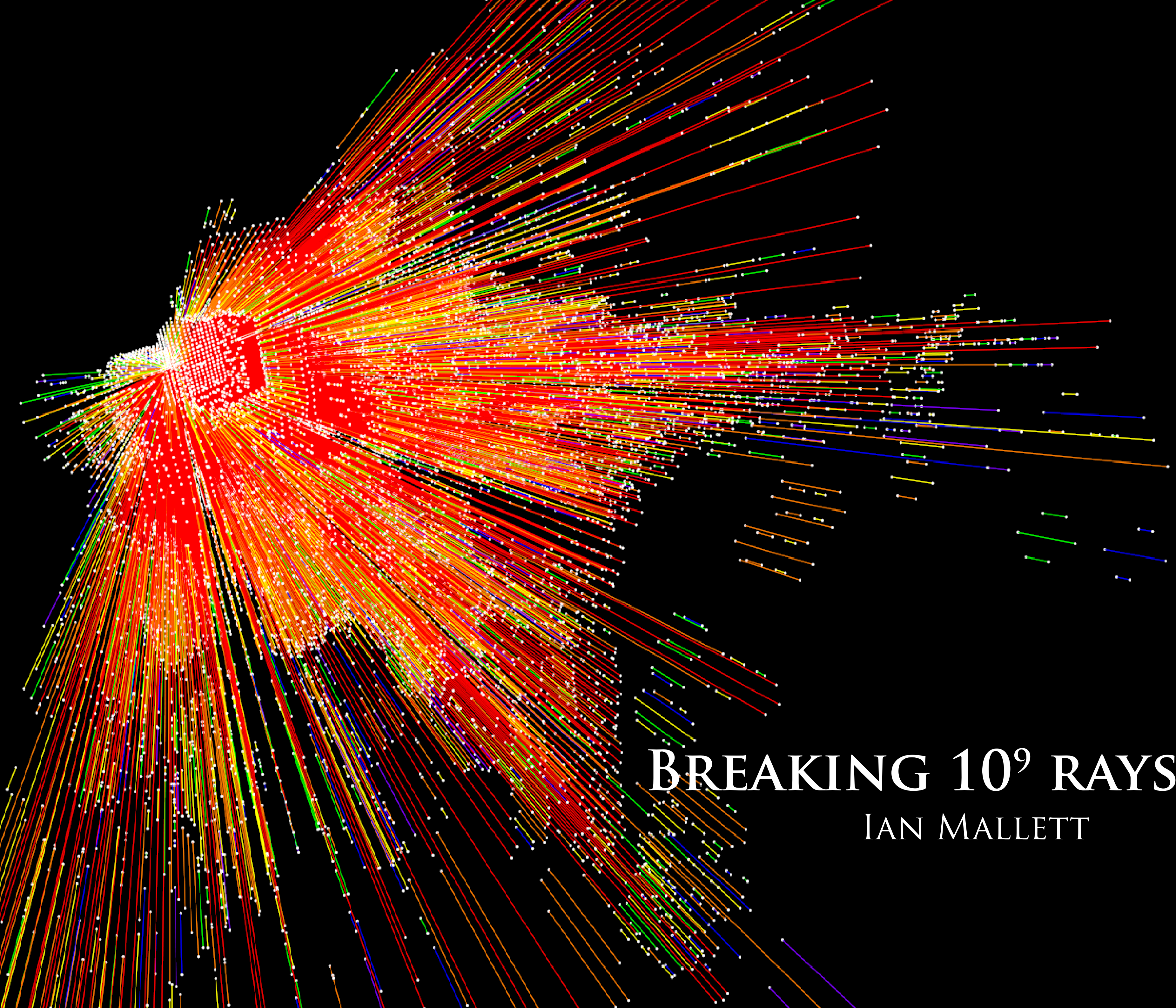
