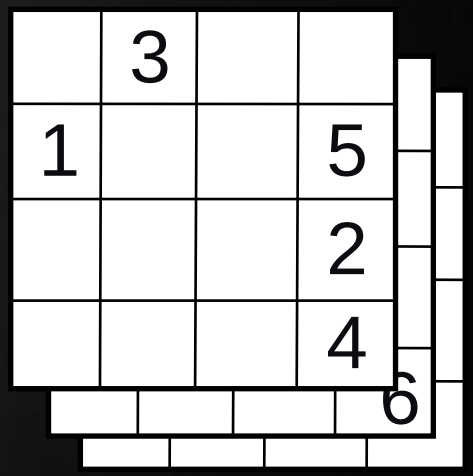
Query Point



Bin Counter

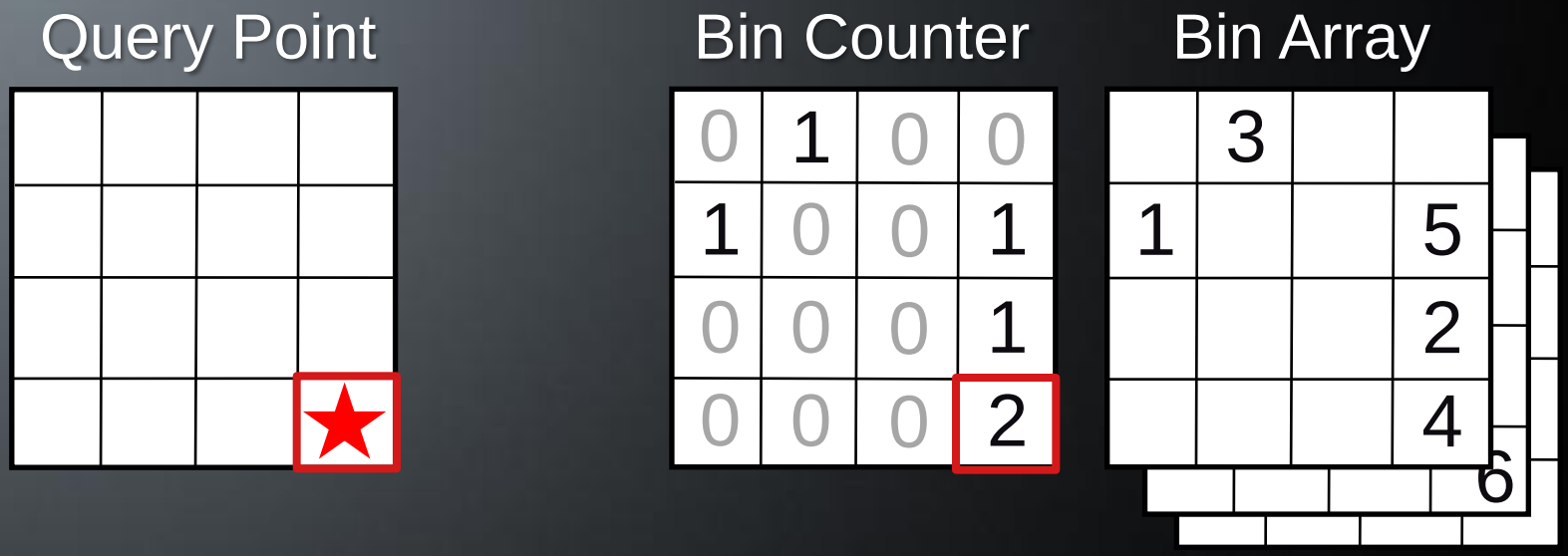
0	1	0	0
1	0	0	1
0	0	0	1
0	0	0	2

Bin Array



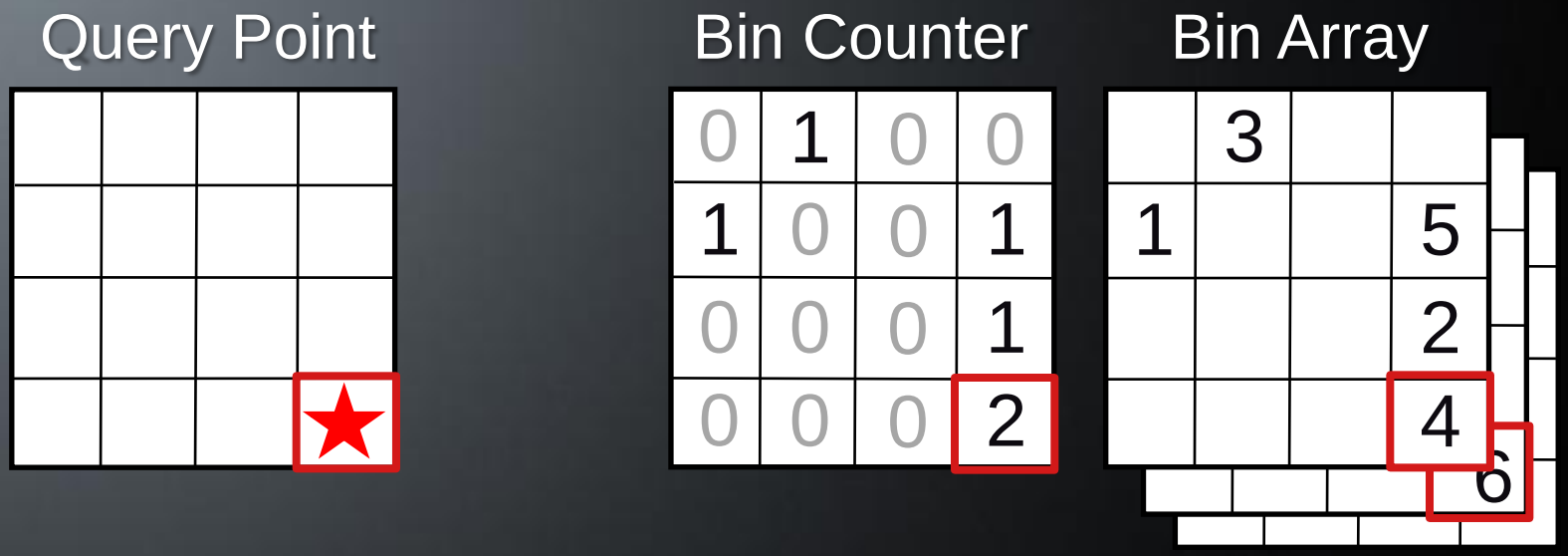
