Reduced Precision Hardware for Ray Tracing

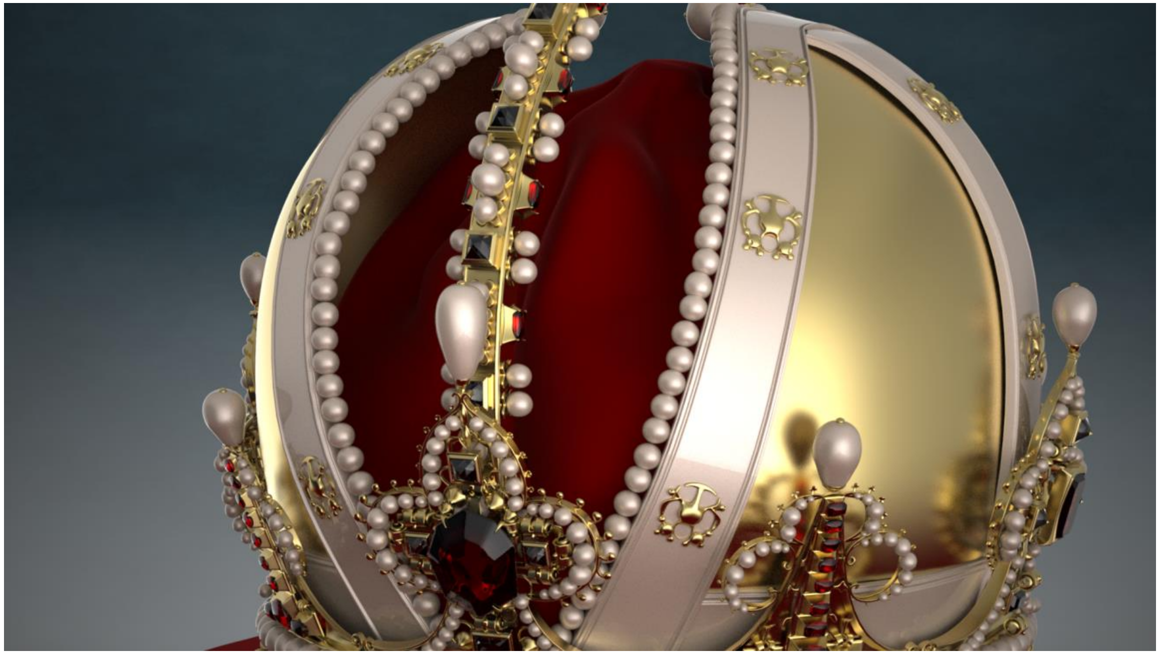
Sean Keely

University of Texas, Austin

Question

Why don’t GPU’s accelerate ray tracing?

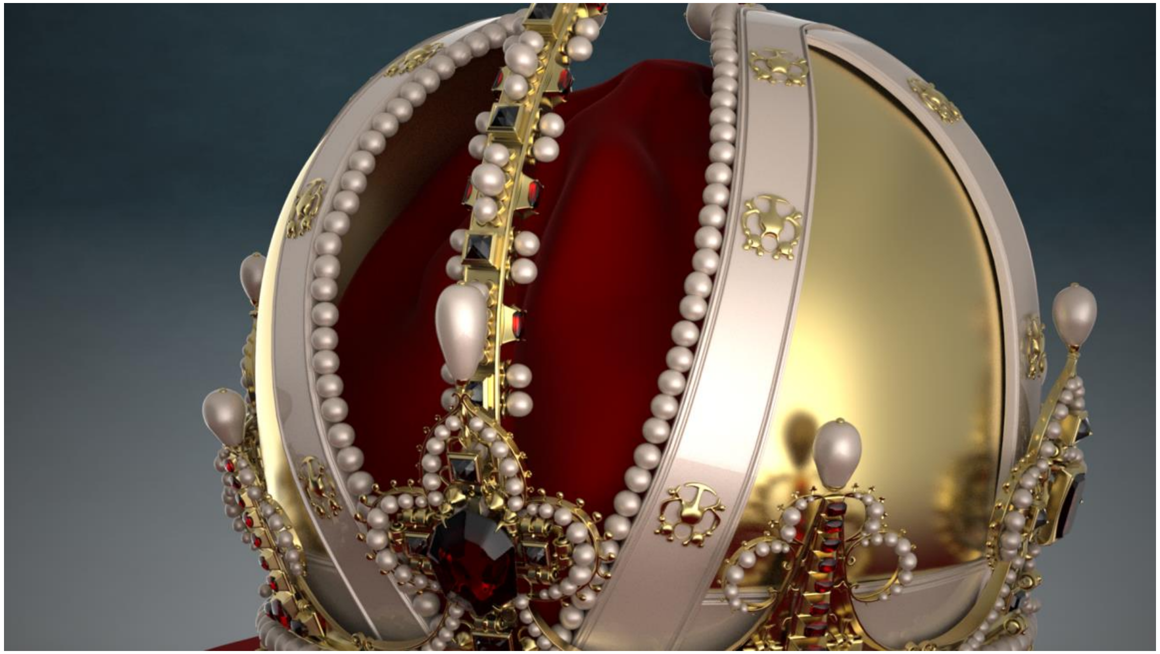
Real time ray tracing needs very high ray rate



Example Scene: 3 area lights + AO/GI

64 rays/pixel, 1080p, 30Hz -> **4 Billion Rays/s**

Real time ray tracing needs very high ray rate

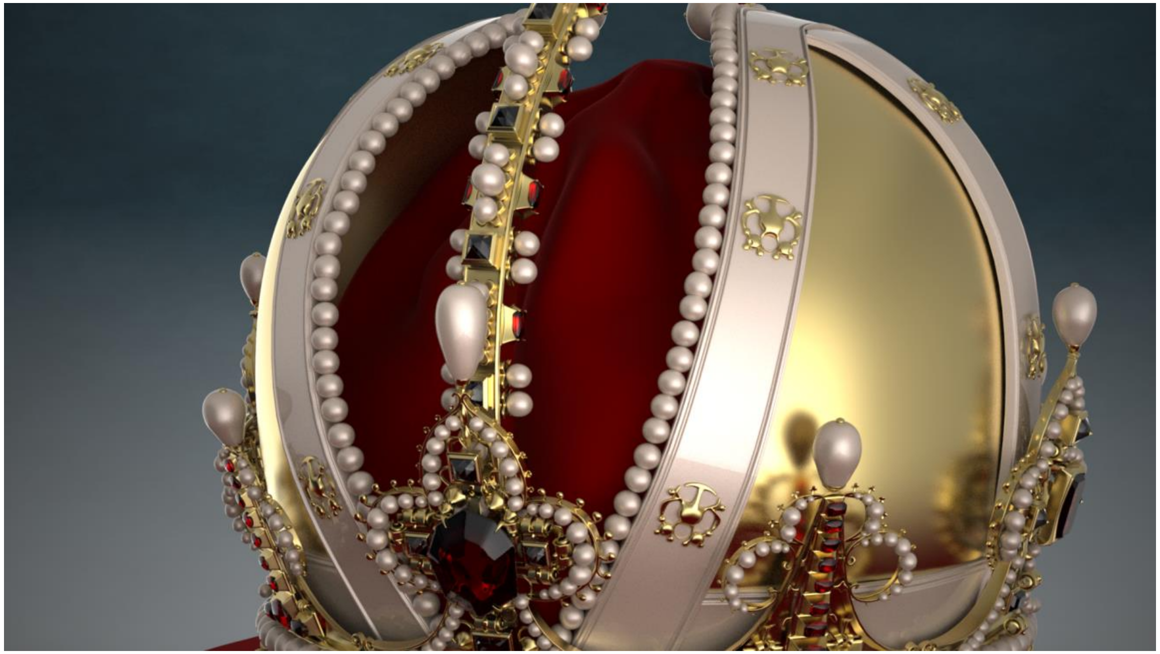


Example Scene: 3 area lights + AO/GI

64 rays/pixel, 1080p, 30Hz -> **4 Billion Rays/s**

Software approaches give 80-400M Rays/s

Single precision ASIC's will be large



Approx. 4 Tflops Trace + 2 Tflops Intersect + ?? Shading

Will need roughly the die area of a high end GPU for a trace & intersect co-processor

Previous work competes with current GPU’s

- STRaTA, D. Kopta… 2013
  - MIMD configurable pipeline, avoids divergence penalty
  - Repurpose L2 as ray cache

- SGRT, W. Lee… 2014
  - MIMD co-processor design
  - Most recent work addresses the performance cost of co-processor design

Goal: Accelerate ray tracing in a GPU, not next to one

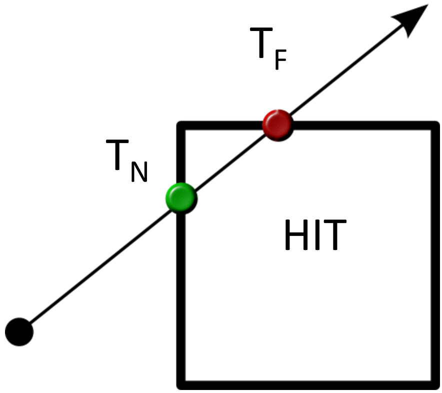
- Add high performance BVH traversal acceleration to current GPU architecture
  - MIMD traversal
  - SIMD programs
- Constraints: low impact, not a co-processor
  - small die area
  - low power
  - low bandwidth

How: Reduced Precision and Integration

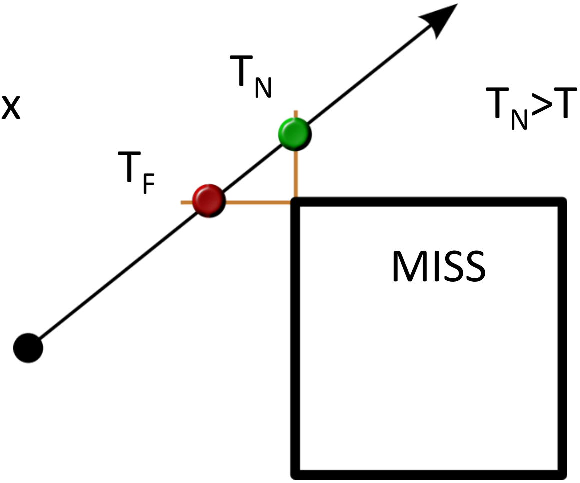
- 4 Billion RPS needs ~24W <u>just for multiplies</u> in traversal.
  - Can be reduced to ~1W with reduced precision.
- One off-chip data access is ~100x more energy than a FMA.
  - Reduced precision saves here too!
- Don’t build new registers, cache, wires…
  - Integrate to a GPU and get this for free!

Reduced Precision BVH Traversal

Robust Ray-AABB Test

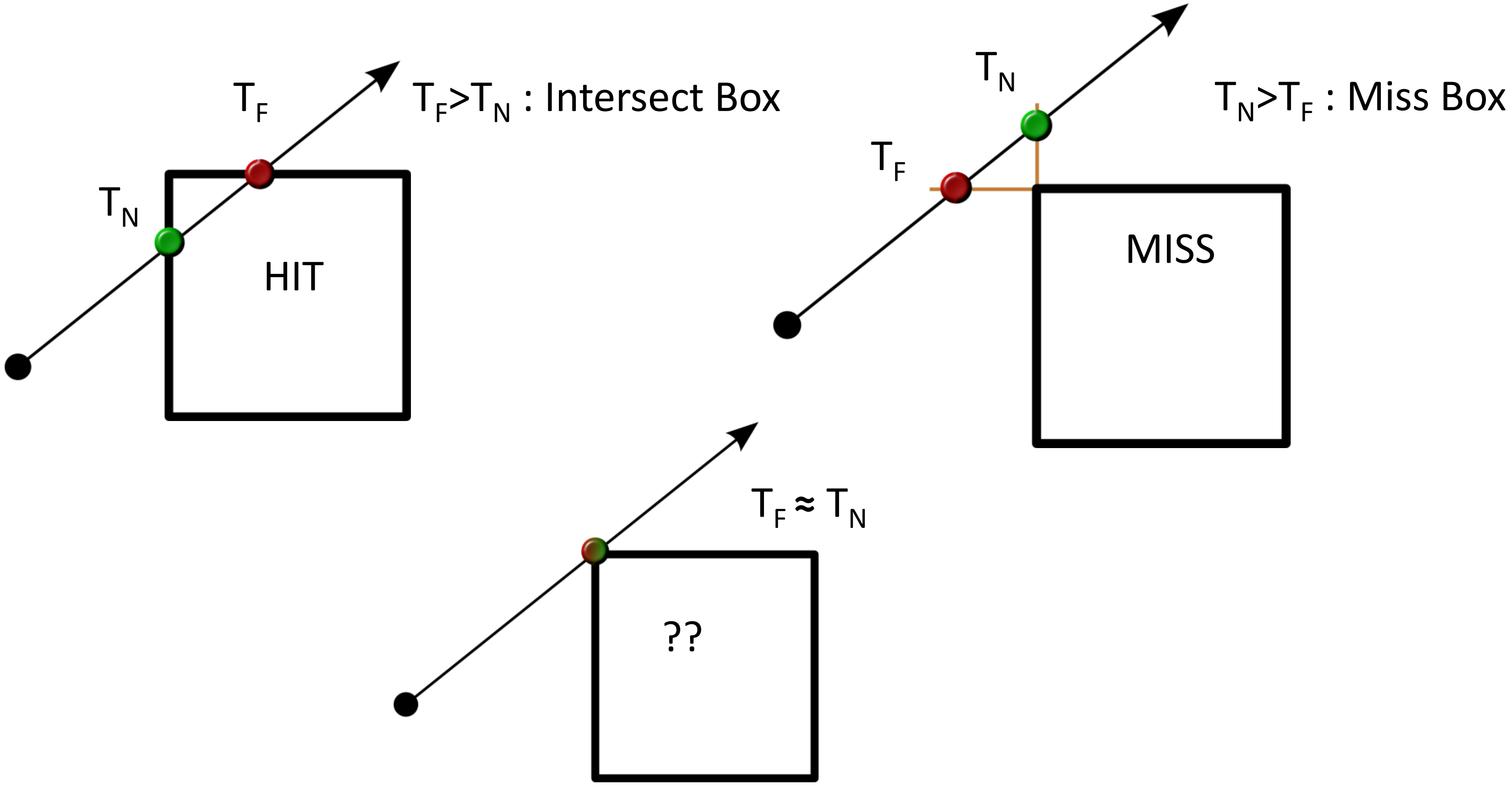


T_F>T_N : Intersect Box



T_N>T_F : Miss Box

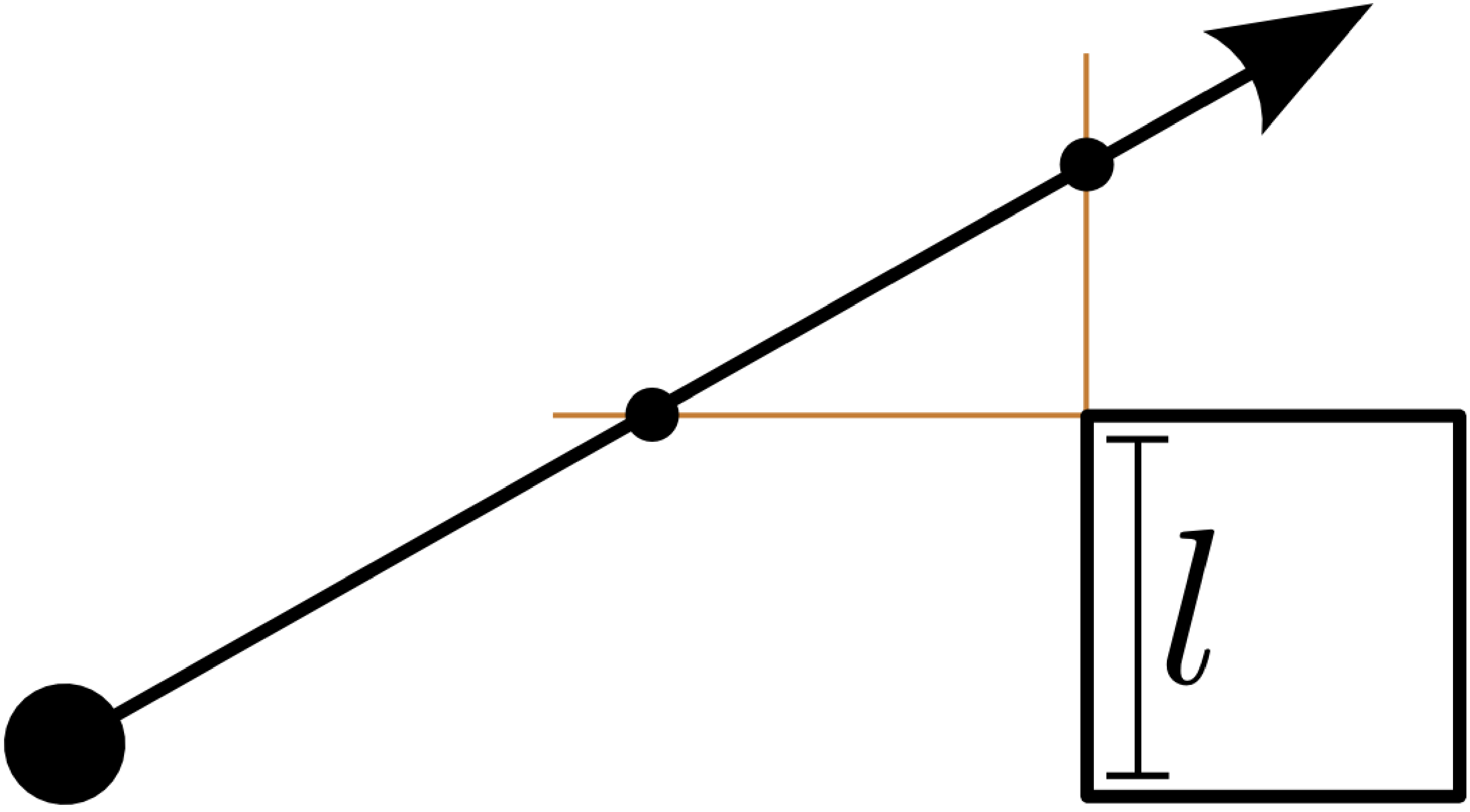
Robust Ray-AABB Test



Ambiguity if $T_F \approx T_N$.