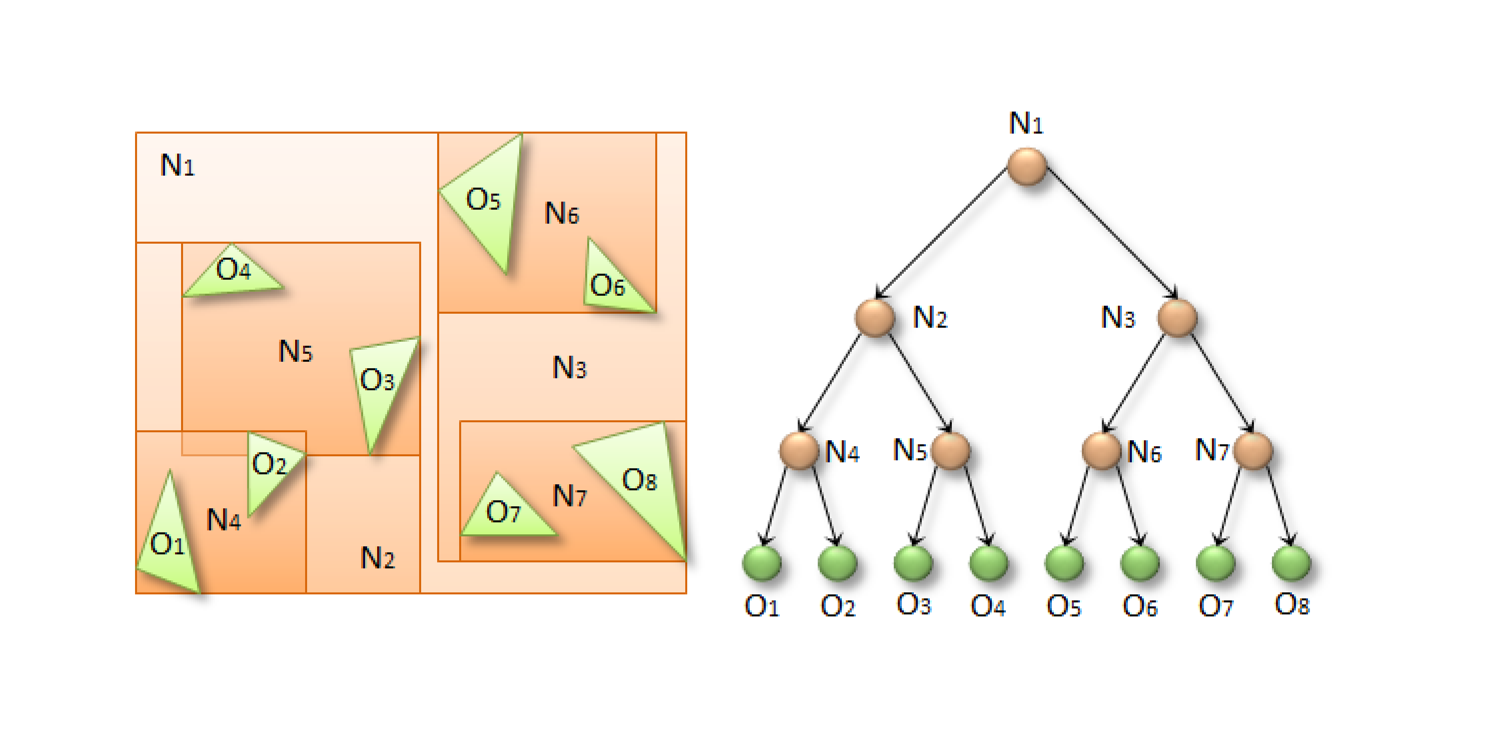


