GiST Scan Acceleration Using Coprocessors

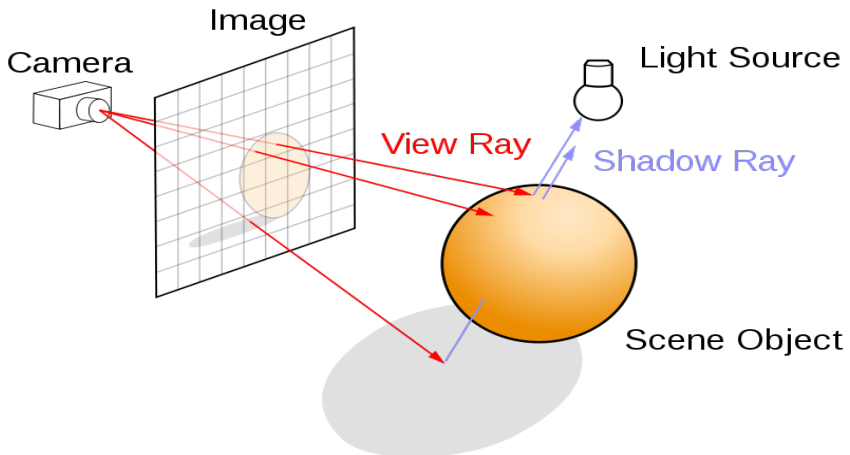
Felix Beier, Torsten Kilias, Kai-Uwe Sattler

Ilmenau University of Technology

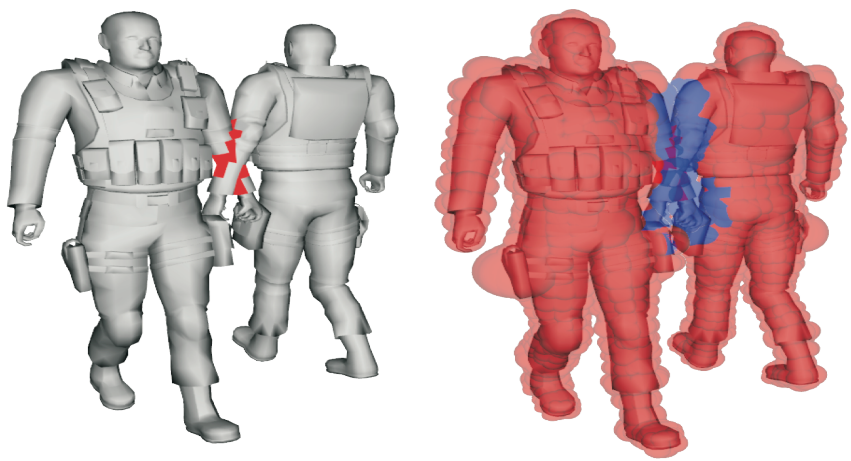
05/21/2012

Outline

Co-processing Index Searches



Source: [1]

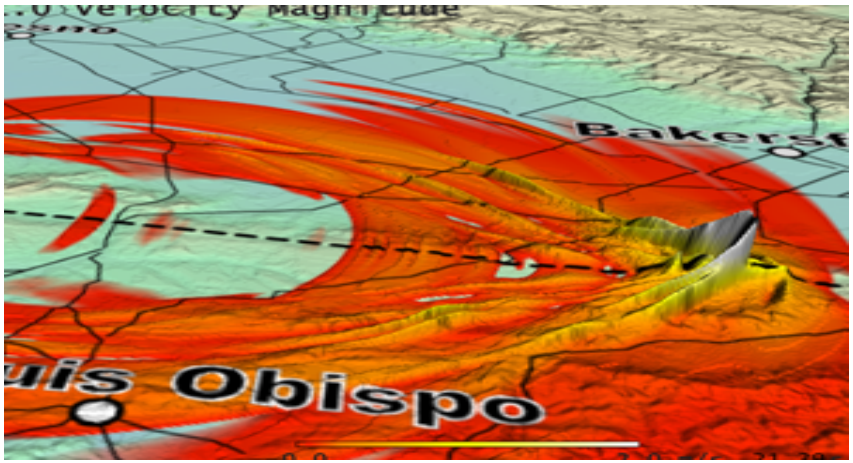


Source: [5]