- Transform point to 2D grid index

Spatial Queries



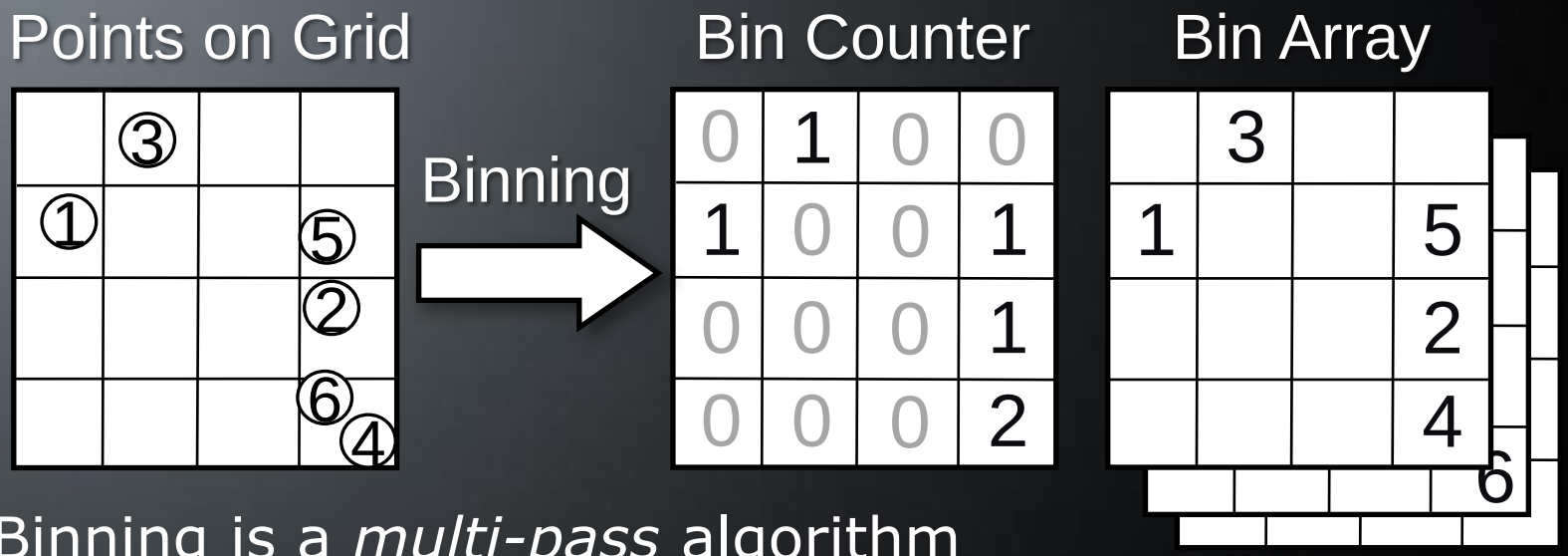
- Transform point to 2D grid index
- Fetch bin load from *Bin Counter*