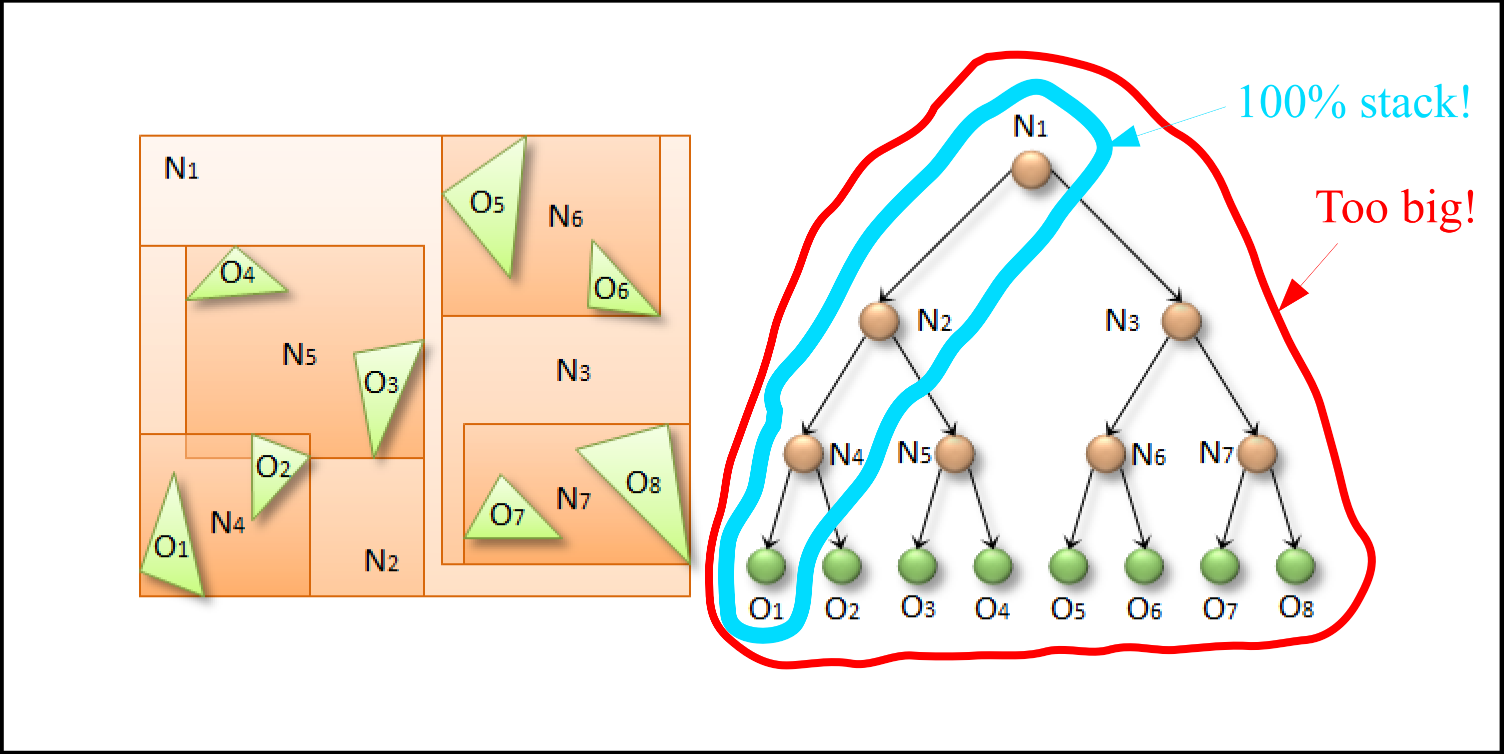
Breaking 10^9 rays/sec

Ian Mallett

Idea 1 of 3


N1
O5
N6
O6
O4
N5
O3
N3
O2
N4
O1
N2
O7
N7
O8
N1
N2
N3
N4
N5
N6
N7
O1
O2
O3
O4
O5
O6
O7
O8

Idea 1 of 3



100% stack!

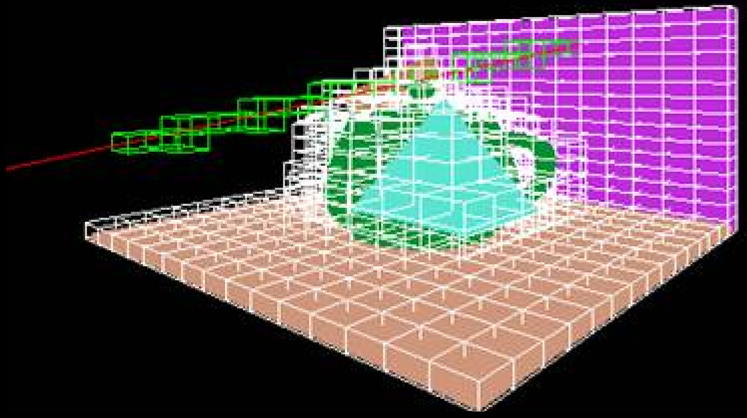
Too big!

Idea 1 of 3

- Advantages:
  - Secondary rays leave from where primary rays end!
  - Redundancy (e.g., we *always* need the root node)
  - Adapts to scene; loaded data is pooled corresponding to where rays concentrate
- Disadvantage
  - Still per-thread . . .

Idea 2 of 3

- Store uniform grid bitmap on stack; maps to 3D array of BVHes.
- Use DDA to step through
- Advantage:
  - Fast coarse traversal (and coarse traversal is the most expensive!)



Idea 3 of 3

- Ray rescheduling:
  - After DDA trace, spit rays into atomic queues
  - Threads pull from the largest queues (one queue each)
  - Dynamically reschedule themselves
- Advantage:
  - Queuing hides memory latency!
  - Threads concentrate work where it's needed!
  - No/Very little scene traffic except when a thread context switches (rare, especially in the worst scenes)!
- Disadvantage:
  - Complex. Might not happen (optional).