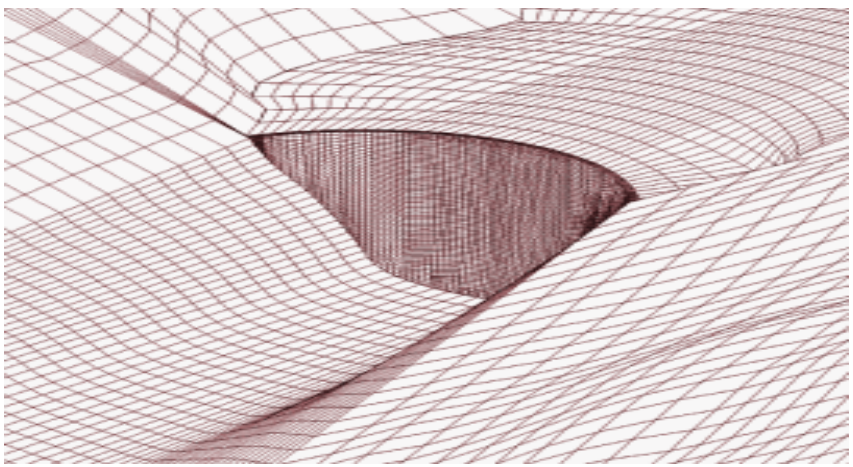
- Ray tracing: many independent point queries
- Collision detection (spatial join): many independent range queries
- Utilization of massive parallelism offered by modern coprocessors

→ Special index structures carefully tuned for specific hardware

Index Frameworks



Source: [3]



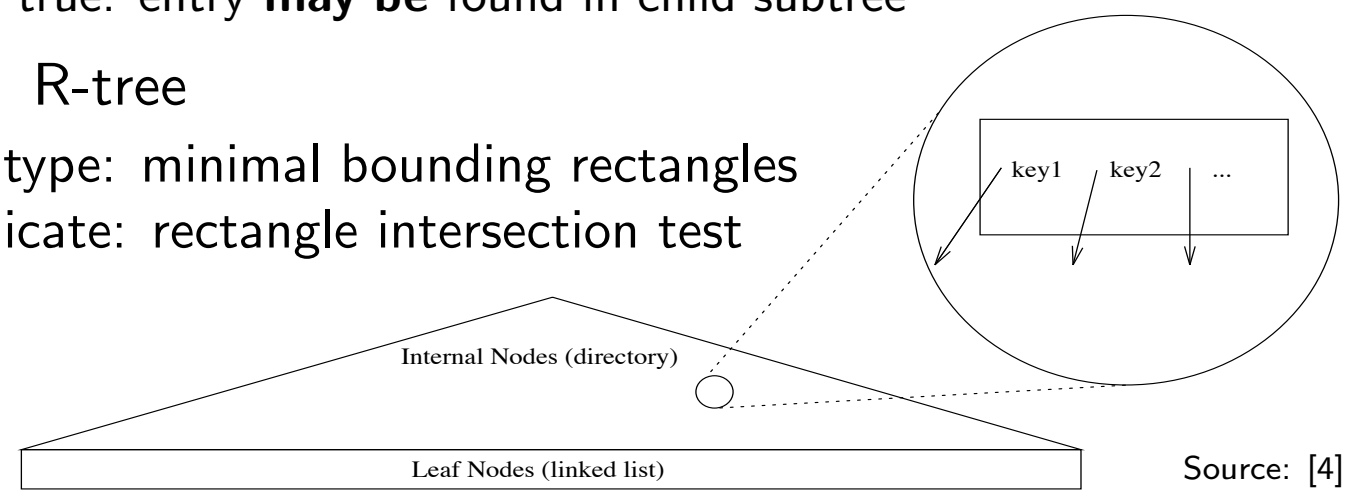
Source: [2]