Spatial Queries



- Transform point to 2D grid index
- Fetch bin load from *Bin Counter*
- Fetch item IDs from *Bin Array*

Updating Data Structure: Overview



- Binning is a *multi-pass* algorithm
  - Update one slice of *Bin Array* per pass
  - Once binned, points are removed from working set
  - Continue until working set is empty
  - Overflow is possible & detectable

Updating Data Structure: Initialization

Initialize data structure

- Clear *Bin Counter* to 0
- Clear *Bin Array* to **`MAX_DEPTH`**
- Working set is array of ***all*** item IDs

Updating Data Structure: Pass 1

- Bind bin counter & first slice of bin array
- Draw working set as point primitives
- Vertex shader:
  - Map item position to 2D bin array index
  - Set point's depth to normalized item ID
- Pixel shader: output "1"
- Depth test: **`LESS_THAN`**

Updating Data Structure: Pass 2…n

- Next slice of bin array bound as depth buffer
- VS: Sample ID from previous slice of bin array
  - Reject points less than or equal to previous
- GS: stream out non-rejected points
- PS: write pass number
- Depth test: **`LESS_THAN`**

Results of Pass 2..n

- VS test ensures only points that haven't yet been binned get streamed-out and rasterized
- Depth test ensures the point with lowest ID gets binned
- Results in points binned in sorted order
  - Like depth peeling
- Stream-out buffer becomes new working set

Outer Loop Termination

- Need to halt algorithm once all items are binned
  - Do not query size of stream-out buffer
  - CPU/GPU synchronization results in stalls
- Would like GPU to control execution

Avoiding Synchronization Stalls

We know max number of iterations

- Number of bin array slices

Make all the draw calls for the max number of iterations

- Use cascading predicated draw calls
- Use along with "`DrawAuto`" to issue draws
- Predicate on number of stream-out elements