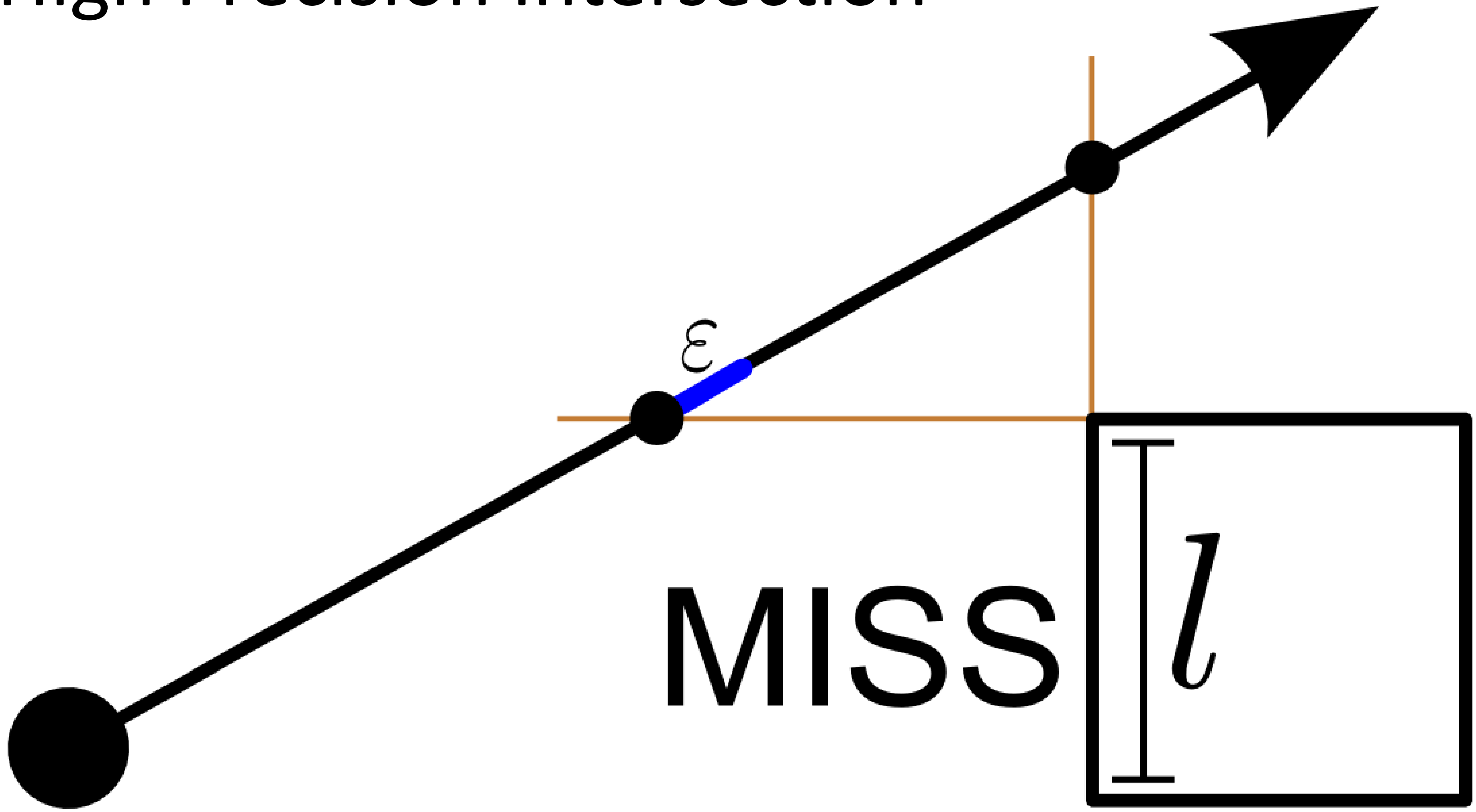
Need $T_N > T_F + \varepsilon$, $\varepsilon = 2*\mathrm{ULP}(T_F)$ to declare a miss [T. Ize, ‘13].

Hit or Miss?



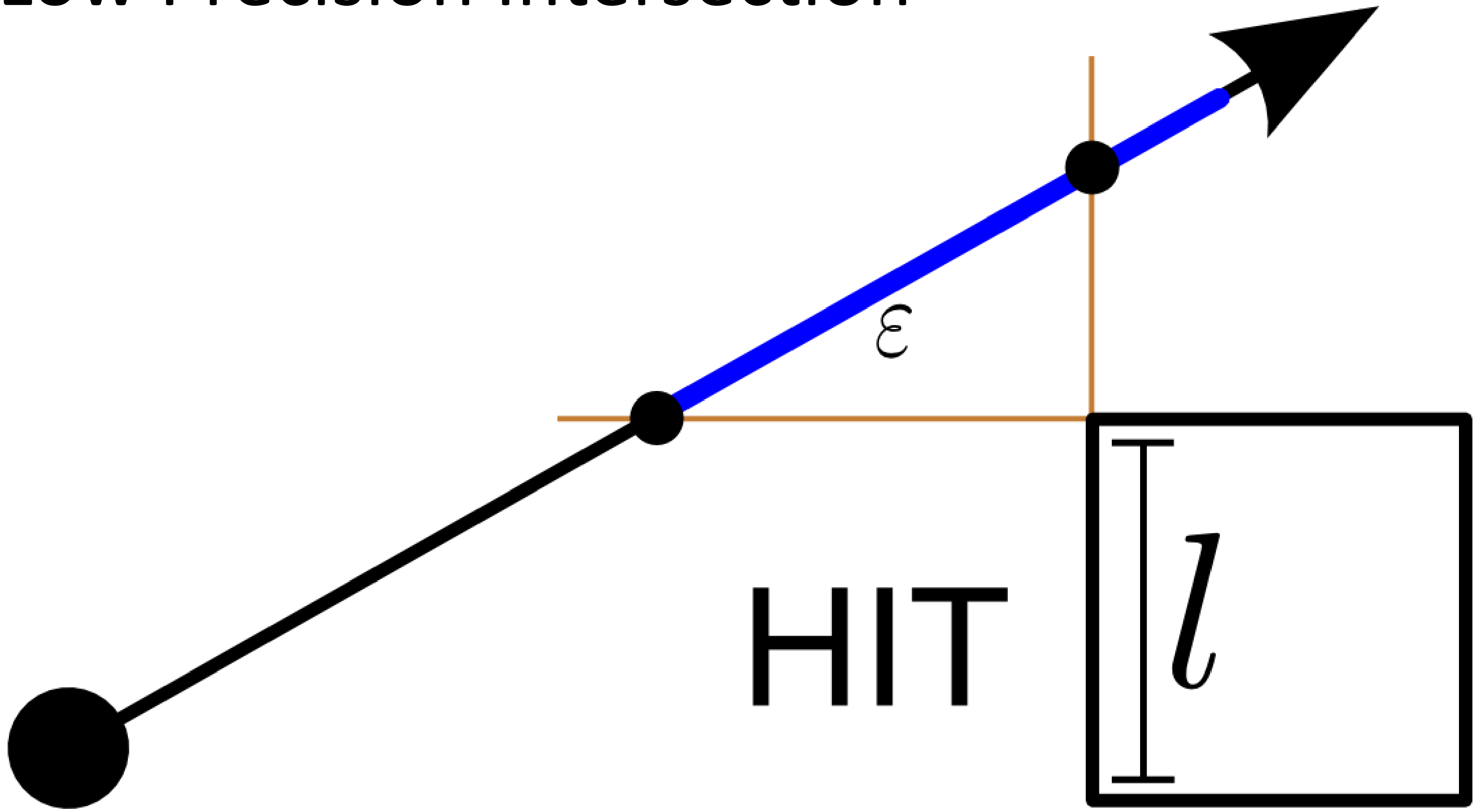
Hit or Miss?

High Precision Intersection

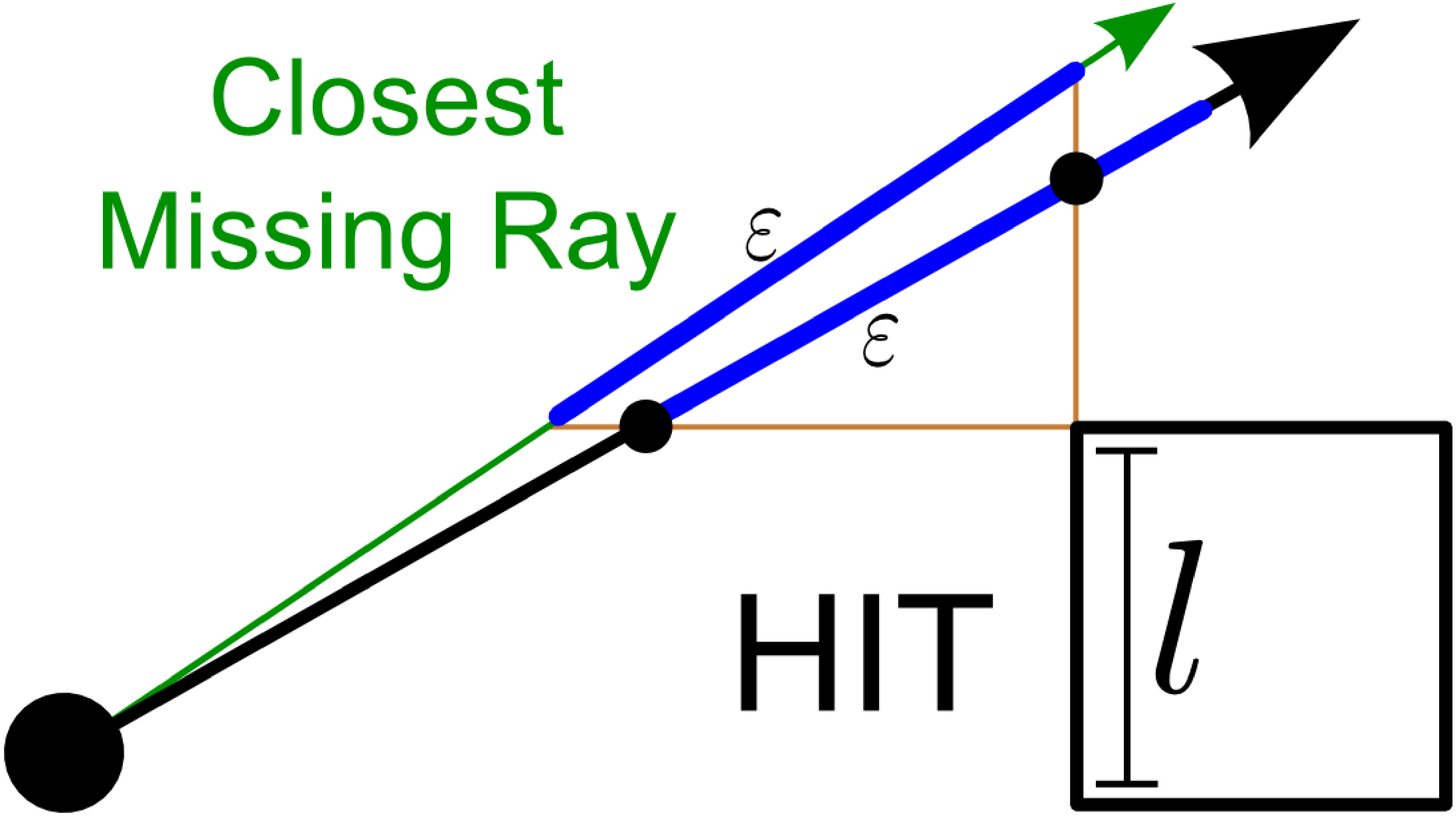


Hit or Miss?

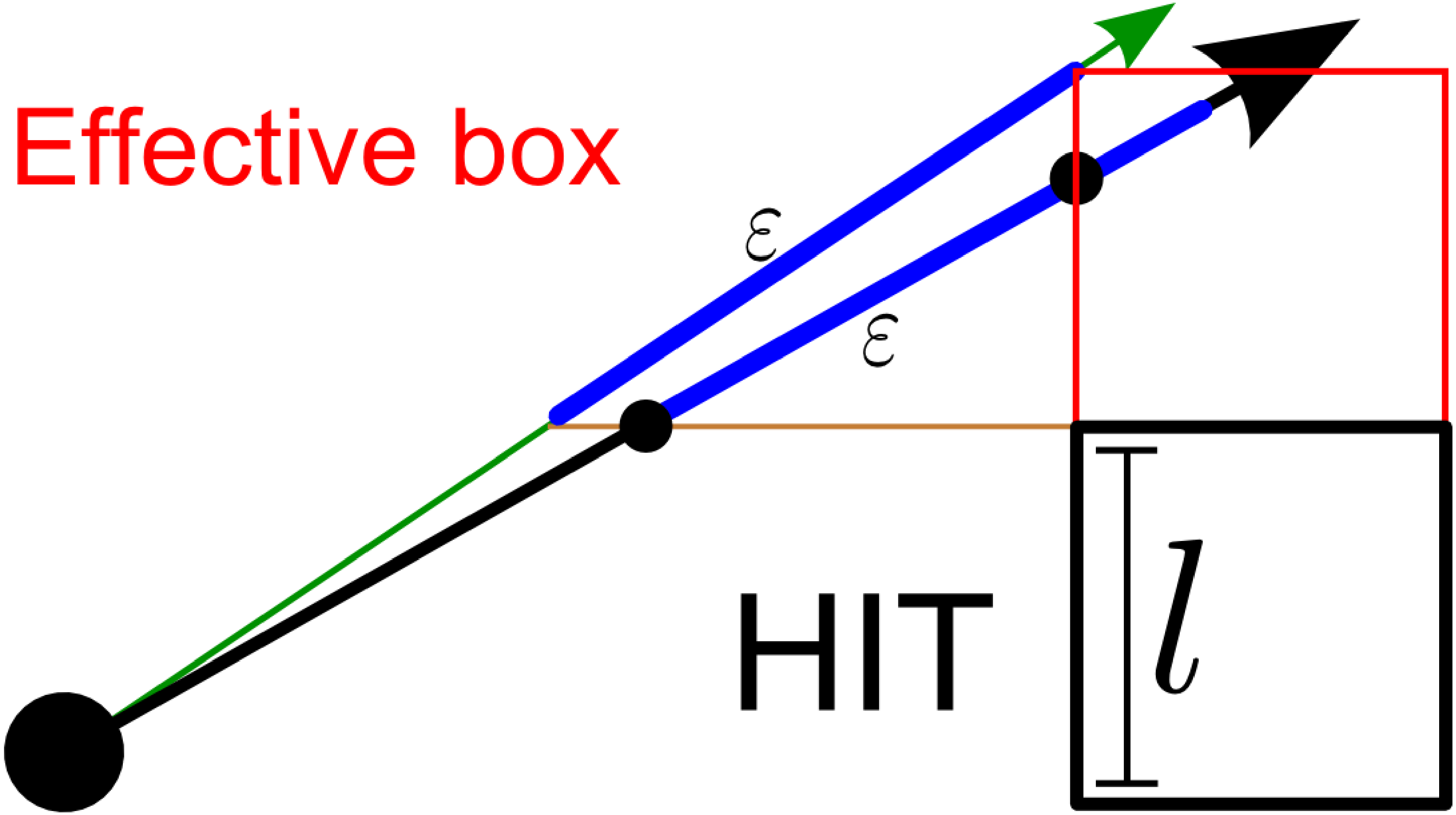
Low Precision Intersection



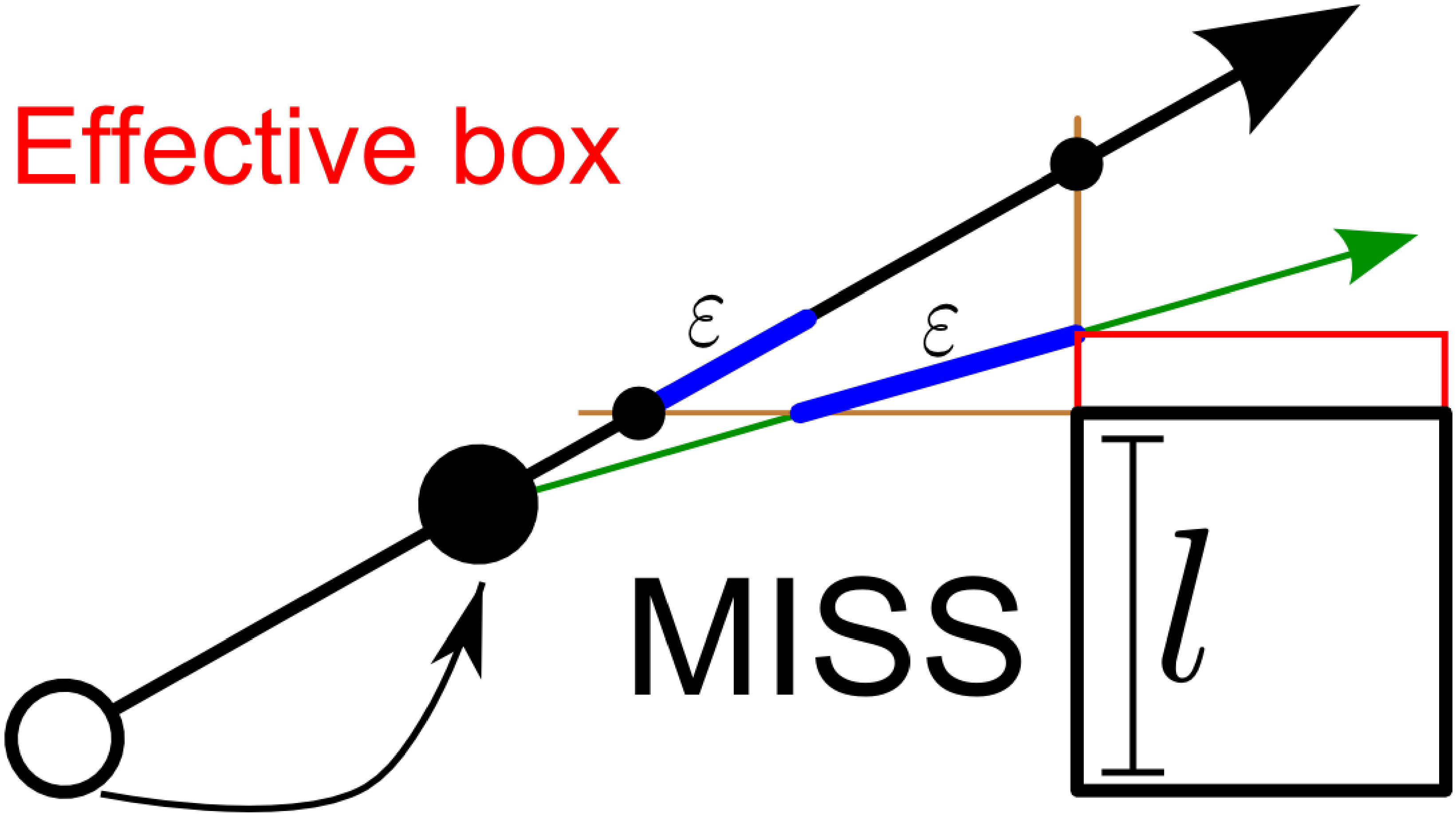
Hit or Miss?