- Various applications require specialized index structures
- Scientific data
  - Enormous data volumes
  - Unknown data characteristics
  - Costly prototyping
  - Gap: scientists vs. system developers

→ Rapid index development with frameworks like GiST

GiST - Generalized Index Search Tree

- Framework for implementation of height-balanced search trees
  - Implements common tree operations (insertions, deletions, node splits, height-balancing)
  - Developer specifies key data type and type-specific operations
  - Lookup predicate returns
    - false: entry can **not** be found in child subtree
    - true: entry **may be** found in child subtree
- Example: R-tree
  - Key type: minimal bounding rectangles
  - Predicate: rectangle intersection test



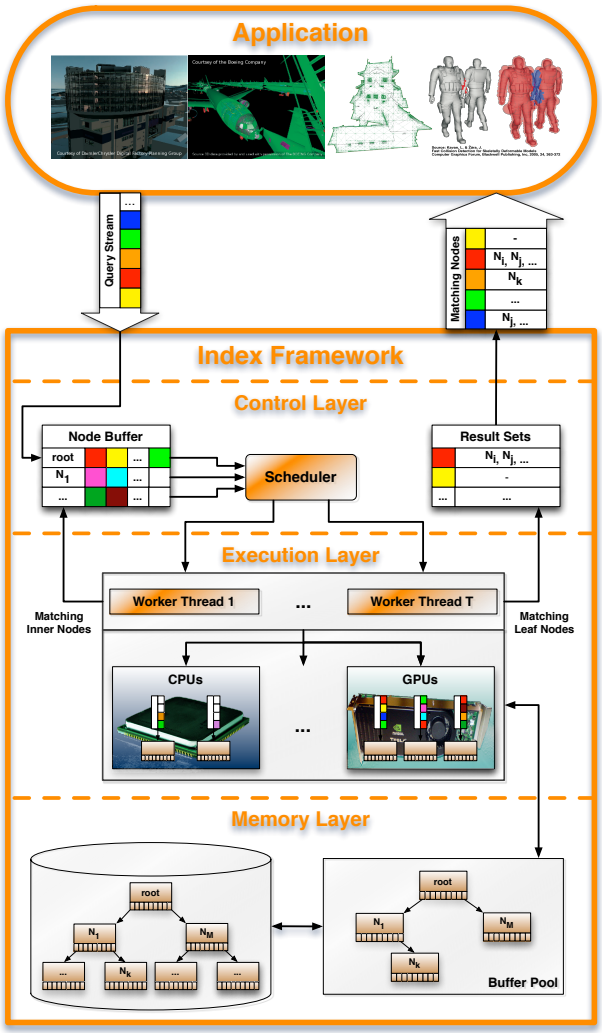
Source: [4]

Motivation

- Combining the best of both worlds
  - Extensibility of GiST framework
  - Performance improvements through co-processing
- Challenges
  - Finding fine-grained parallel algorithms
  - Utilizing hardware capabilities
  - Out-of-core implementation
  - Consideration of co-processing overheads

Framework Design

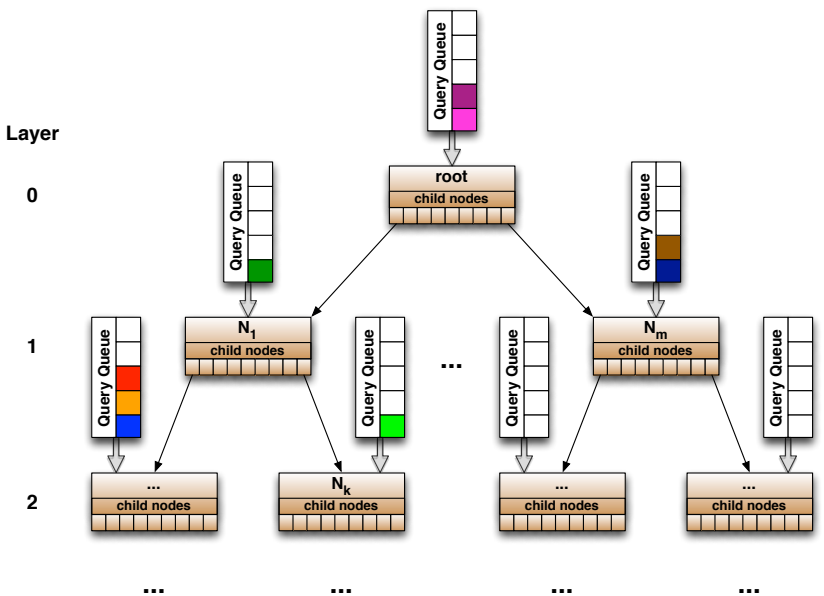
- Applications issue stream of query batches
- Iterator for matching leaf nodes
- Grouping queries to node batches for better locality
- Specialized scan implementation for various (co)processors
- Automatic scheduling to best execution unit
- Out-of-core implementation



