OBJECT DISCOVERY FOR VIRTUAL WORLDS
metaverses are user-generated worlds
metaverses

are user-generated worlds
metaverses

are user-generated worlds
metaverses

are user-generated worlds
metaverses

are user-generated worlds
metaverses are user-generated worlds
metaverses are user-generated worlds
Object Discovery

which objects should I know about?
Object Discovery

which objects should I know about?

display?
Object Discovery

which objects should I know about?
display?
interact with?
Object Discovery

which objects should I know about?
display?
interact with?

bootstraps everything
Object Discovery

Virtual World System

Client
Object Discovery

Client → Query → Virtual World System
Object Discovery

Client

Query

Object IDs

Virtual World System
Object Discovery

Client

Query

Object IDs

Virtual World System

gemetric
Object Discovery

Client

Query

Object IDs
Initial Set + Updates

Virtual World System

geometric  continuous
Object Discovery

Client

Query

Virtual World System

Object IDs
Initial Set + Updates

distorted

geometric  continuous
distributed
distance queries
distance queries
distance queries
distance queries
distance queries
distance queries scale trivially, but limit scope
Scalability/Distributed

User Generated

Less

More

One Server

Many Servers

Scalability/Distributed

Visual Fidelity

Low

High

Minecraft

The Sims

Half-Life/Counter-Strike

Neverwinter Nights

Multiplayer Game

WoW

EVE Online

ActiveWorlds

There

Second Life

OpenSim

There

Second Life

OpenSim

The Sims

Half-Life/Counter-Strike

Neverwinter Nights

Multiplayer Game

WoW

EVE Online

ActiveWorlds

There

Second Life

OpenSim
challenge: see (interact with) the entire user-generated world yet scale to large worlds
what type of query provides an immersive experience?
only a global query can provide an immersive experience
solid angle queries

select objects by how large they appear to the querier
solid angle queries

select objects by how large they appear to the querier
solid angle queries

select objects by how large they appear to the querier
solid angle queries

select objects by how large they appear to the querier
solid angle queries

select objects by how large they appear to the querier
Solid Angle & Aggregates, 3000 Objects
Related Work

Builds on existing data structures & algorithms:

- R-Tree
- BVH
- BVH Refitting
- Async BVH
- LOD/Simplification
- Safe Periods/Lazy Updates
- Query Aggregation
- Approximate Queries

But has a unique set of requirements:

<table>
<thead>
<tr>
<th></th>
<th>Distributed</th>
<th>Global View</th>
<th>Aggregation</th>
<th>Dynamic &amp; User Generated</th>
<th>Continuous</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobiEyes</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Nam &amp; Sussman</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>GigaVoxels</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>HLOD</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>StarTrack</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Chaudhuri et al.</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
OUTLINE

Motivation: Metaverses & Immersion

Local Queries
  Query Data Structure
  Incorporating Aggregation
  Cuts to Ensure Complete Views

Scaling & Distribution

Applications & Extensions

Conclusion
how are geometric queries evaluated efficiently?
goal: quickly cull many objects from consideration
goal: quickly cull many objects from consideration
goal: quickly cull many objects from consideration
BOUNDING VOLUME HIERARCHY

goal: quickly cull many objects from consideration
BOUNDING VOLUME HIERARCHY

goal: quickly cull many objects from consideration
BOUNDING VOLUME HIERARCHY

goal: quickly cull many objects from consideration
Bounding Volume Hierarchy

culls many nodes for distance queries
track largest object in each subtree

Largest
Bounding
Volume
Hierarchy
aggregates
add objects that appear too small individually
aggregates are cheaper to download and render
aggregates are cheaper to download and render
aggregates are cheaper to download and render
aggregates are cheaper to download and render
aggregates are cheaper to download and render
The diagram illustrates how aggregates are cheaper to download and render.
cuts

ensure queries see the entire world

(and make query updates faster too)
Cuts avoid redundant work
20 - 56% faster query evaluation
Cuts avoid redundant work
20 - 56% faster query evaluation
Cuts avoid redundant work
20 - 56% faster query evaluation
Cuts avoid redundant work
20 - 56% faster query evaluation
Cuts avoid redundant work
20 - 56% faster query evaluation
LBVH culls subtrees for efficient solid angle queries

tests 75-90% fewer nodes then a BVH
LBVH culls subtrees for efficient solid angle queries
tests 75-90% fewer nodes than a BVH

cuts ensure a complete view
20-56% more efficient
LBVH culls subtrees for efficient solid angle queries
tests 75-90% fewer nodes then a BVH

cuts ensure a complete view
20-56% more efficient

static/dynamic split handles moving objects efficiently
10-15% more efficient
LBVH culls subtrees for efficient solid angle queries
tests 75-90% fewer nodes than a BVH

cuts ensure a complete view
20-56% more efficient

static/dynamic split handles moving objects efficiently
10-15% more efficient

aggregates provide simplified meshes to fill in the complete view
LBVH culls subtrees for efficient solid angle queries
tests 75-90% fewer nodes then a BVH

cuts ensure a complete view
20-56% more efficient

static/dynamic split handles moving objects efficiently
10-15% more efficient

aggregates provide simplified meshes to fill in the complete view on one server...
Motivation: Metaverses & Immersion