Hit or Miss?



Hit or Miss?



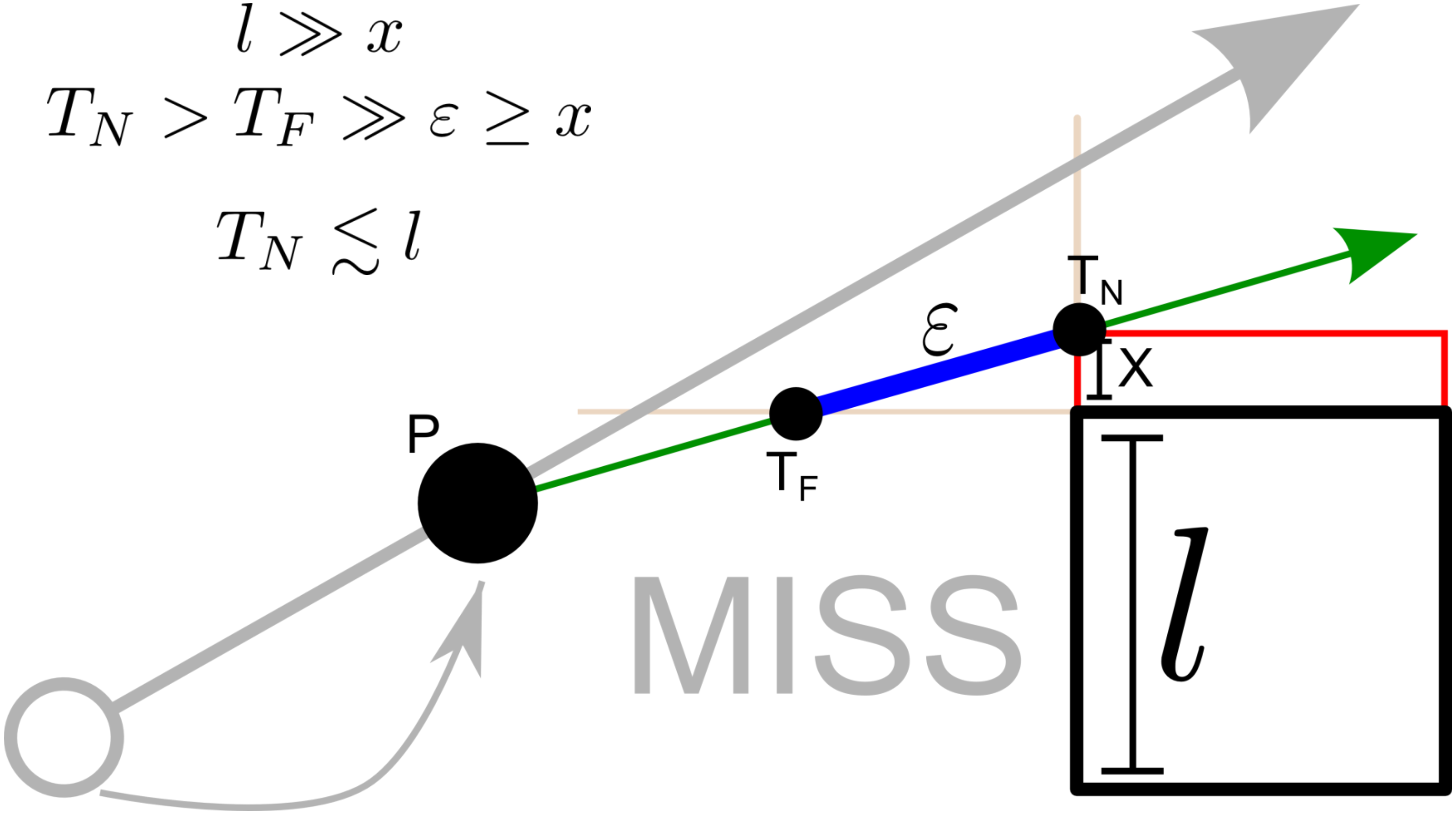
Minimize the *effective* box size

Seek P such that $l + x \approx l$

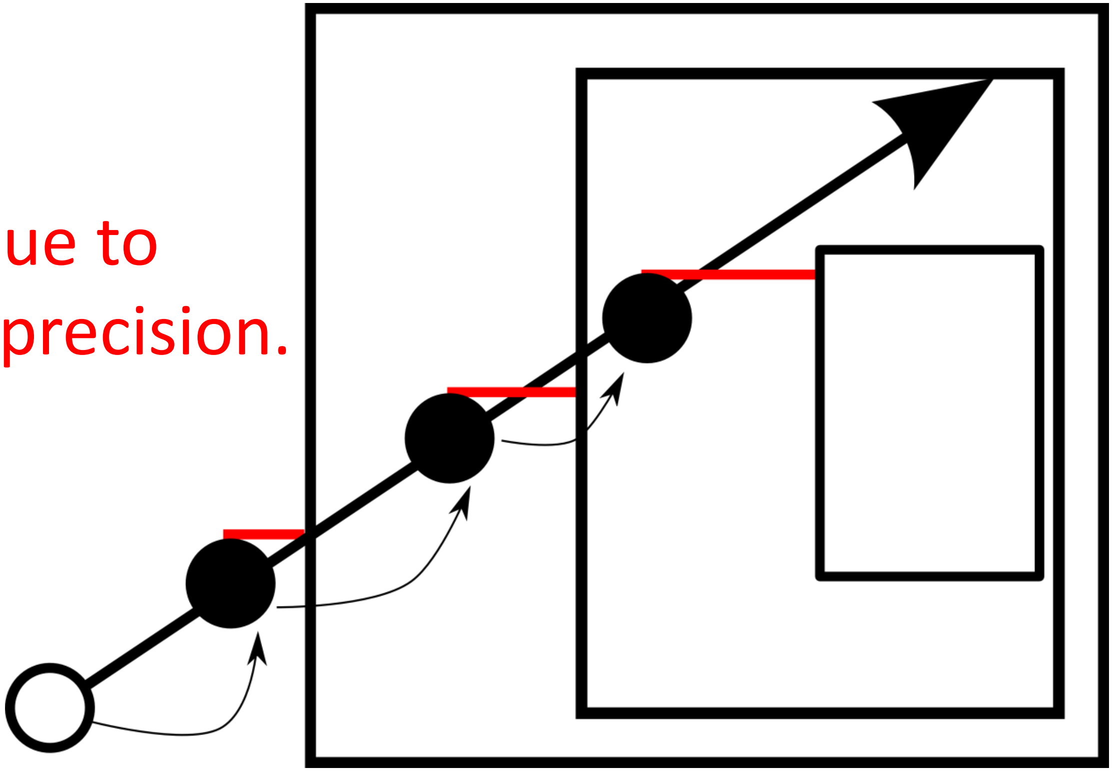
$$l \gg x$$

$$T_N > T_F \gg \varepsilon \geq x$$

$$T_N \lesssim l$$



Traversal Point Update

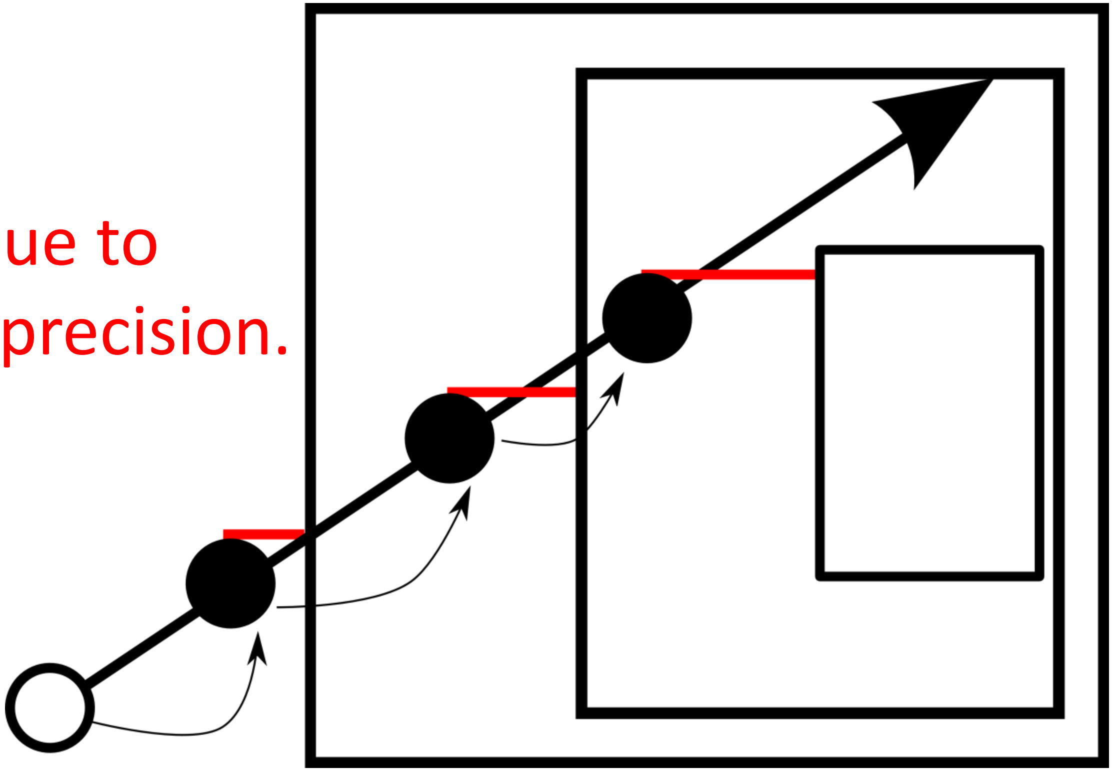


Gap due to finite precision.

Problems

- Computing T_N needs multiplies
- Unbounded box size
- Multiple applications

Traversal Point Update



Gap due to finite precision.

$$f(C+S)=1$$

f Precision of update procedure

S Maximum parent-child edge ratio

C Maximum internal parent-child offset

Traversal Point Update

- 5 bit child boxes -> S=32
  - Need 8 bit arithmetic to guarantee $x \ll l$

- Can actually use just **1 bit**!
  - Incorrectly taken paths are quickly aborted
  - Much smaller than even 8 bit arithmetic

- Reduces precision of the adds and divides as well
  - No need to share divider, replaced with LUT.

Reduced Precision Traversal Unit

- Fixed function traversal unit implements:
  1. **Two 1 bit precision traversal point updates**
  2. Two 5 bit precision robust Ray-AABB tests
  3. Near child detection
  4. Single bit traversal stack
  5. …

Reduced Precision Traversal Unit

- Fixed function traversal unit implements:
  1. Two 1 bit precision traversal point updates
  2. **Two 5 bit precision robust Ray-AABB tests**
  3. **Near child detection**
  4. **Single bit traversal stack**
  5. **…**

Reduced Precision Traversal Unit

- Fixed function traversal unit implements:
  1. Two 1 bit precision traversal point updates
  2. Two 5 bit precision robust Ray-AABB tests
  3. Near child detection
  4. Single bit traversal stack
  5. …

- A fully pipelined traversal unit is roughly 6% the area of a single SIMD FPU in a current GPU.

BVH Compression

- Traversal inputs drop to 5 bits early on
    - Can’t distinguish all boxes

- Opportunity to save bandwidth!

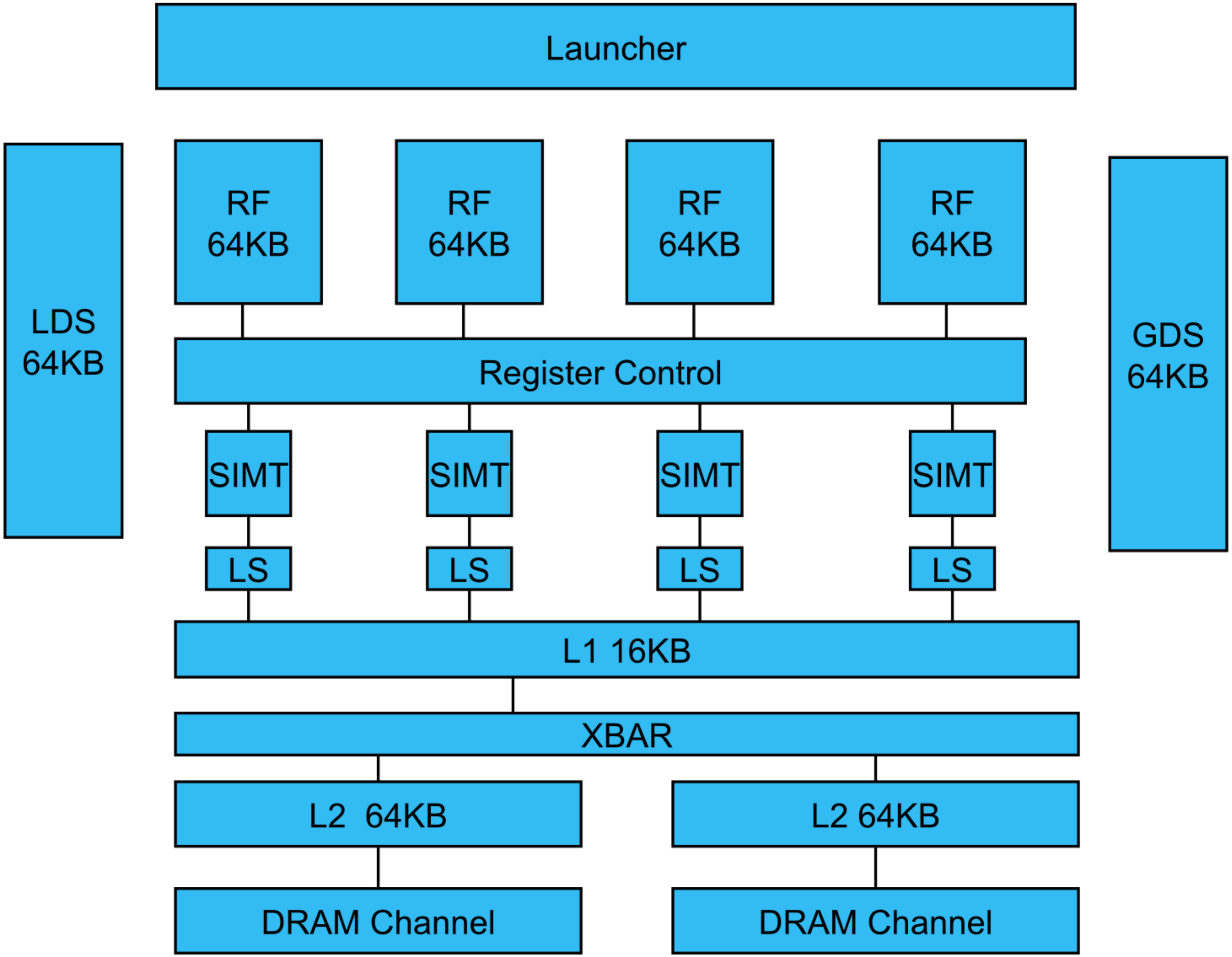
- BVH boxes are quantized to 5 bits
    - 12 byte node format holding two boxes