Scan Parallelization

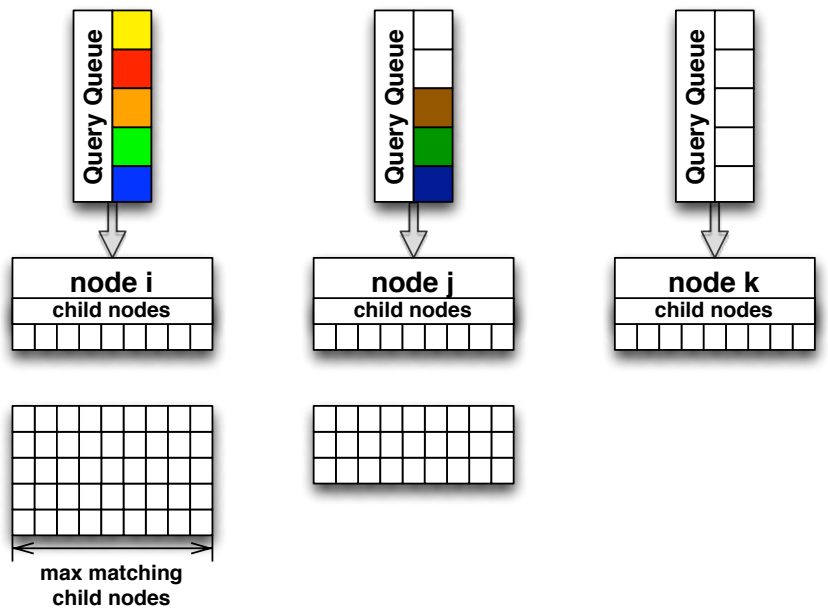
Inter-Node Parallelization:

- Independent node batches
- Pipelining



Intra-Node Parallelization:

- Independent predicate tests
- SIMD features



GPU Implementation - Processing Model

- Nvidia CUDA implementation
- Multiple cores per GPU device
  - Execution of independent subtasks without synchronization
  - Subtask = node - query batch pair
- Multiple thread processors per GPU core
  - Execution of data parallel instructions by separate threads
  - Synchronization possible
  - Instruction = predicate test

GPU Implementation - Memory Hierarchy

- Global main memory on device
  - Explicit transfer from host memory via PCIe bus
  - Caching of node and query data to avoid transfer overhead
  - Scan preparation phase to determine input data offsets
- Shared memory on each die
  - Two orders of magnitude faster than global memory
  - Software-controlled cache for scan data

