Vectorized Production Path Tracing

Mark Lee
Brian Green
Feng Xie
Eric Tabellion

DreamWorks Animation
Hello MoonRay
Hello MoonRay
MoonRay Overview

- Single executable with internal scalar and vectorized code paths
- Both code paths produce identical output images
Data Oriented Design

- Nothing new, AAA game studios have always done this
- Approach the problem from the point of view of the hardware
  - ... which means caring greatly about data access patterns
- Lower level data structures are flat and uncomplicated
How would it change your approach to coding if main memory access was 10x slower?
Threading
Keep all vector lanes of all cores busy all the time with meaningful work!
Keep all vector lanes busy....

- How can we use the vector hardware effectively?
- How can we gather batches of work effectively?
- How can we minimize control flow divergence in vector code?
- How can we access memory effectively?
Will the performance gains outweigh the additional work?
AOS Format

SOA Format
AOS to SOA Conversion
ISPC

- ISPC = Intel SPMD Program Compiler
- Can target multiple vector architectures, e.g. SSE, AVX, AVX512
- Automatic masking
- C like syntax
Queuing

- Strategy - use “queues” to build up large coherent batches of work
- Queues entries typically consist of a 32-bit index and a 32-bit “sort key”
- Each queue has an associated handler to process entries
When a queue becomes full, the thread which filled it flushes it
Scalar Path Tracing
Scalar Path Tracing

Generate primary rays
Scalar Path Tracing

Generate primary rays

Trace primary ray
Scalar Path Tracing

Generate primary rays

Trace primary ray

Shader graph eval at hit point
Texture eval at hit point
Integration at hit point
Spawn and trace secondary rays
Spawn and trace occlusion rays
Accumulate radiance
Scalar Path Tracing

Generate primary rays

Trace primary ray

Shader graph eval at hit point
Texture eval at hit point
Integration at hit point
Spawn and trace secondary rays
Spawn and trace occlusion rays
Accumulate radiance

Frame Buffer
Vectorized Path Tracing
Vectorized Path Tracing
Vectorized Path Tracing

Primary Ray Queue Handler

Generate primary rays

Primary Ray Queue
Vectorized Path Tracing

- Generate primary rays
- Primary Ray Queue
- Primary Ray Queue Handler
- Shade Queue (shared)
Vectorized Path Tracing

Primary Ray Queue Handler

- Generate primary rays
- Primary Ray Queue
- Shade Queue (shared)
- Radiance Queue
Vectorized Path Tracing
Vectorized Path Tracing
Shade Queue Handler

- Radix sort queue entries
- Vectorized shader graph eval
- Vectorized texture eval
- Vectorized integration
Shade Queue Handler

- Radix sort queue entries
- Transform AOS entries to SOA
- Vectorized shader graph eval
- Vectorized texture eval
- Vectorized integration
- Transform SOA entries back to AOS
Shade Queue Handler

- Radix sort queue entries
- Transform AOS entries to SOA
- Vectorized shader graph eval
- Vectorized texture eval
- Vectorized integration
- Transform SOA entries back to AOS

Entry sort criteria
1. UDIM tile
2. Mip-level
3. Uv coordinates
Radix sort queue entries
Transform AOS entries to SOA
Vectorized shader graph eval
Vectorized texture eval
Vectorized integration
Transform SOA entries back to AOS
Shade Queue Handler

- Radix sort queue entries
- Transform AOS entries to SOA
- Vectorized shader graph eval
- Vectorized texture eval
- Vectorized integration
- Transform SOA entries back to AOS

- 100% ISPC code
- Optional JIT compilation via LLVM
- Returns a shader closure in SOA format
Shade Queue Handler

- Radix sort queue entries
- Transform AOS entries to SOA
- Vectorized shader graph eval
- Vectorized texture eval
- Vectorized integration
- Transform SOA entries back to AOS

- Custom OIIIO vectorized texture sampler
- Custom set-associative cache built on top of main OIIIO tile cache
Shade Queue Handler

- Radix sort queue entries
- Transform AOS entries to SOA
- Vectorized shader graph eval
- Vectorized texture eval
- Vectorized integration
- Transform SOA entries back to AOS

• Ability to add to arbitrary queues
• Supports arbitrary ray spawning
Shade Queue Handler

- Radix sort queue entries
- Transform AOS entries to SOA
- Vectorized shader graph eval
- Vectorized texture eval
- Vectorized integration
- Transform SOA entries back to AOS
Vectorized Path Tracing
Vectorized Path Tracing

- Primary Ray Queue Handler
  - Generate primary rays

- Shade Queue Handler
  - Radix sort queue entries
  - Vectorized shader graph eval
  - Vectorized texture eval
  - Vectorized integration

- Radiance Queue Handler

- Primary Ray Queue
- Shade Queue (shared)
- Radiance Queue
- Frame Buffer
Vectorized Path Tracing

- **Primary Ray Queue Handler**
  - Generate primary rays

- **Shade Queue Handler**
  - Radix sort queue entries
  - Vectorized shader graph eval
  - Vectorized texture eval
  - Vectorized integration

- **Primary Ray Queue**
- **Shade Queue (shared)**
- **Incoherent Ray Queue**
- **Radiance Queue**
- **Radiance Queue Handler**
- **Frame Buffer**
Vectorized Path Tracing

- Primary Ray Queue Handler
  - Generate primary rays
- Shade Queue Handler
  - Radix sort queue entries
  - Vectorized shader graph eval
  - Vectorized texture eval
  - Vectorized integration
- Incoherent Ray Queue Handler
- Radiance Queue Handler
  - Radiance Queue
  - Frame Buffer