GPU Integration & Scheduling

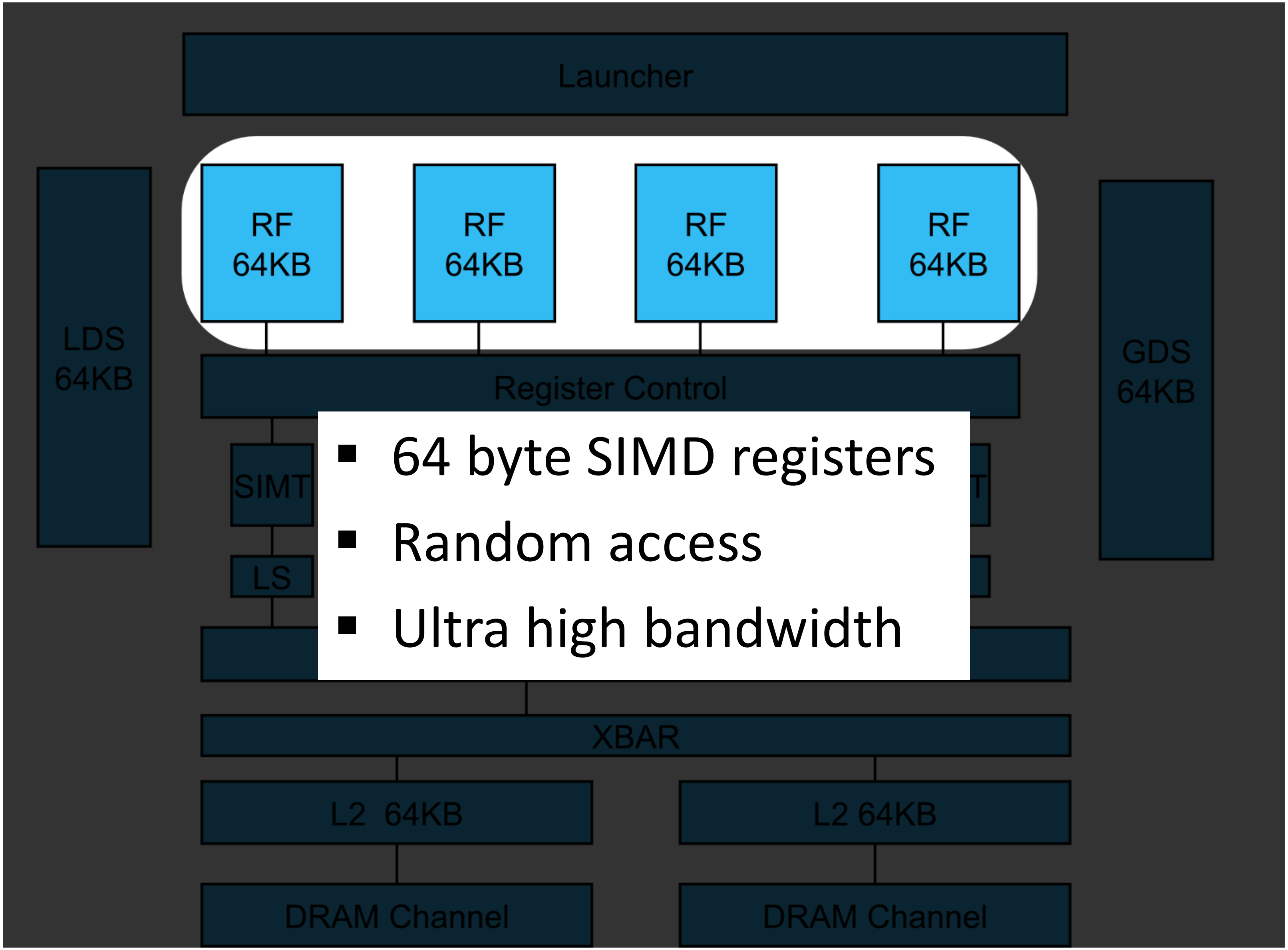
Schedule to minimize off chip traffic

- Treelet scheduling [T. Aila,... ‘10]
  - Smaller is better…but for queuing traffic.
- Stay out of memory!
  - On chip queuing.
- Lots of rays
  - Improve odds of reuse.

We can use everything here

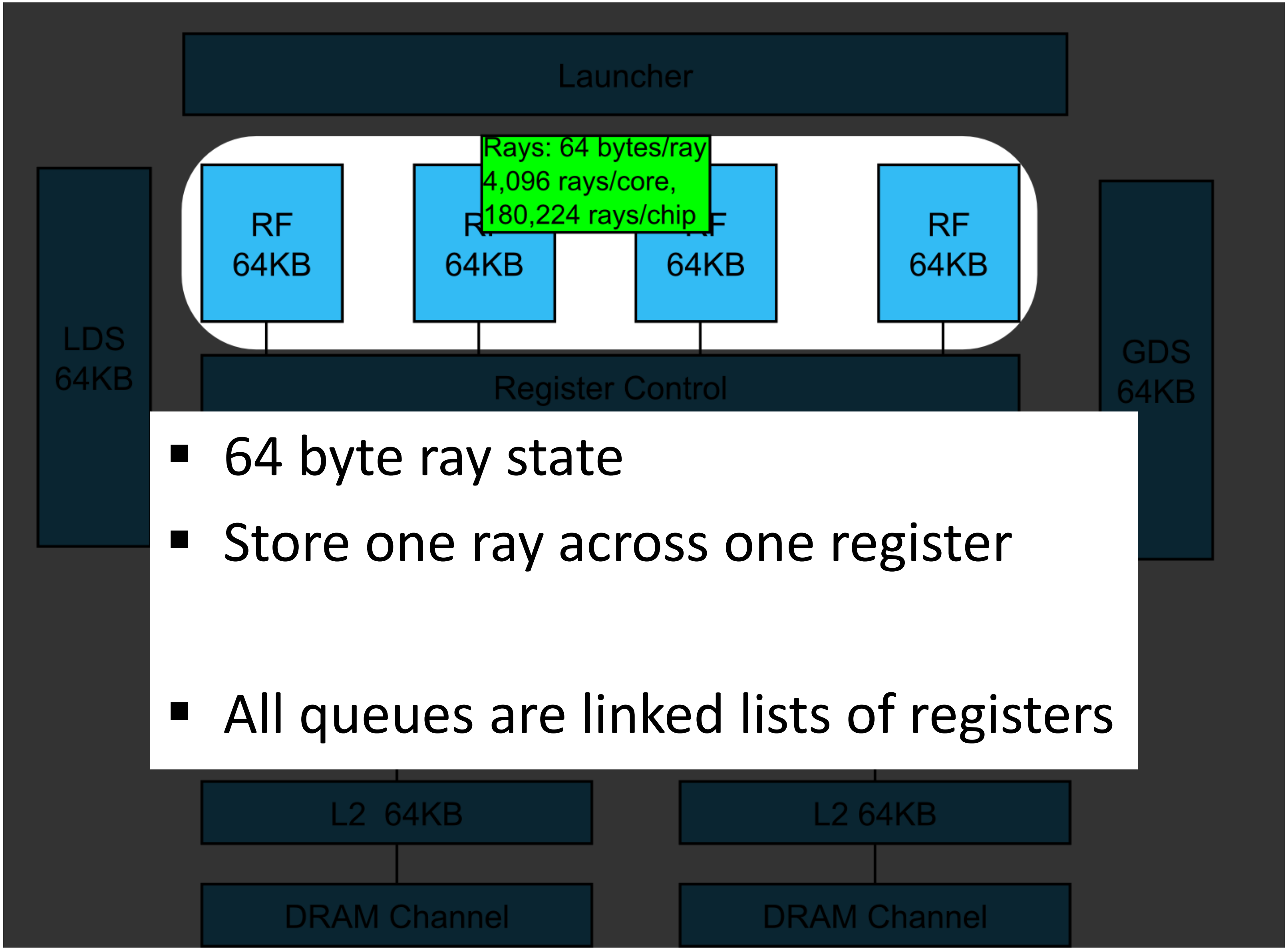


Registers are great for rays



- 64 byte SIMD registers
- Random access
- Ultra high bandwidth

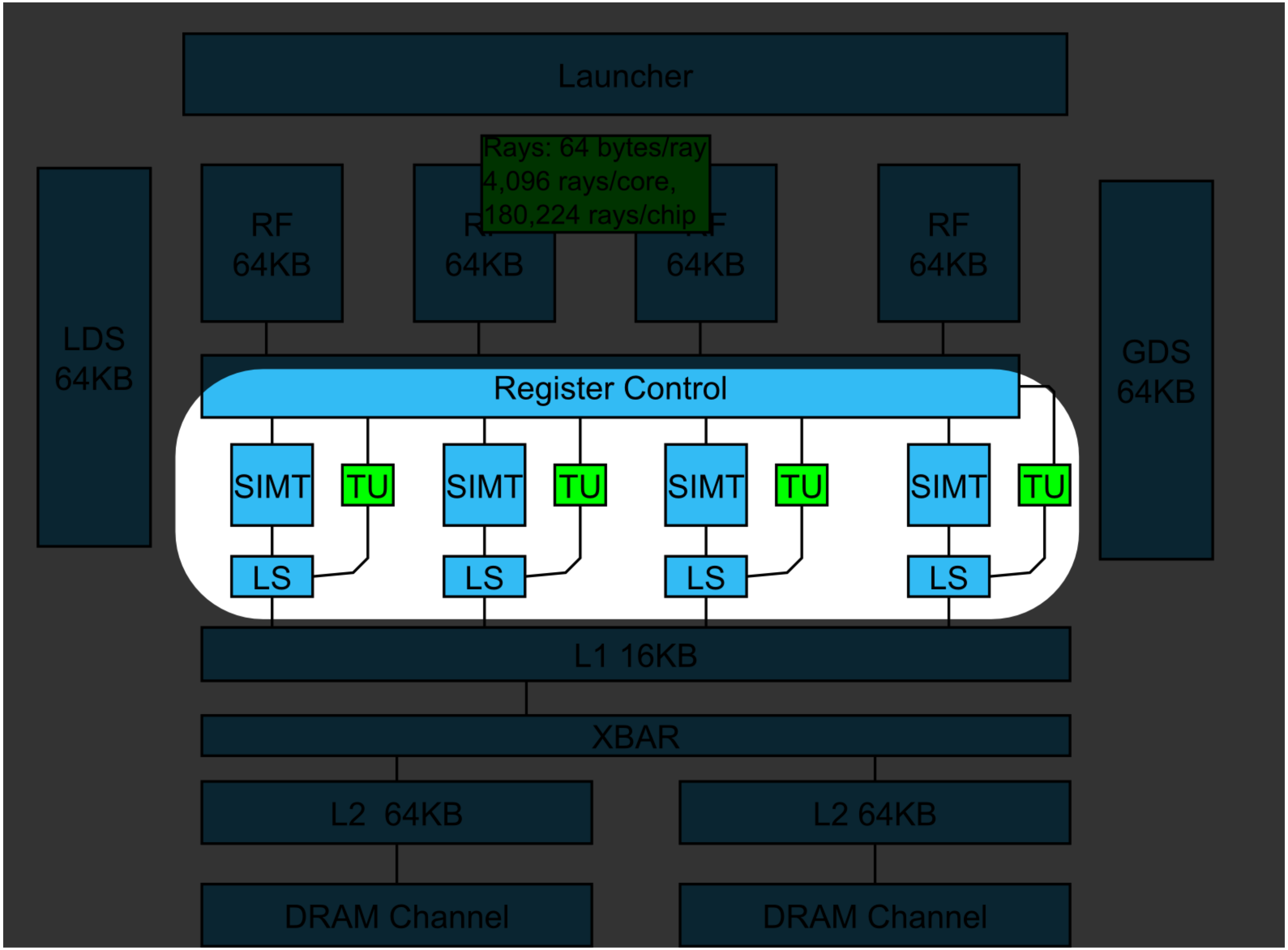
Registers are great for rays



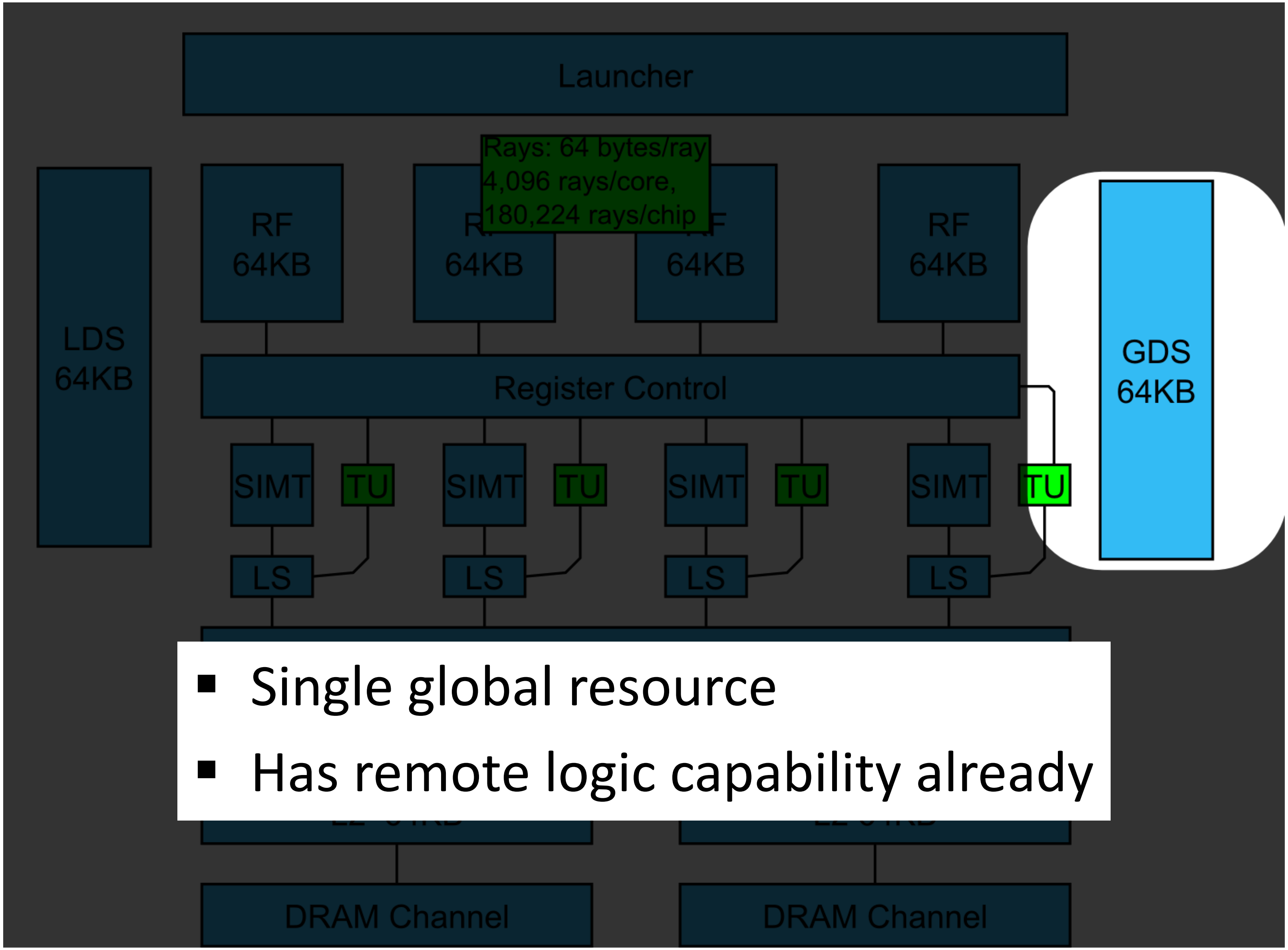
Rays: 64 bytes/ray
4,096 rays/core,
180,224 rays/chip

- 64 byte ray state
- Store one ray across one register
- All queues are linked lists of registers

MIMD traversal shares with SIMD core.