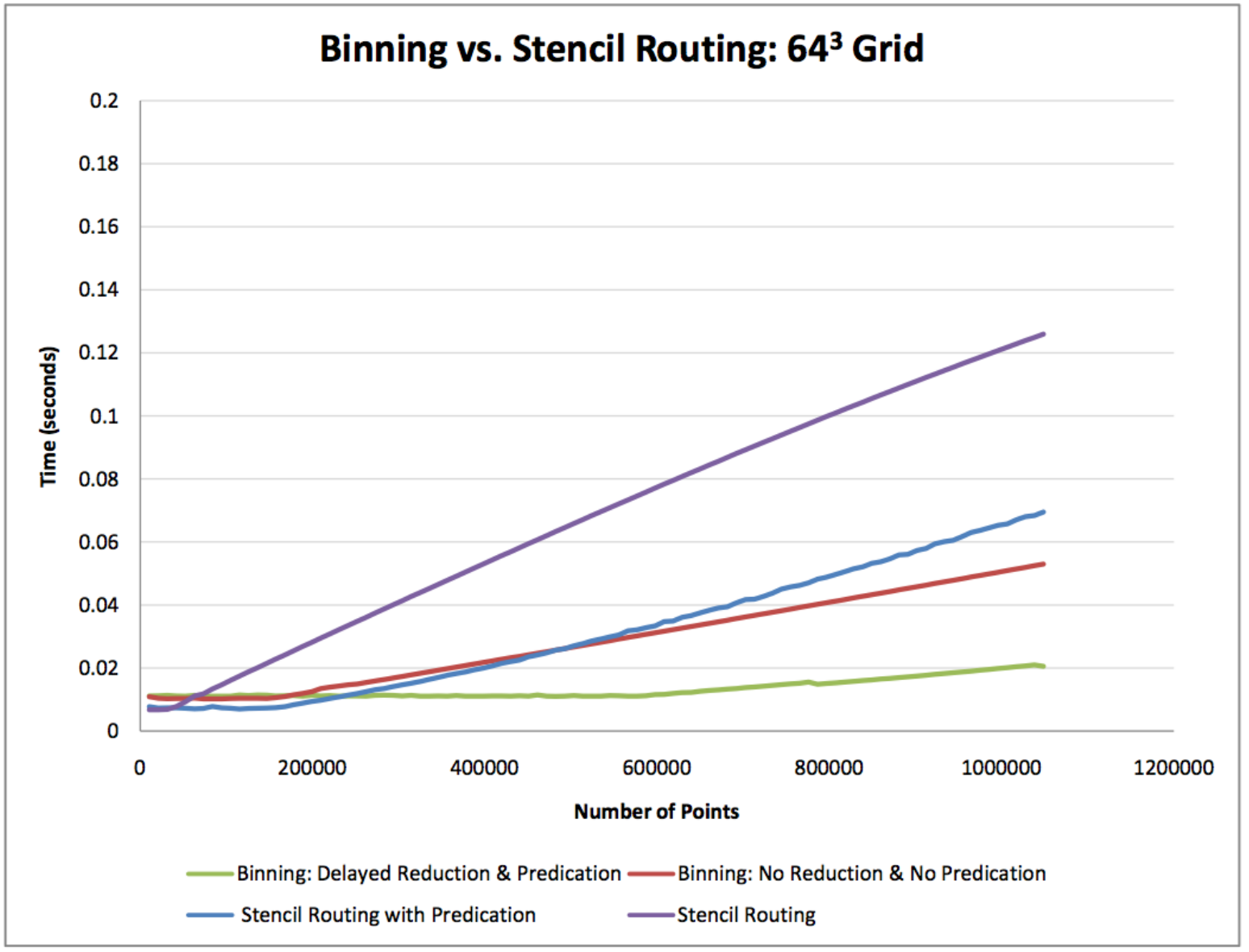
Delayed Reduction

Reduction uses stream out bandwidth

- If only a few points are binned in first few passes…
- Bandwidth usage is heavy
- Streaming out almost entire working set