Setup

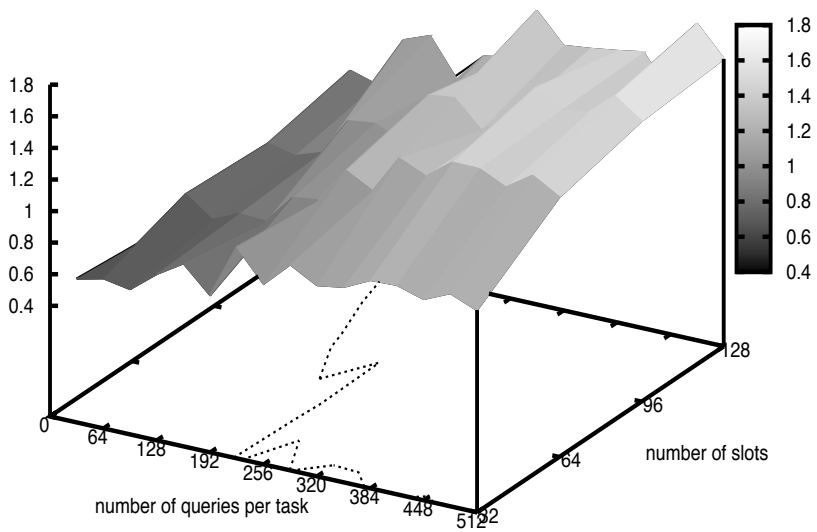
- 3-D R-tree implementation
- Generated index nodes
- Intel Xeon X5690 CPU
- Nvidia Tesla C2050 GPU

→ Where shall a scan be executed?

→ How can the performance be improved with hybrid processing?

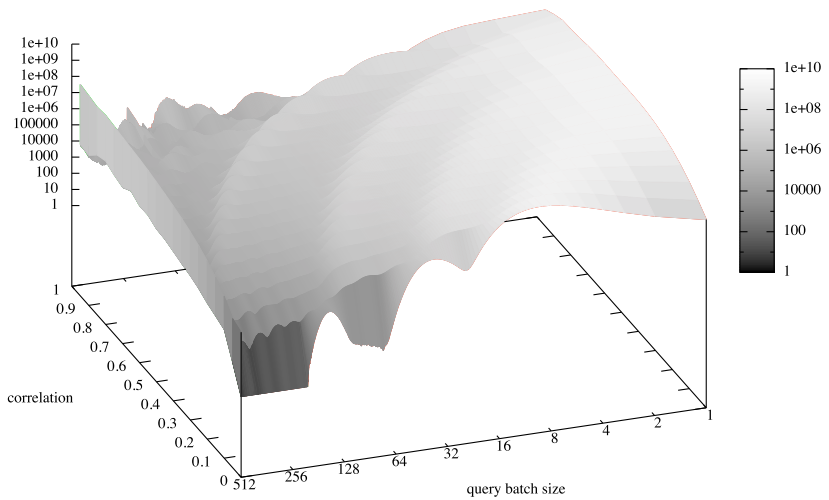
Tree Parameters

- Generated index nodes
- Generated predicates
- Full processor utilization
- Overheads included for GPU measurements
- $speedup = \frac{CPU\ \ time}{GPU\ \ time}$