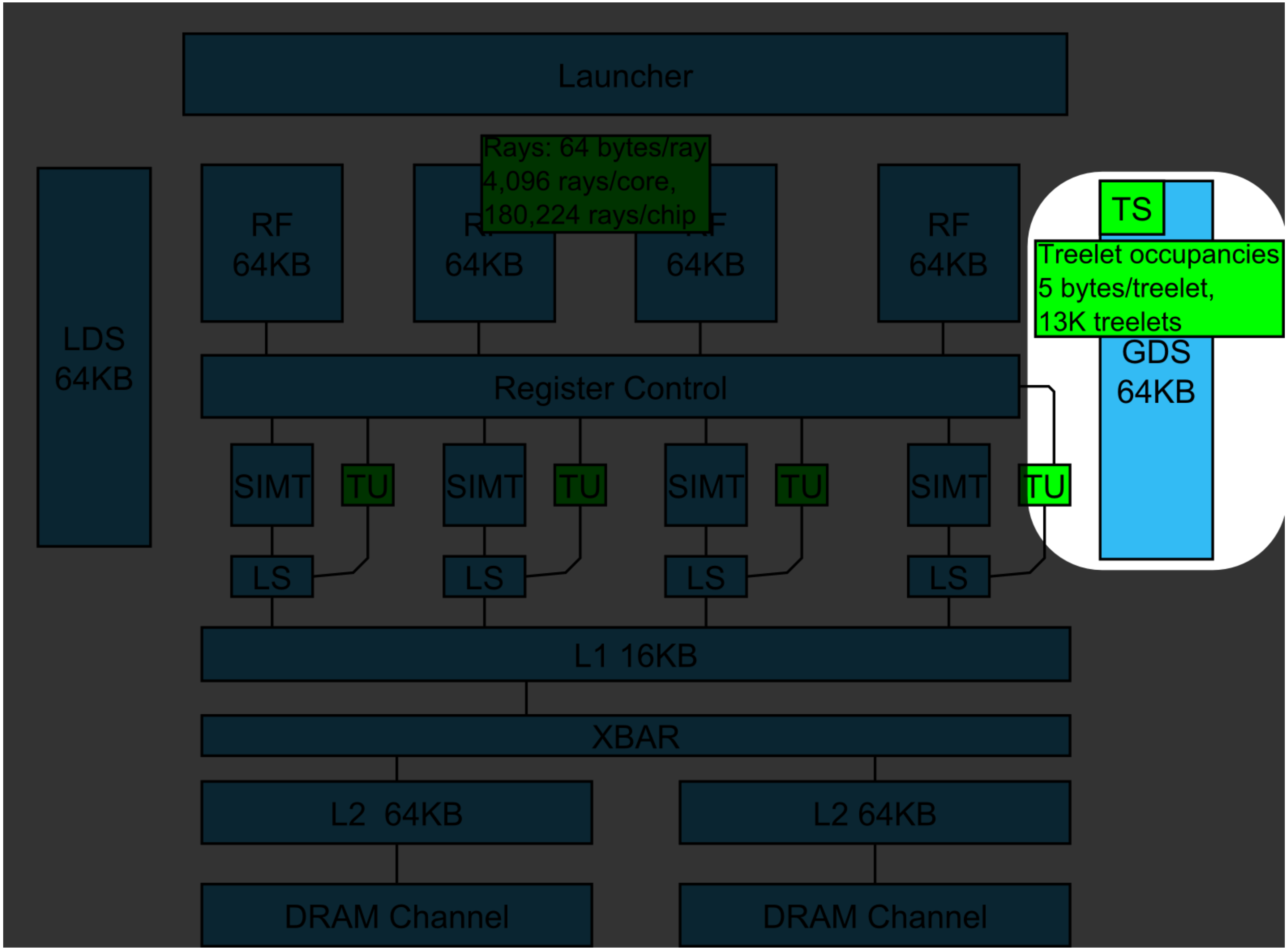


Queue data stays on chip



- Single global resource
- Has remote logic capability already

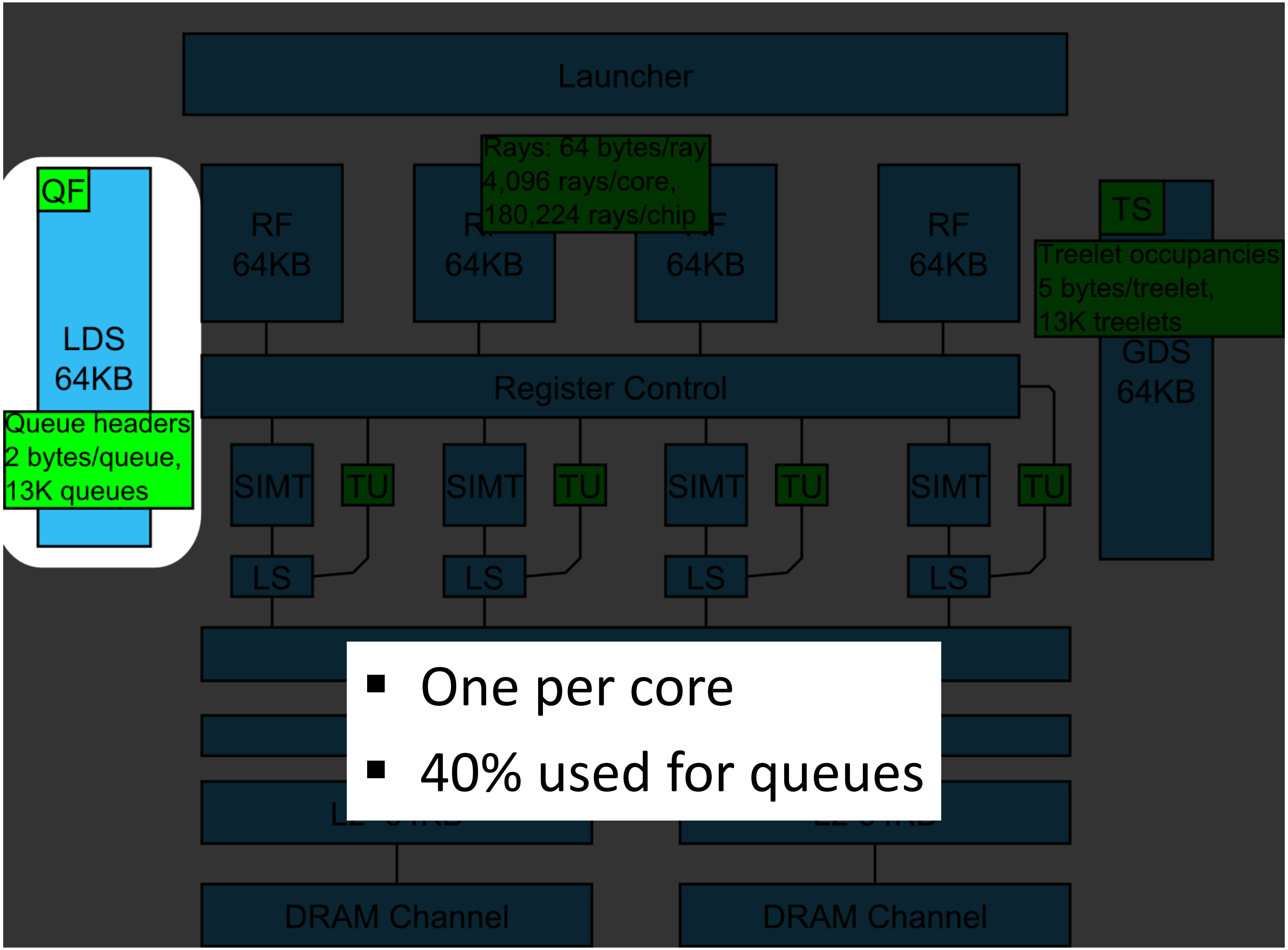
Queue data stays on chip



Rays: 64 bytes/ray
4,096 rays/core,
180,224 rays/chip

Treelet occupancies
5 bytes/treelet,
13K treelets

Queue data stays on chip



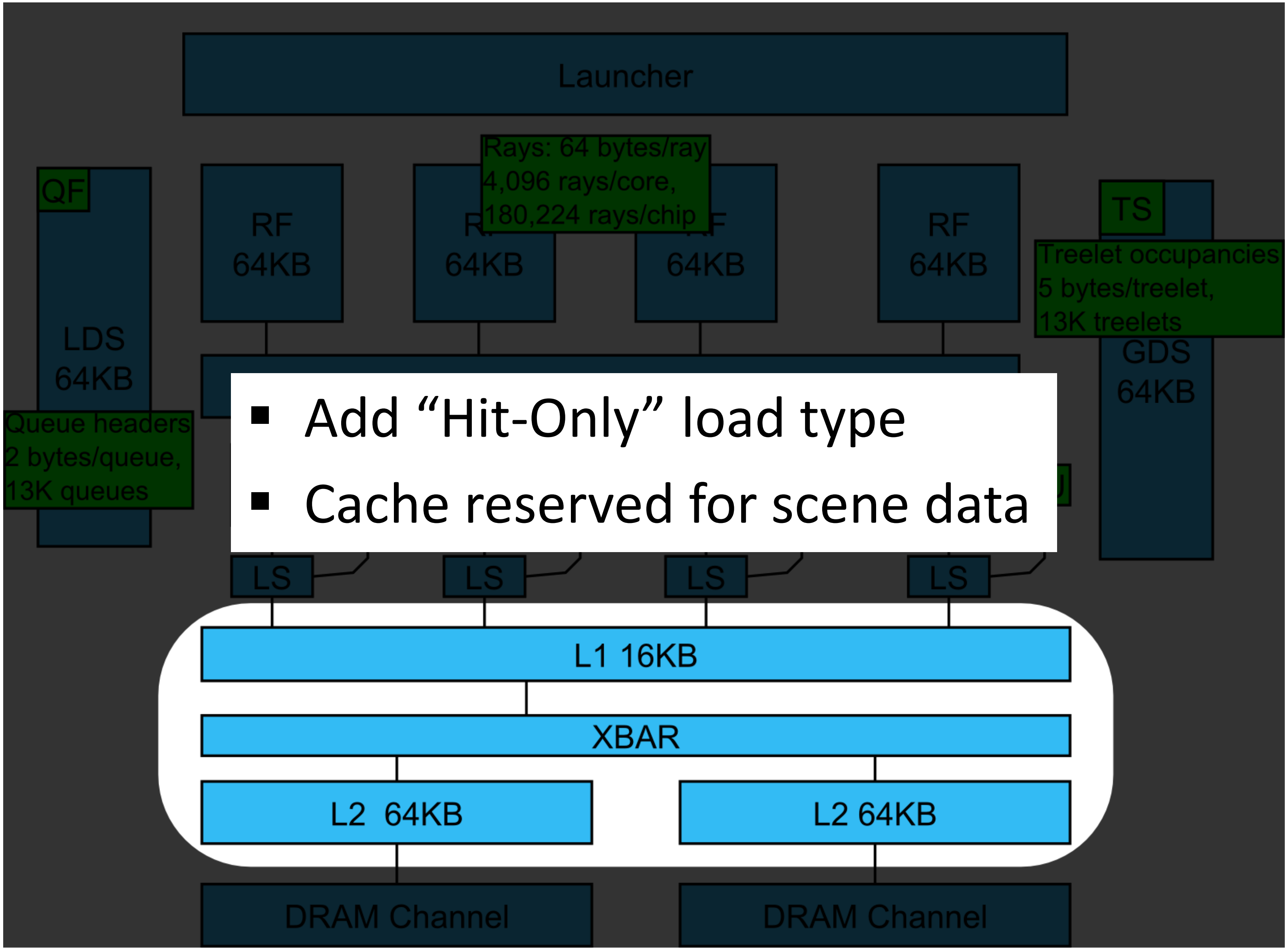
Rays: 64 bytes/ray
4,096 rays/core,
180,224 rays/chip