- Primary Ray Queue
- Shade Queue (shared)
- Incoherent Ray Queue
- Radiance Queue
Vectorized Path Tracing
Vectorized Path Tracing

- Primary Ray Queue Handler
  - Generate primary rays
  - Primary Ray Queue

- Shade Queue Handler
  - Radix sort queue entries
  - Vectorized shader graph eval
  - Vectorized texture eval
  - Vectorized integration
  - Shade Queue (shared)

- Incoherent Ray Queue Handler
  - Incoherent Ray Queue

- Occlusion Ray Queue Handler
  - Occlusion Ray Queue

- Radiance Queue Handler
  - Radiance Queue

- Frame Buffer
  - Radiance Queue
Vectorized Path Tracing

generate primary rays

Primary Ray Queue Handler

Shade Queue Handler
- Radix sort queue entries
- Vectorized shader graph eval
- Vectorized texture eval
- Vectorized integration

Primary Ray Queue

Incoherent Ray Queue Handler

Incoherent Ray Queue

Occlusion Ray Queue Handler

Occlusion Ray Queue

Radiance Queue Handler

Radiance Queue

Frame Buffer
Vectorized Path Tracing

- Primary Ray Queue Handler
  - Generate primary rays
- Shade Queue Handler
  - Radix sort queue entries
  - Vectorized shader graph eval
  - Vectorized texture eval
  - Vectorized integration
- Incoherent Ray Queue Handler
- Occlusion Ray Queue Handler
- Radiance Queue Handler
- Primary Ray Queue
- Shade Queue (shared)
- Incoherent Ray Queue
- Occlusion Ray Queue
- Radiance Queue
- Frame Buffer

Vectorized Path Tracing
Will the performance gains outweigh the additional work?
Time

Scalar

Vectorized

3rd party libraries

3rd party libraries
Time

Scalar

3rd party libraries

Unsuitable for vectorization

Vectorized

3rd party libraries

Unsuitable for vectorization
Time

Scalar Code

Unsuitable for vectorization

3rd party libraries

Unsuitable for vectorization

3rd party libraries

Scalar

Vectorized
The diagram illustrates the comparison between scalar and vectorized code. It shows the time axis on the left and the scalar and vectorized code on the horizontal axis. The diagram highlights the following:

- **Scalar Code**: This section is unsuitable for vectorization and includes third-party libraries and queuing overhead.
- **Vectorized Code**: Despite being vectorized, it also involves third-party libraries and queuing overhead.

The diagram emphasizes the challenges of vectorization in the presence of third-party libraries and overhead, indicating that both scalar and vectorized code face similar issues.
- Scalar Code
- 3rd party libraries
- Unsuitable for vectorization

- Sorting Overhead
- Queuing Overhead
- Vectorized Code
- 3rd party libraries
- Unsuitable for vectorization
Time

Scalar Code

Unsuitable for vectorization

3rd party libraries

AOS/SOA Overhead

Sorting Overhead

Queuing Overhead

Vectorized Code

Unsuitable for vectorization

3rd party libraries
Results
Bergen Town

Integration

Texturing

Intersect incoherent

Shading

Vect overhead

Scalar

Vectorized

Legend:
- Red: Embree rtcIntersect primary
- Green: Other
- Blue: Differential geometry
- Pink: Vectorization overhead
- Purple: Embree rtcOccluded
- Light Green: Shading (excl. texturing)
- Orange: Texturing
- Dark Orange: Integration
- Light Blue: Embree rtcIntersect incoherent
Bergen Town Vectorization Speedups

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ray intersection subsystem speedup</td>
<td>1.20x</td>
</tr>
<tr>
<td>Shading subsystem speedup</td>
<td>6.19x</td>
</tr>
<tr>
<td>Texturing subsystem speedup</td>
<td>4.24x</td>
</tr>
<tr>
<td>Integration subsystem speedup</td>
<td>2.75x</td>
</tr>
<tr>
<td>Overall speedup</td>
<td>1.77x</td>
</tr>
</tbody>
</table>
Texturing
Integration
Intersect incoherent

Astrid

Scalar

Vectorized

Embree rtcIntersect primary
Other
Differential geometry
Vectorization overhead
Embree rtcOccluded
Shading (excl. texturing)
Texturing
Integration
Embree rtcIntersect incoherent
## Astrid Vectorization Speedups

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ray intersection subsystem speedup</td>
<td>1.00x</td>
</tr>
<tr>
<td>Shading subsystem speedup</td>
<td>4.54x</td>
</tr>
<tr>
<td>Texturing subsystem speedup</td>
<td>3.43x</td>
</tr>
<tr>
<td>Integration subsystem speedup</td>
<td>2.68x</td>
</tr>
<tr>
<td>Overall speedup</td>
<td>1.60x</td>
</tr>
</tbody>
</table>
# Hotspur Vectorization Speedups

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ray intersection subsystem speedup</td>
<td>1.14x</td>
</tr>
<tr>
<td>Shading subsystem speedup</td>
<td>4.20x</td>
</tr>
<tr>
<td>Texturing subsystem speedup</td>
<td>2.90x</td>
</tr>
<tr>
<td>Integration subsystem speedup</td>
<td>2.72x</td>
</tr>
<tr>
<td>Overall speedup</td>
<td>2.31x</td>
</tr>
</tbody>
</table>
Conclusion

- Significant speedups over scalar mode in all scenes tested
- Vectorization speedup is scene dependent
- Future looking architecture
  - Linear scaling with increased core counts
  - Non-trivial speedups with increased vector lane counts
  - Suitable for GPU hardware
Good Luck MoonRay!
Acknowledgements: We’d like to thank the MoonRay and MoonShine teams at DWA, and the open source community

Questions?

mark.lee@dreamworks.com

Full paper is available at: research.dreamworks.com