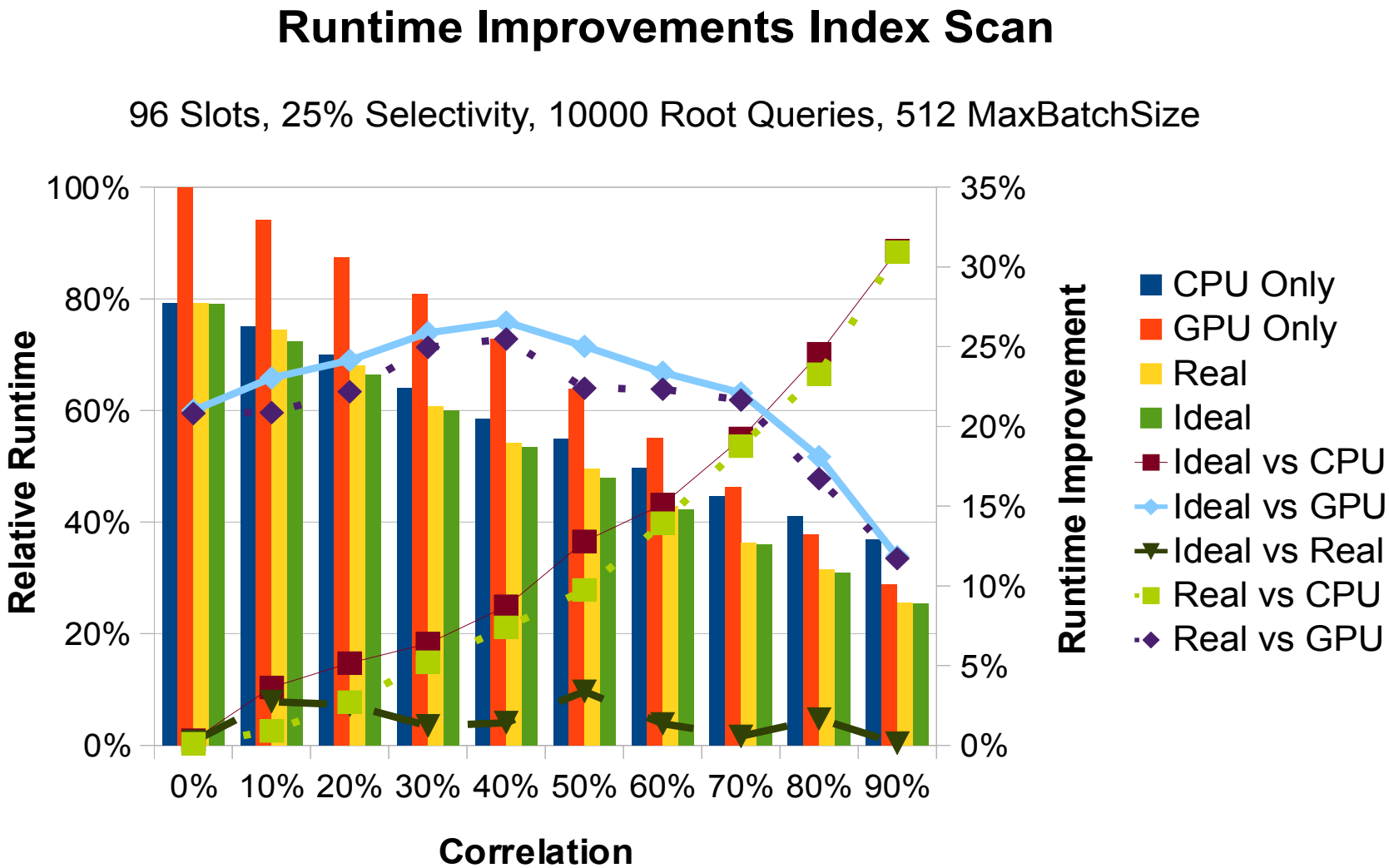


Workload Simulation

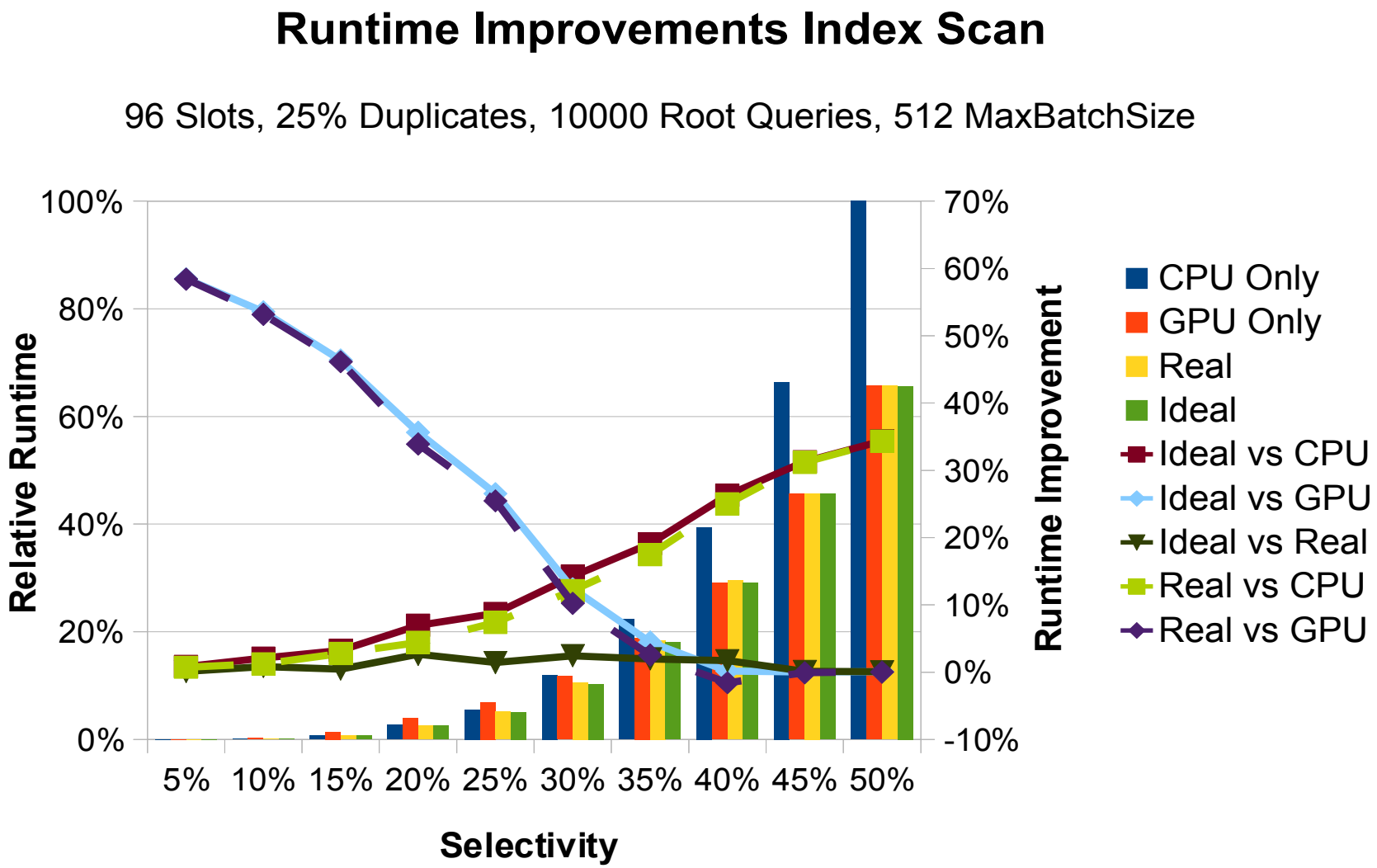
- CPU for small batches
- GPU for large batches
- How do batch sizes change when queries are streamed through the tree?
- Simulation for full R-tree
  - 96 slots per node
  - 5 layers $\rightarrow$ 8 billion indexed entries
  - 10.000 root queries



Workload Simulation - Parameter Correlation



Workload Simulation - Parameter Selectivity



Conclusion & Outlook

Conclusion

- Extended GiST with hardware abstraction layer
- Performance improvements are possible
- Overheads are not negligible!

Next steps

- Prototype improvements
- Specialization for other tree types
- Full support for all GiST operations

References

[1] http://en.wikipedia.org/wiki/File:Ray_trace_diagram.svg.

[2] Science and Technology Review, June 2007.
"Virtual Dams Subjected to Strong Earthquakes".

[3] Chourasia, A., Olsen, K., Cui, Y., Lee, K., Zhou, J., Ely, G., Small, P., Roten, D., Day, S., Maechling, P., Jordan, T., Panda, D. K., and Levesque, J.
Ground motion visualization of M8 earthquake simulation using height field.
In *SciDAC* (2011).
Available at http://www.mcs.anl.gov/uploads/cels/papers/scidac11/.

[4] Hellerstein, J. M., Naughton, J. F., and Pfeffer, A.
Generalized Search Trees for Database Systems.
In *VLDB* (1995).

[5] Kavan, L., and Zara, J.
Fast Collision Detection for Skeletally Deformable Models.
Computer Graphics Forum 24, 3 (2005), 363–372.

Discussion

Thanks for your attention!

Questions?