Queue headers
2 bytes/queue,
13K queues

Treelet occupancies
5 bytes/treelet,
13K treelets

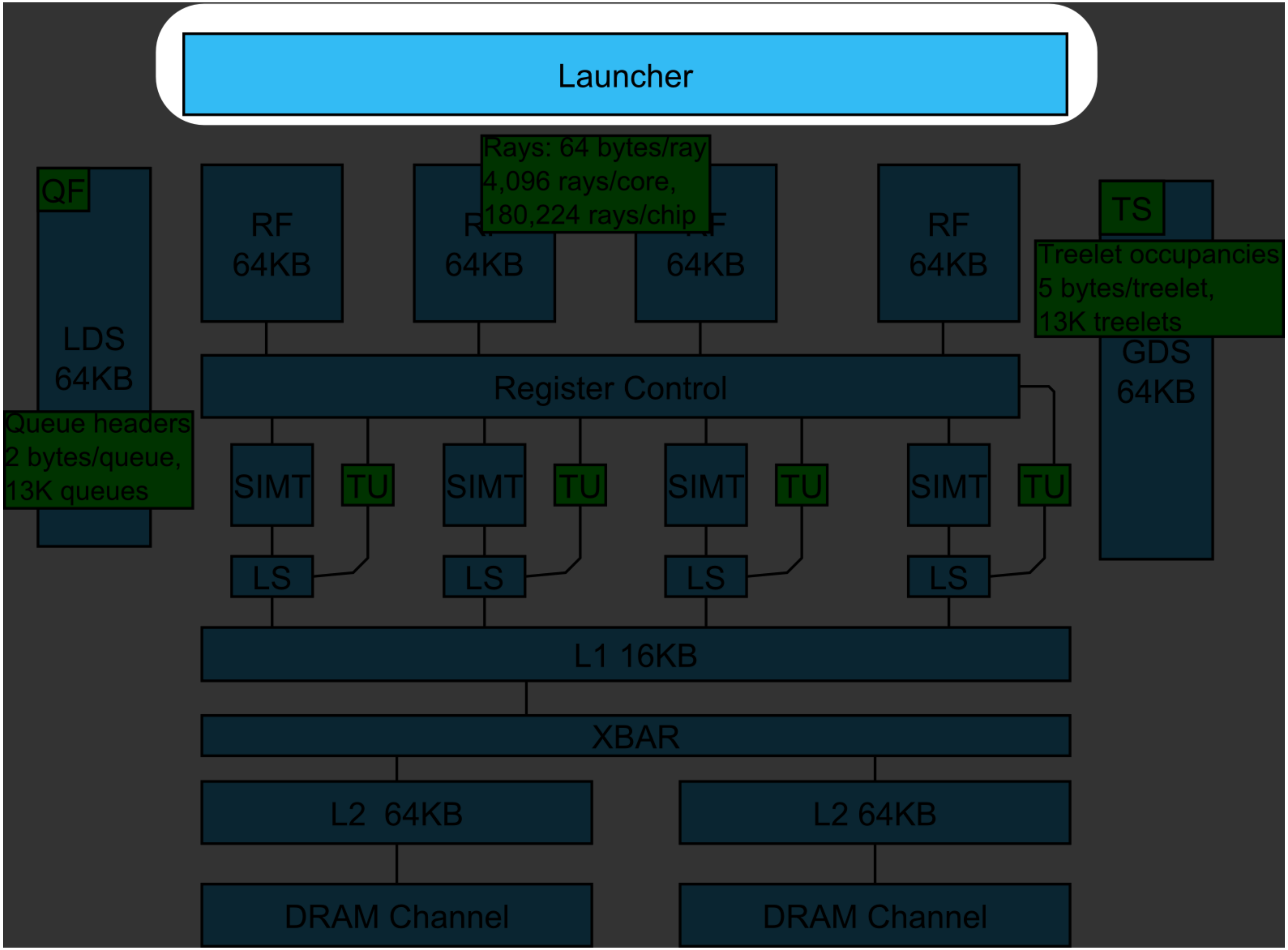
- One per core
- 40% used for queues

Queue rays when data is expensive



- Add “Hit-Only” load type
- Cache reserved for scene data

Queue based intersect and shading

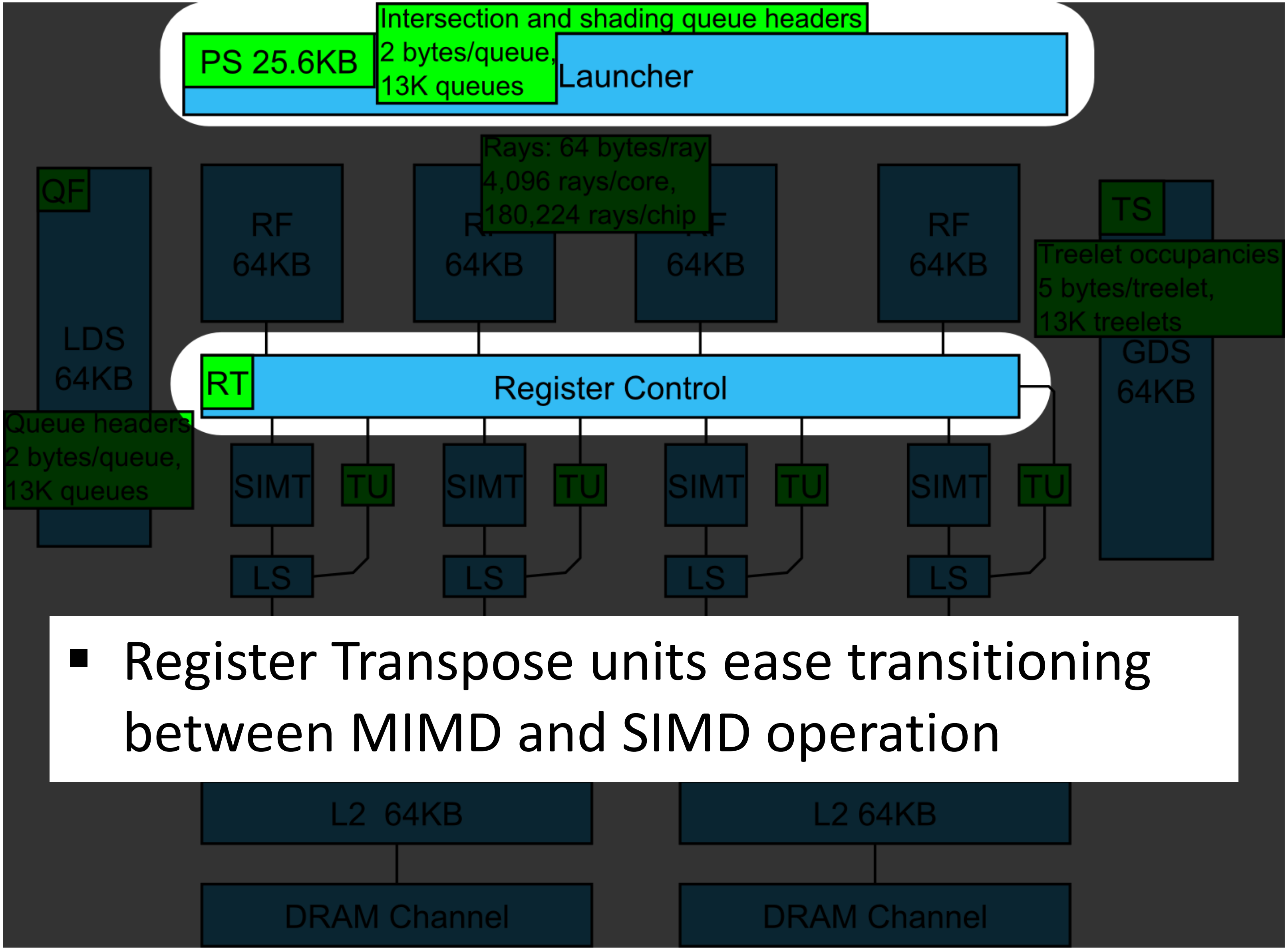


Rays: 64 bytes/ray
4,096 rays/core,
180,224 rays/chip

Treelet occupancies
5 bytes/treelet,
13K treelets

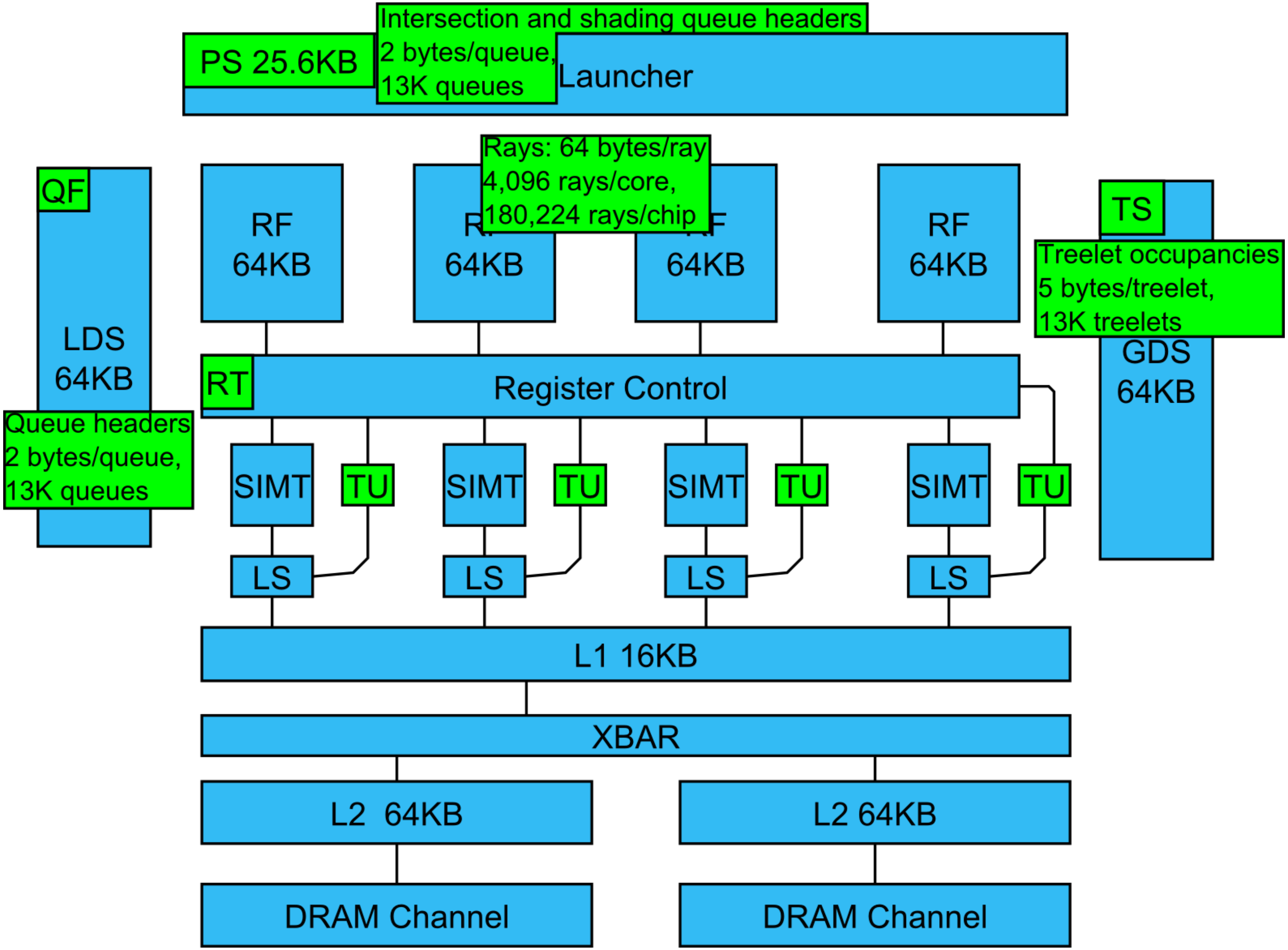
Queue headers
2 bytes/queue,
13K queues

Queue based intersect and shading



- Register Transpose units ease transitioning between MIMD and SIMD operation

Ray tracing fits in current GPUs

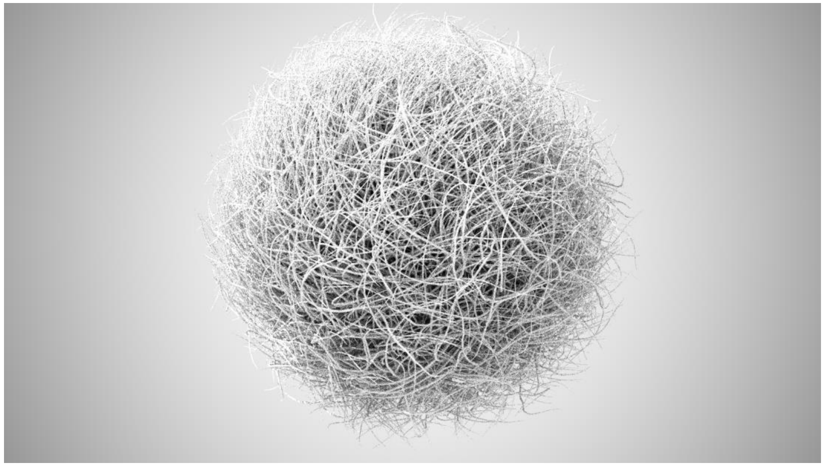


Evaluation

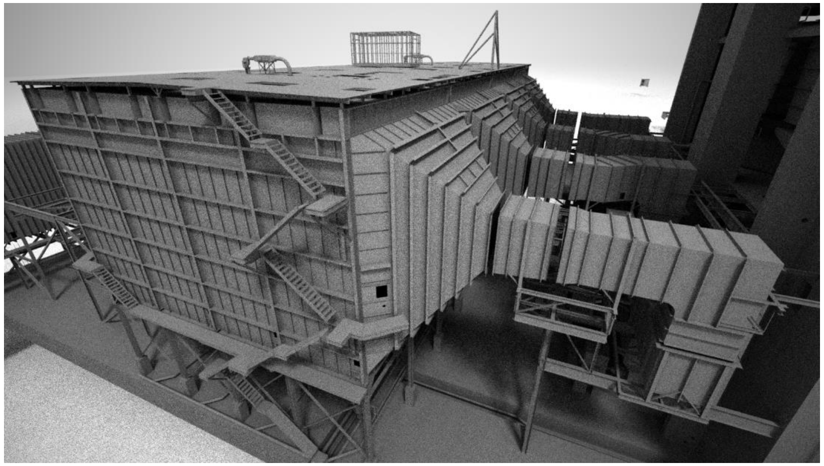
Experimental setup

Two simulators measure key metrics

Crown: 4.9M Triangles



Hairball: 2.9M Triangles

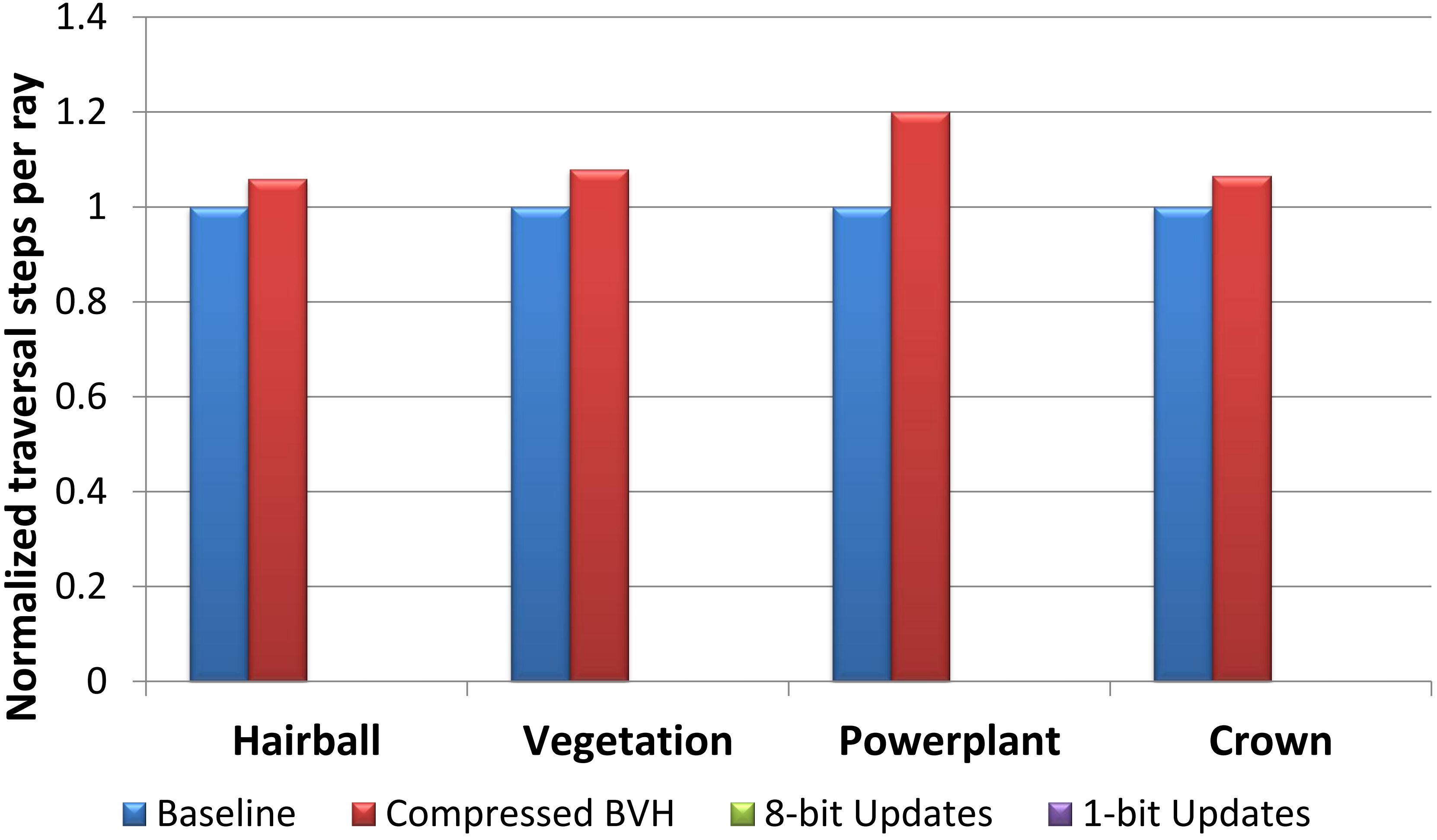


Powerplant: 12.8M Triangles

Vegetation: 1.1M Triangles

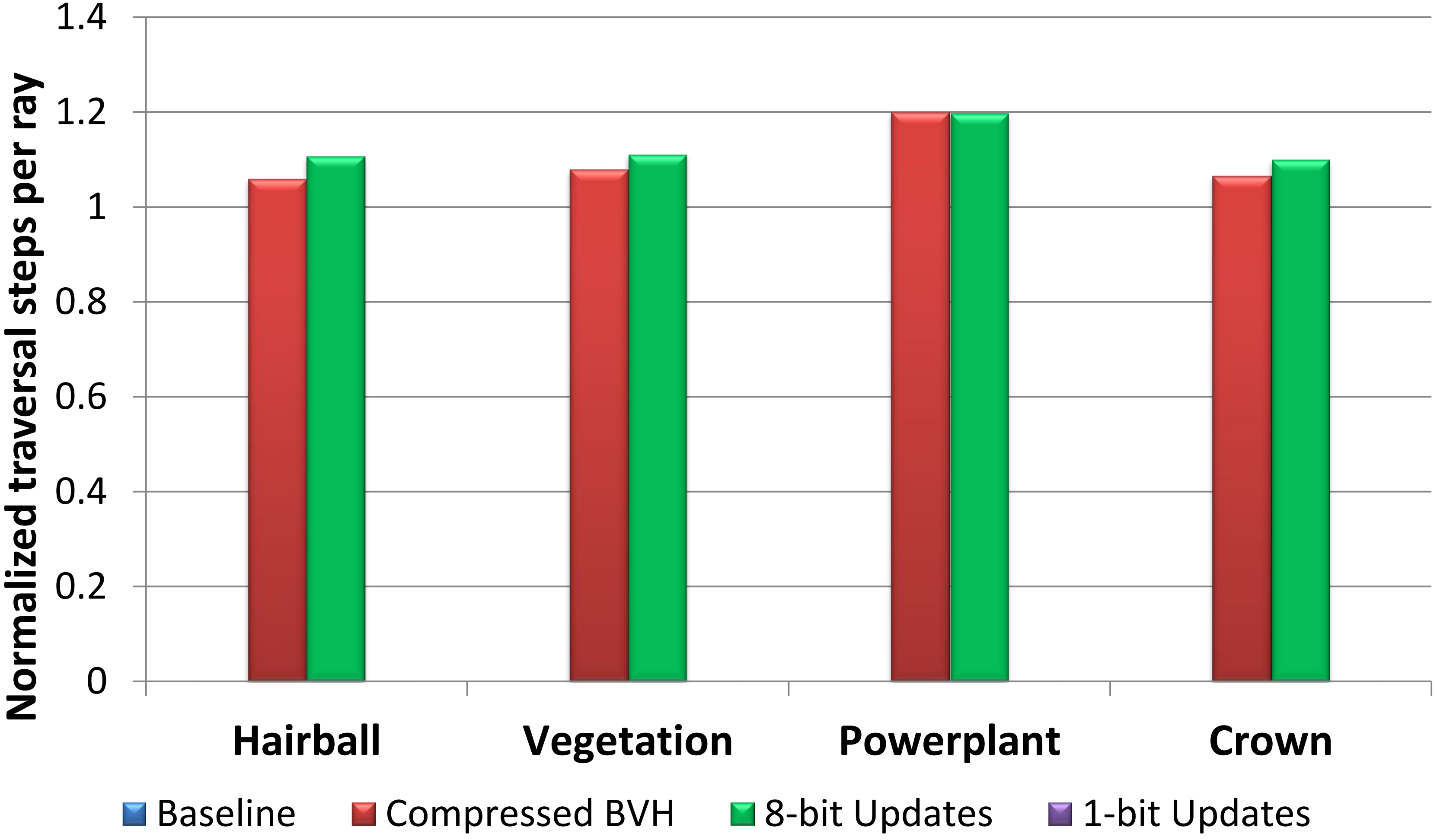
Reduced precision has low costs

Compressed BVH: 7% more work (usually)