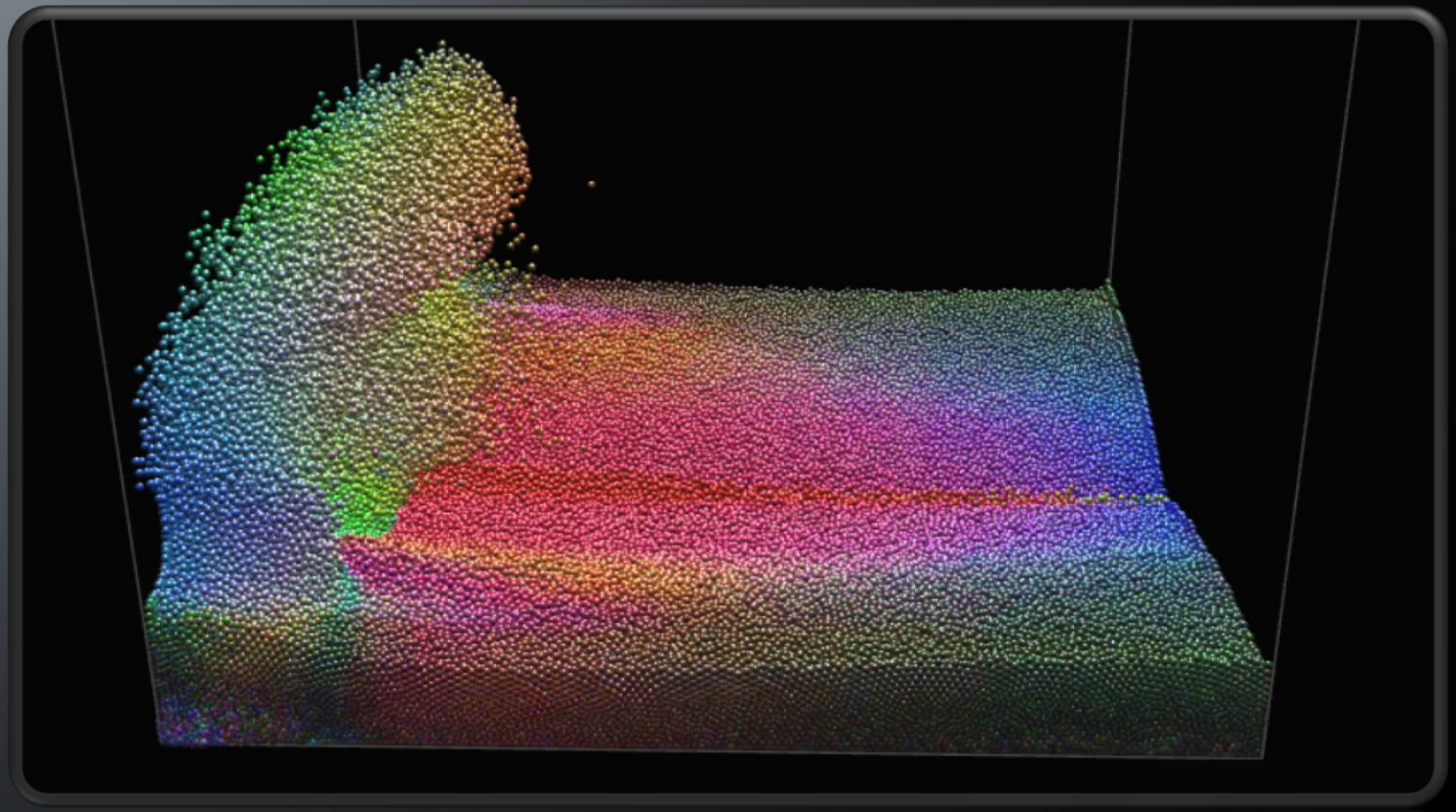
Use delayed reduction

- Begin reducing after several iterations
- We have found delay of E[Bin Load] works well



Configuration: AMD reference platform with AMD Athlon™ 64 X2 Dual-Core Processor 4600+, 2.40GHz, 2GB RAM. GPU: ATI Radeon™ HD 4870 Graphics. Motherboard: ASUSTek M2R32-MVP. Memory: DDR2-800 400 MHz. Operating System: Windows Vista® SP1.

GPU-Particle Demo



Conclusion

GPU Binning

- Efficient queries
  - Early-out on empty bins, known bin load, sorted order
- Overflow detection
- Stream-out reduction of working set
- GPU termination control

Many applications

Questions?

Thank You!

Feel free to email me:

Chris.Oat@amd.com

AMD
The future is fusion

References

Bell, N., Yu, Y., and Mucha, P. J. 2005. Particle-Base Simulation of Granular Materials. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, New York, NY, USA, 77-86.

Harada, T., Tanaka, M., Koshizuka, S., and Kawaguchi, Y. 2007. *Acceleration of Rigid Body Simulation using Graphics Hardware*. Symposium on Interactive 3D Graphics and Games, Seattle, April 30 – May 2, 2007.

Harris, M. J., Baxter, W. V., Scheuermann, T., and Lastra, A. 2003. Simulation of Cloud Dynamics on Graphics Hardware. In *HWWS '03: Proceedings of the ACM SIGGRAPH/Eurographics Conference of Graphics Hardware, Eurographics Association, Aire-la-Ville, Switzerland, 92-101.*

Purcell, T. J., Donner, C., Commarano, M., Jensen, H. W., and Hanrahan, P. 2003. Photon Mapping on Programmable Graphics Hardware. In *Proceedings of the ACM SIGGRAPH/Eurographics Conference on Graphics Hardware*, Eurographics Association, 41-50.

Shopf, J., Barczak, J., Oat, C., and Tatarchuk, N. 2008. March of the Froblins: Simulation and Rendering Massive Crowds on Intelligent and Detailed Creatures on the GPU. In *SIGGRAPH '08: ACM SIGGRAPH 2008 classes*, ACM, New York, NY, USA, 52-101.

Yasuda, R., Harada, T., Kawaguchi, Y. 2008. Real-Time Simulation of Granular Materials Using Graphics Hardware. *Fifth International Conference on Computer Graphics, Imaging and Visualization*, 28-31.

Trademark Attribution

AMD, the AMD Arrow logo and combinations thereof are trademarks of Advanced Micro Devices, Inc. in the United States and/or other jurisdictions. Other names used in this presentation are for identification purposes only and may be trademarks of their respective owners.

AMD
The future is fusion