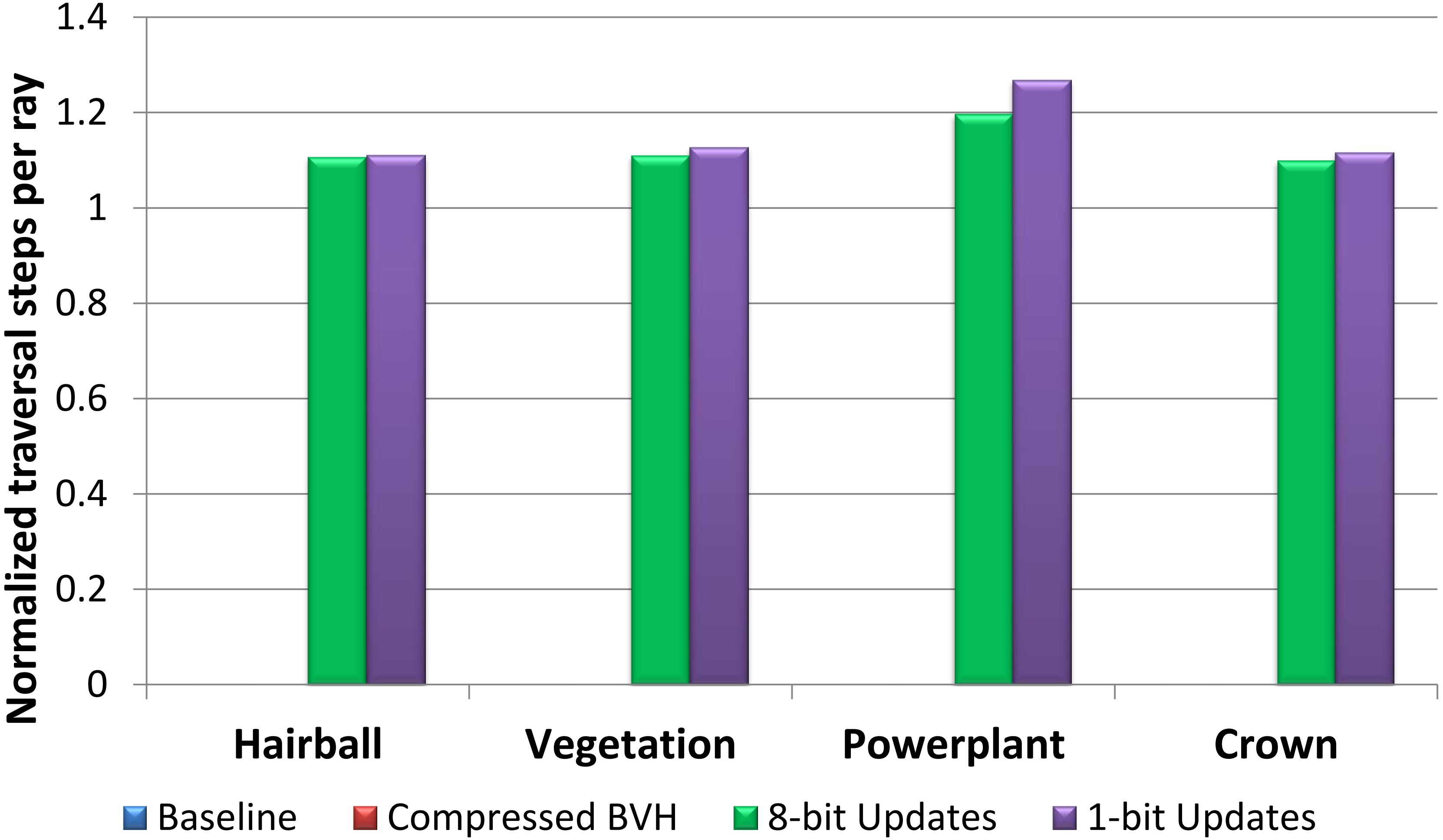
Reduced precision has low costs

Reduced Precision Traversal: 3% more work



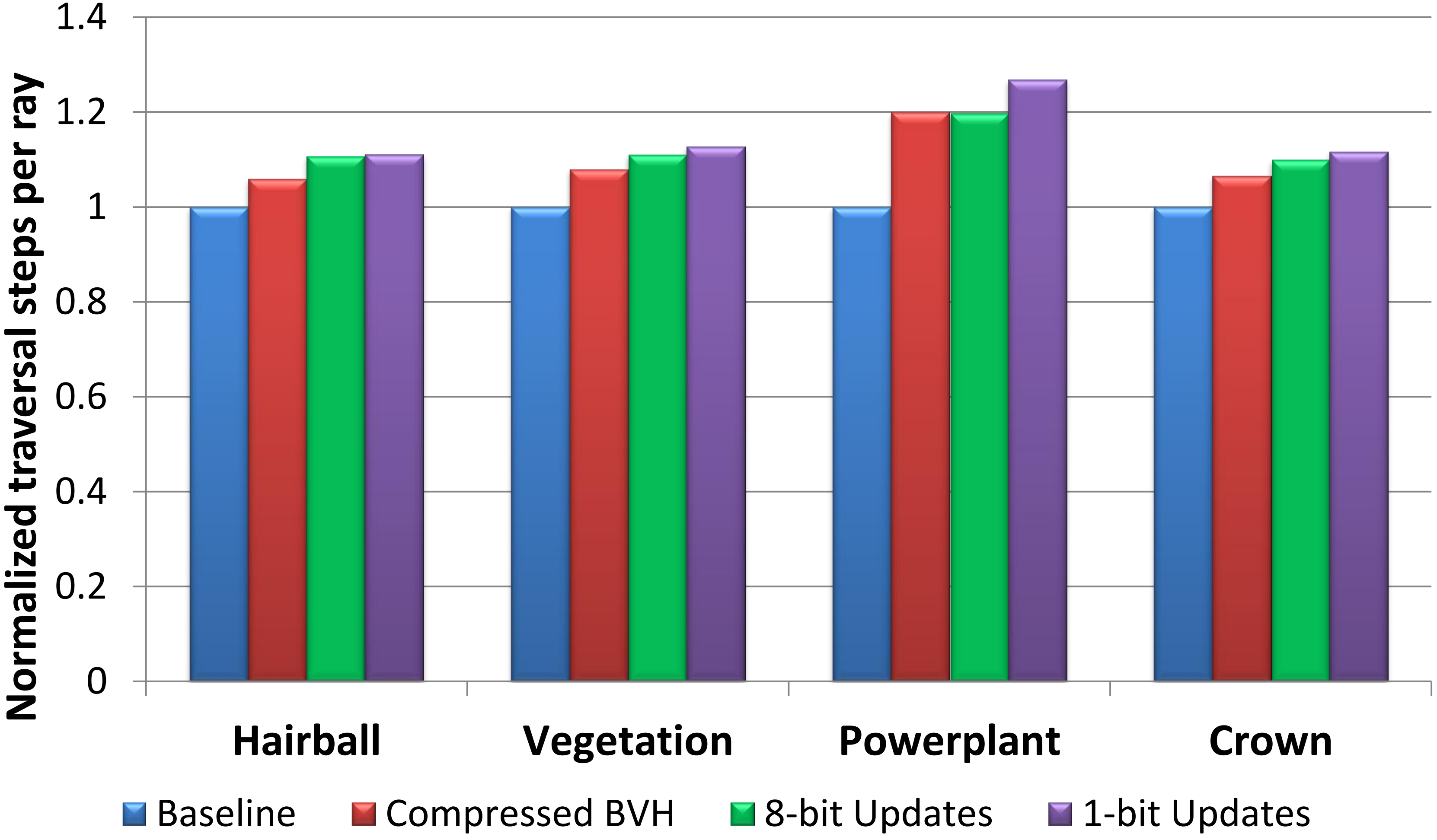
Reduced precision has low costs

1-bit Traversal Point Update: 1-2% more work



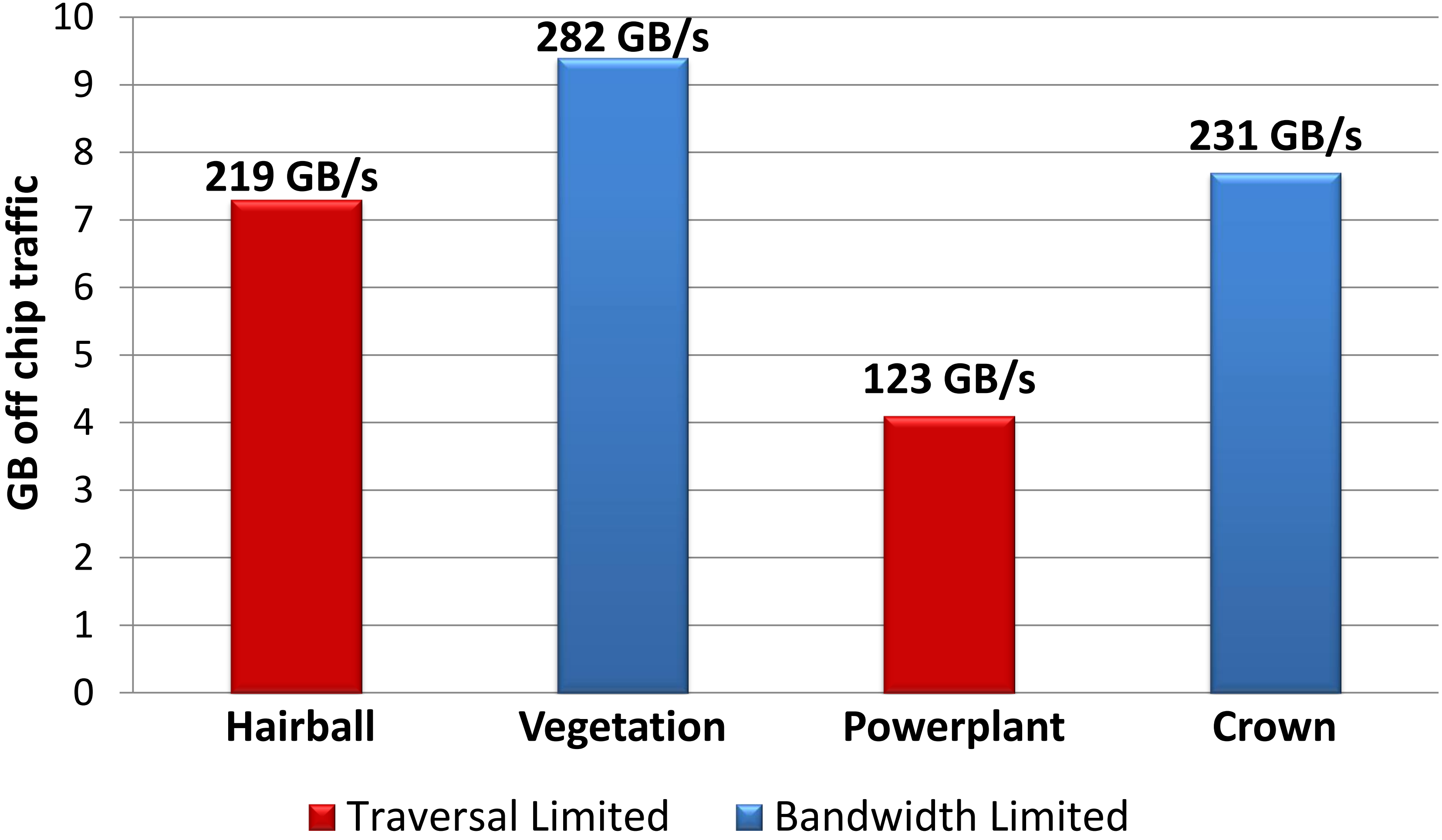
Total overhead is small

10-15% (usually)



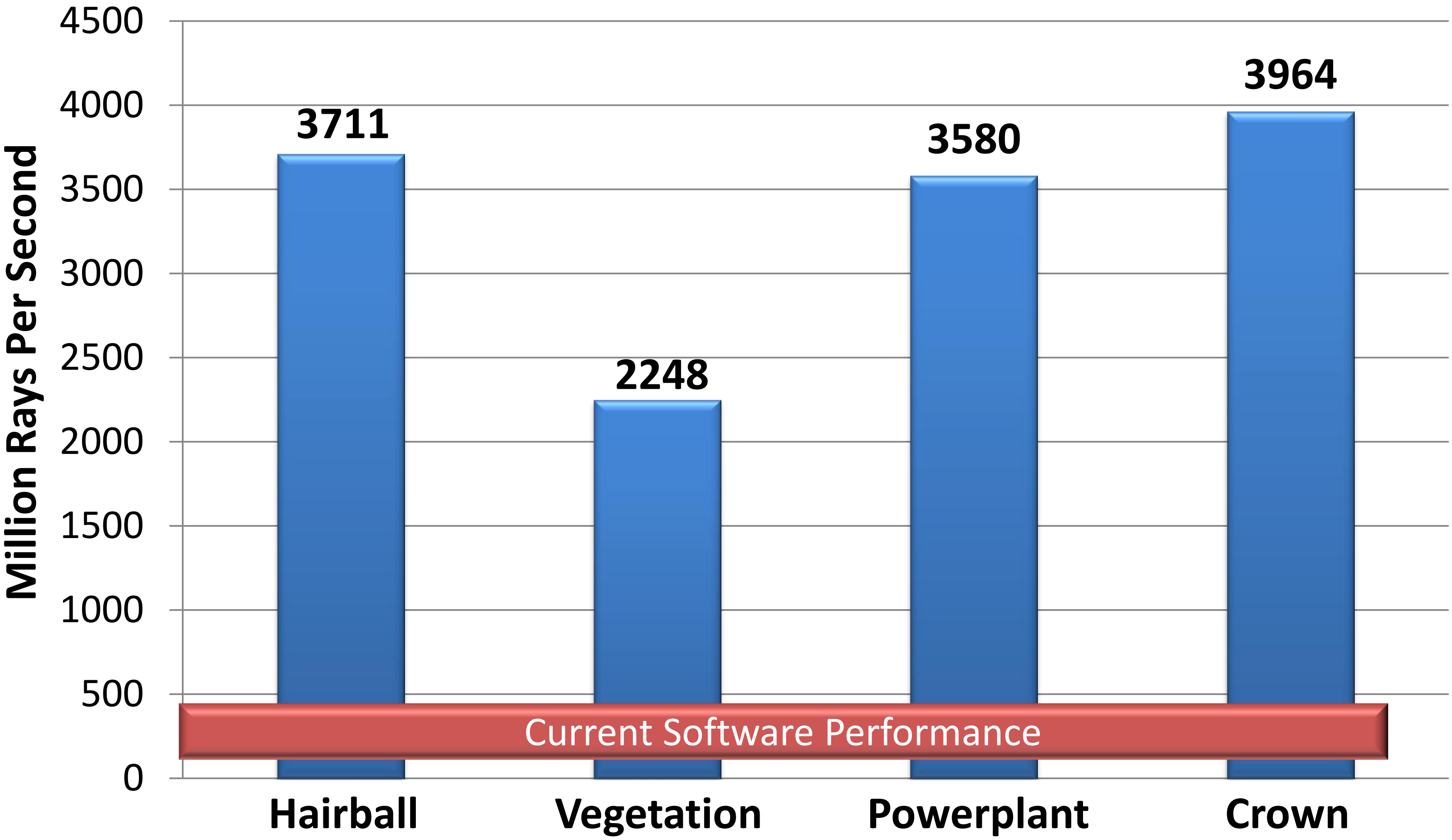
Off chip traffic is in the real time range

Simulated workload set to 30 frames per second



Simulated ray rate

Average ray rate: 3.4 Billion rays/s



Conclusion

- Reduced precision yields surprising performance benefits…roughly 20x.

- Hardware ray tracing acceleration can be a lightweight feature of modern GPUs.

Acknowledgements

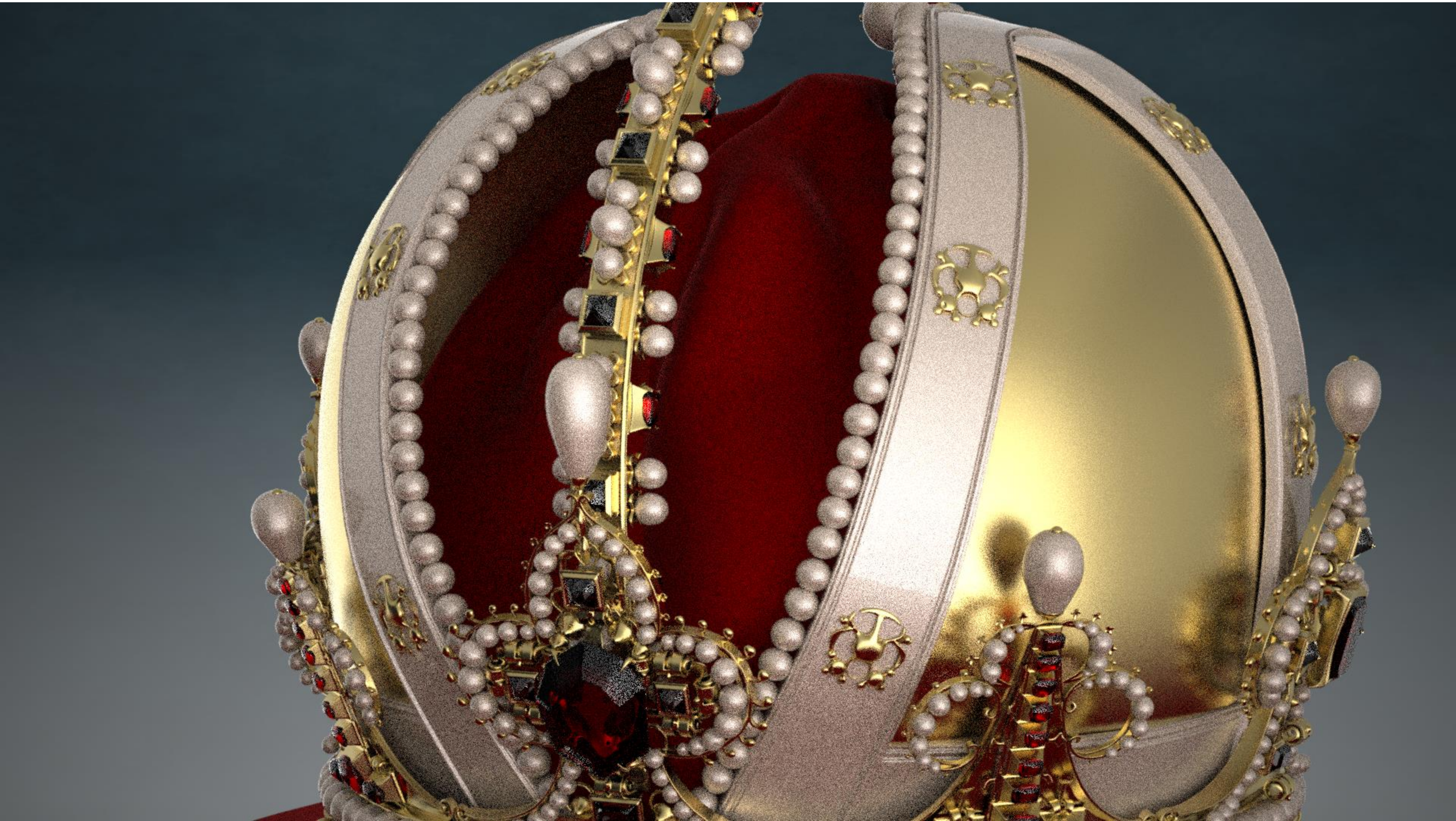
- Samuli Laine for use of Vegetation and Hairball

- Martin Lubich & Intel for use of Crown

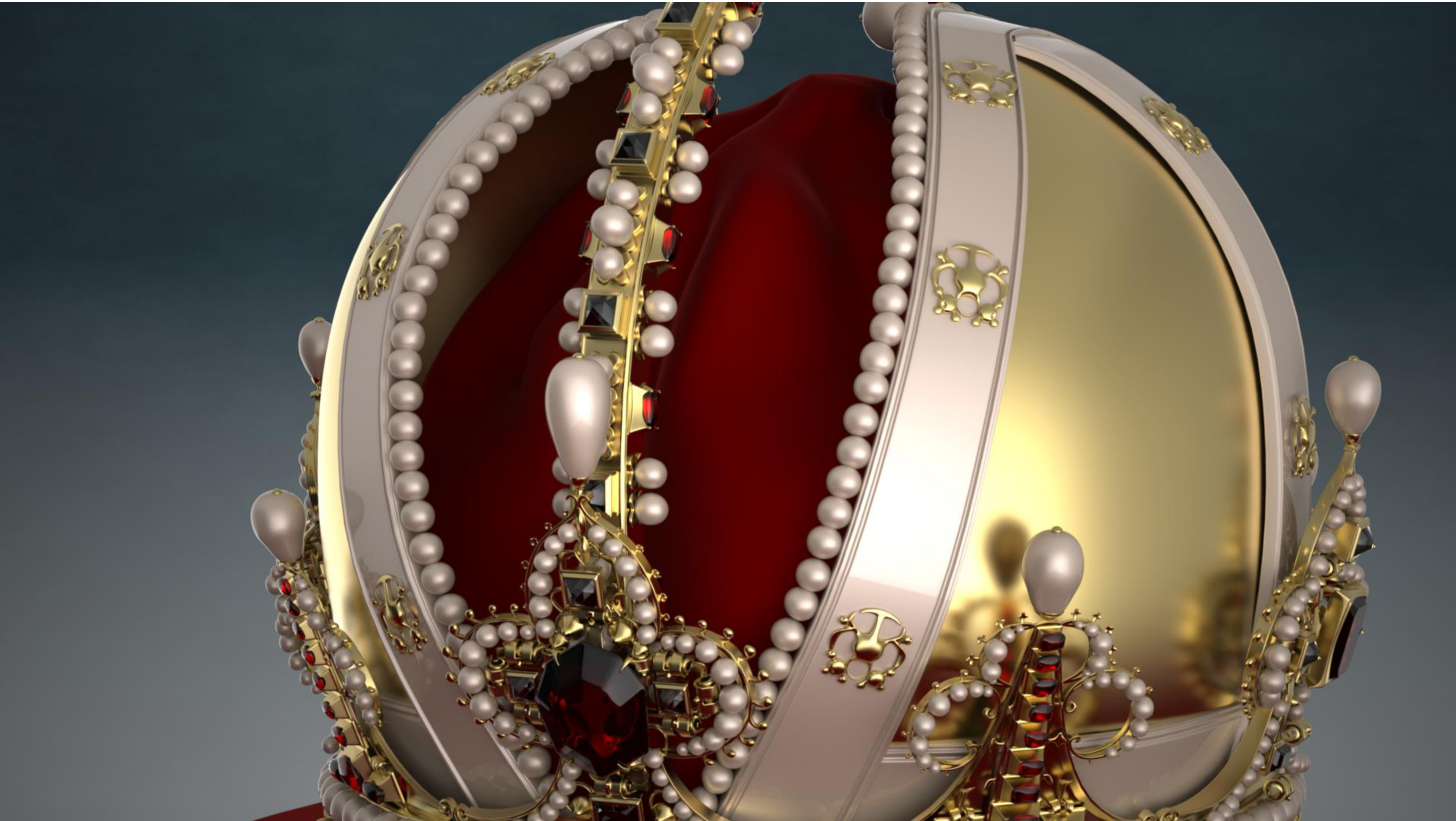
- Daniel Kopta for assistance with comparisons in the paper

- UT Graphics group and others for many useful discussions

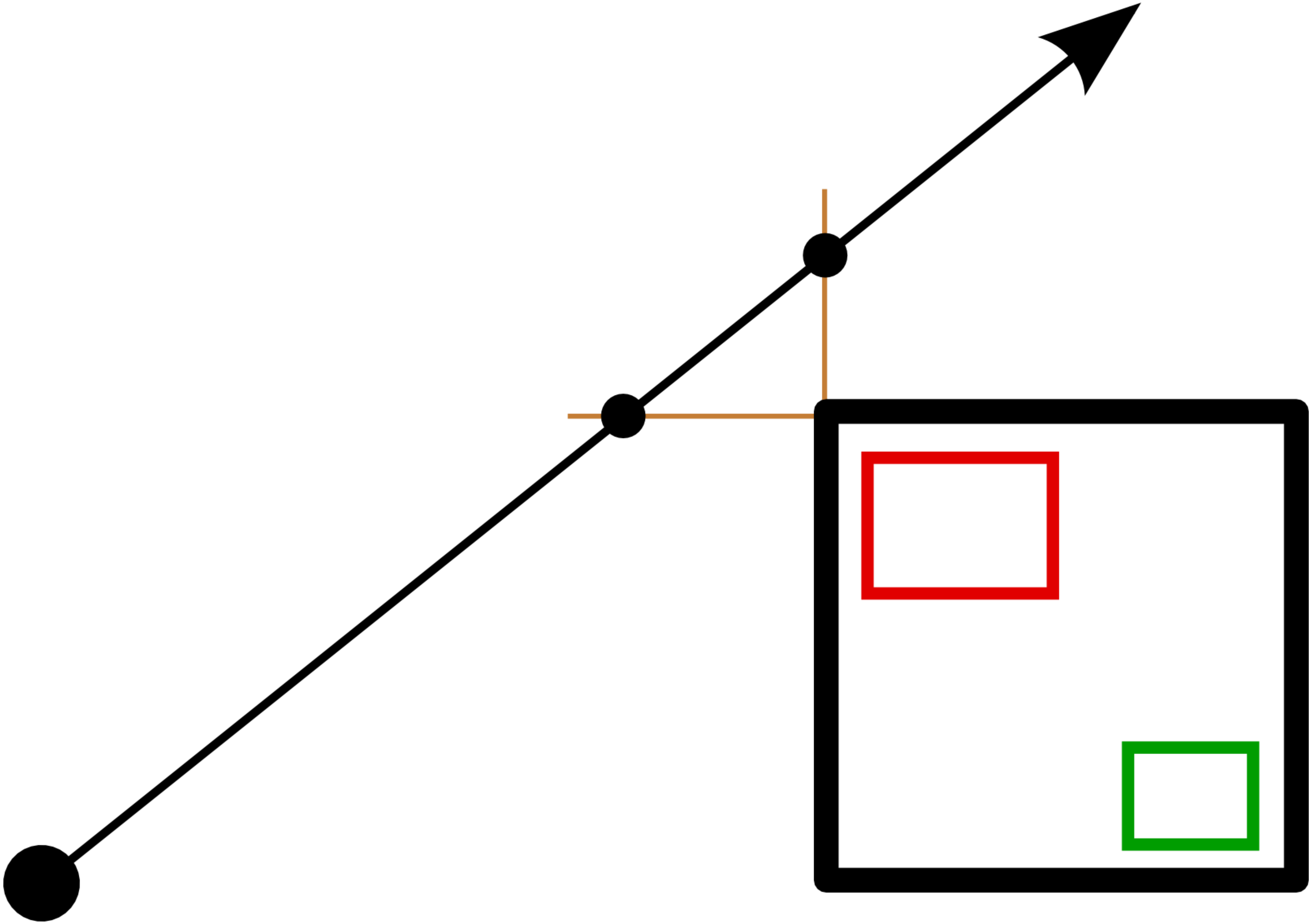
Questions?



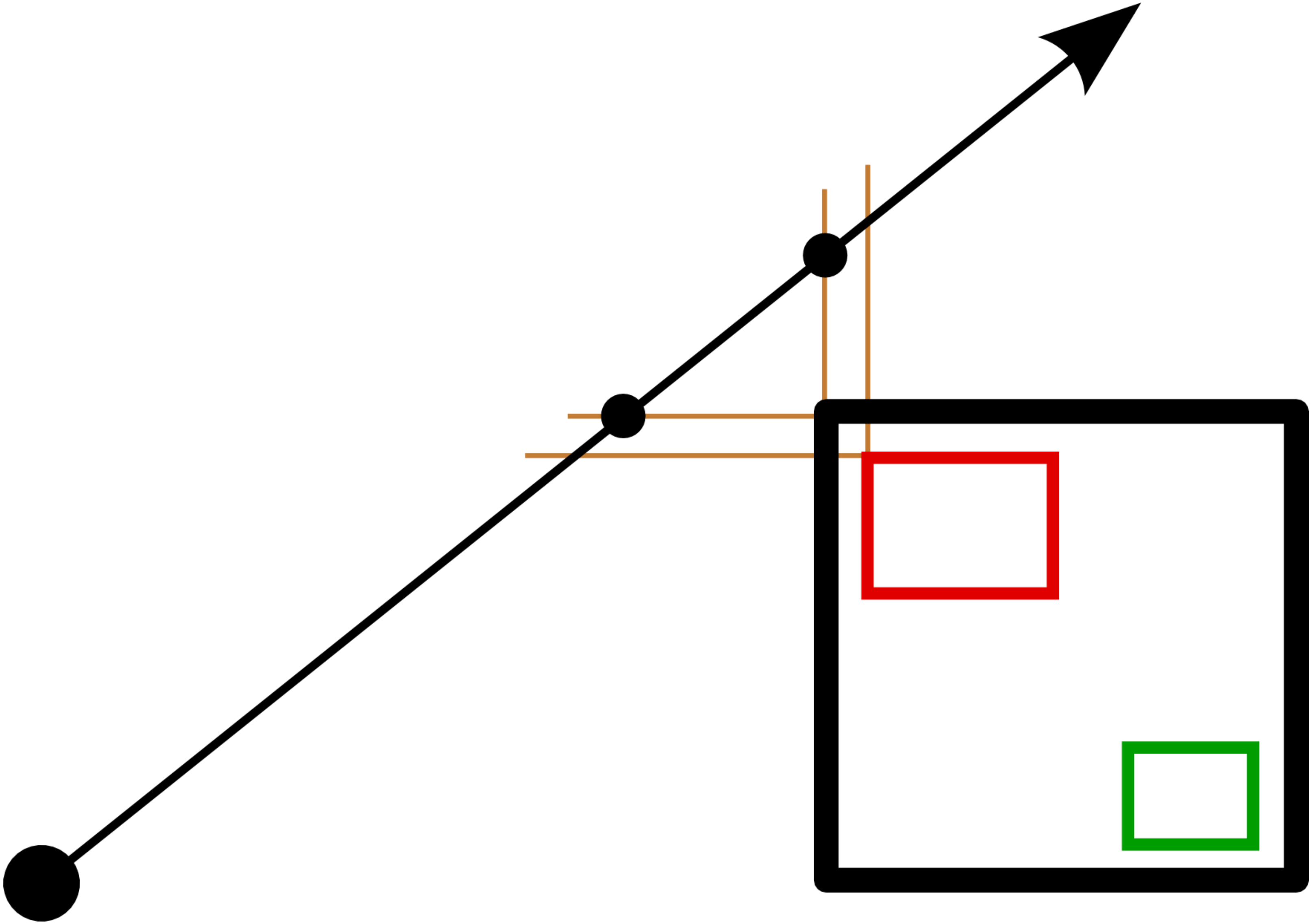
Ground Truth



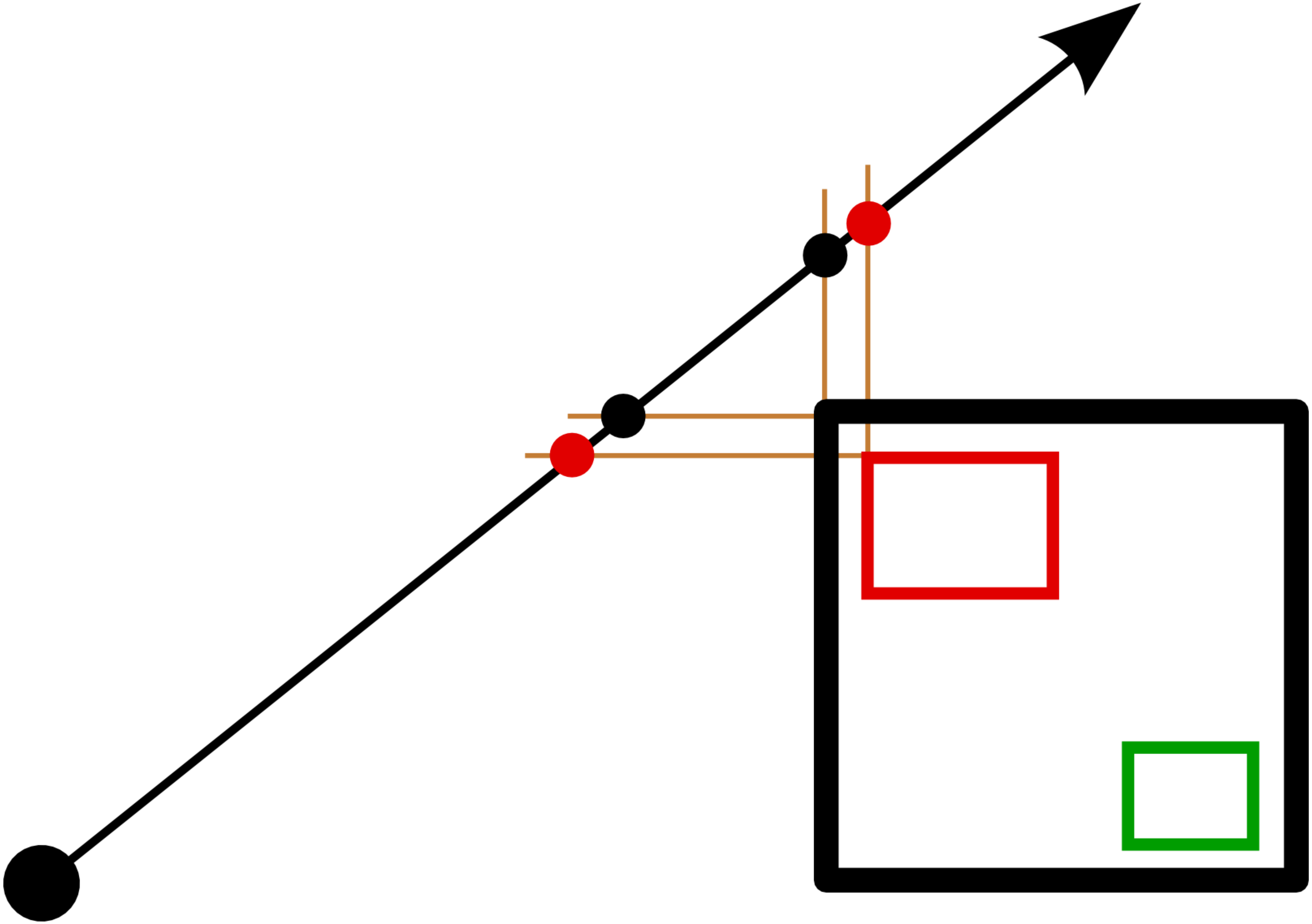
Incorrectly taken paths abort quickly



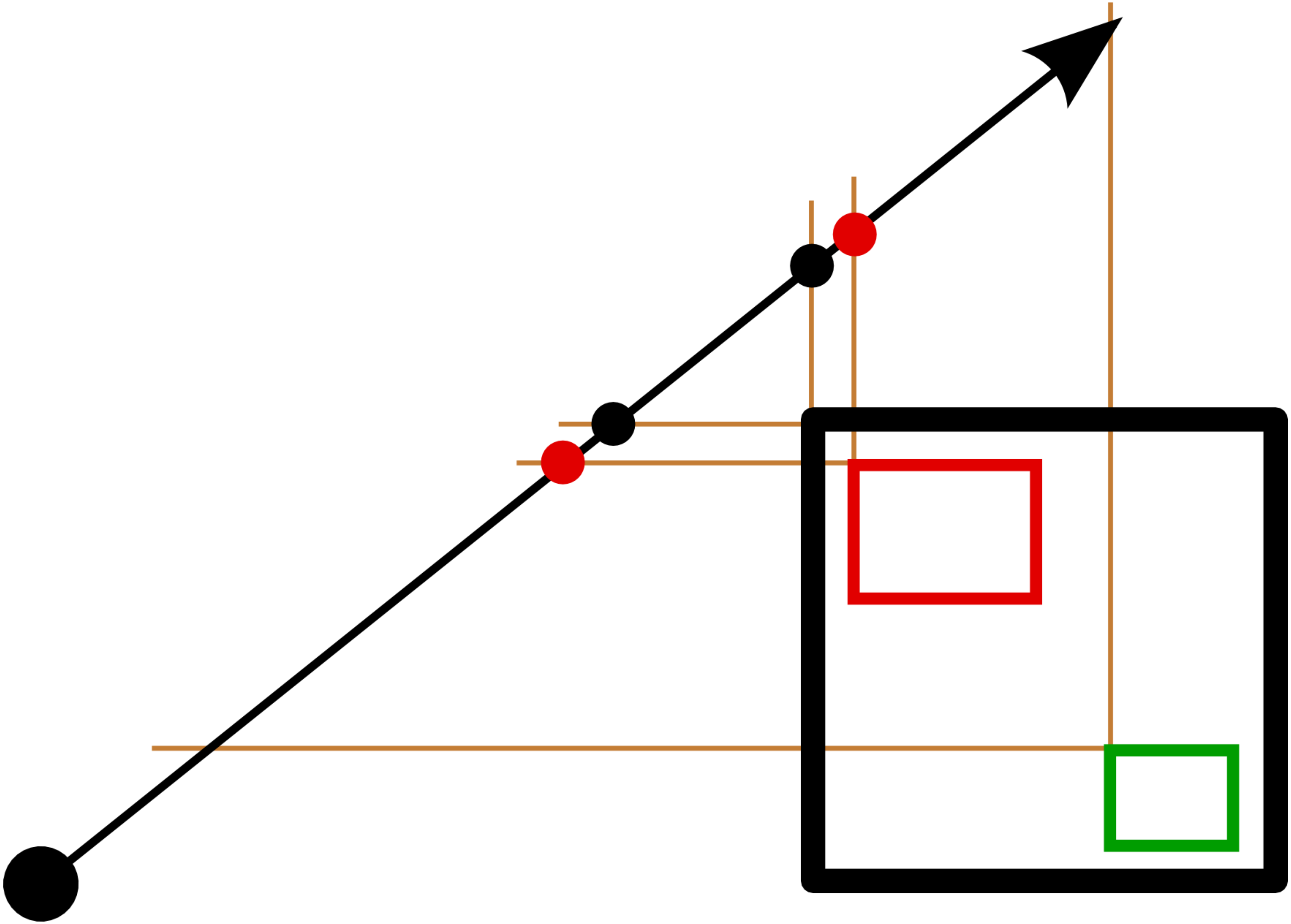
Incorrectly taken paths abort quickly



Incorrectly taken paths abort quickly



Incorrectly taken paths abort quickly



Incorrectly taken paths abort quickly