Efficient Local Queries

Scaling & Distribution
  Traditional Architectures Don’t Scale
  Global Aggregation Tree

Applications & Extensions

Conclusion
how can we efficiently evaluate global queries across many servers?
what do other distributed query processors do?
We distribute query processing when we have:

a) too much data
b) too many queries
c) both

A Taxonomy by Kossmann 2000
query shipping
(too much data)
query shipping
(too much data)
query shipping
(too much data)
query shipping
(too much data)
query shipping
(too much data)
query shipping
(too much data)

overloads servers with “popular” objects
data shipping
(too many queries)
data shipping
(too many queries)
data shipping
(too many queries)
data shipping
(too many queries)

Client

Query

Data Request

Replicated Data

Server

D1

D2

D3

D4
data shipping
(too many queries)

Client → Query → Server → Results

Data Request

Replicated Data

D1 D2 D3 D4

Server
data shipping
(too many queries)

(global queries require the entire dataset)
hybrid shipping
(too much data, too many queries)
hybrid shipping
(too much data, too many queries)
hybrid shipping
(too much data, too many queries)
hybrid shipping
(too much data, too many queries)
hybrid shipping
(too much data, too many queries)
hybrid shipping
(too much data, too many queries)

looks like a good balance?
why doesn’t this hybrid approach scale well?
why doesn’t this hybrid approach scale well?

aggregates
hybrid shipping (PIntO)
hybrid shipping (PIntO)

Queries

Client

Local Tree

Server

Client

Server

Server
hybrid shipping (PIntO)
hybrid shipping (PIntO)

Queries

Client

Local Tree

Server

Aggregated Query

Server

Server
hybrid shipping (PIntO)
hybrid shipping (PIntO)
hybrid shipping (PIntO)
hybrid shipping (PIntO)

- Queries
- New aggregates
- Results
- Combined Tree
- Local Tree
- Server
- Aggregated Query
- Replicated Data
- Server
- Client

Diagram showing the flow of data and queries between clients, servers, and trees.
<table>
<thead>
<tr>
<th>Model</th>
<th>Mesh</th>
<th>Texture</th>
<th>Generation Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree</td>
<td>39 KB</td>
<td>1,474 KB</td>
<td>-</td>
</tr>
<tr>
<td>House</td>
<td>16 KB</td>
<td>521 KB</td>
<td>-</td>
</tr>
<tr>
<td>Level 2</td>
<td>384 KB</td>
<td>187 KB</td>
<td>721 ms</td>
</tr>
<tr>
<td>Root Aggregate</td>
<td>1,238 KB</td>
<td>1,549 KB</td>
<td>9,431 ms</td>
</tr>
</tbody>
</table>

aggregates are expensive
hybrid shipping (PIntO)

Client

Queries

Local Tree

Server

Combined Tree

Results

Aggregated Query

Replicated Data

minor query changes trigger aggregate (re)computation
hybrid shipping (PIntO)

Client ➔ Queries ➔ Local Tree ➔ Combined Tree ➔ Server ➔ Results

Aggregated Query ➔ Aggregated Data

Replicated Data

minor query changes trigger aggregate (re)computation
hybrid shipping (PIntO)

Queries

Client

Client

Results

Local Tree

Combined Tree

Server

Aggregated Query

Replicated Data

Server

Client

minor query changes trigger aggregate (re)computation
hybrid shipping (PIntO)

Client 

Queries

Local Tree

Combined Tree

Server

Aggregated Query

Replicated Data

Results

major query changes trigger aggregate (re)computation
hybrid shipping (PIntO)

Minor query changes trigger aggregate (re)computation
hybrid shipping (PlntO)

Client

Queries

Local Tree

Combined Tree

Server

Aggregated Query

Replicated Data

Client

Results

minor query changes trigger aggregate (re)computation
hybrid shipping (PIntO)

- Queries
  - Client
  - Local Tree
    - Combined Tree
      - Server
      - Replicated Data
      - Aggregated Query

- Results
  - minor query changes trigger aggregate (re)computation
  - Server
hybrid shipping (PIntO)

Queries

Client

Local Tree

Combined Tree

Server

Aggregated Query

Replicated Data

Results

minors query changes trigger aggregate (re)computation
hybrid shipping (PIntO)

each server generates its own combined tree based on different client queries
composing local query processors is very expensive because many aggregates must be created, stored, and downloaded
Global Aggregation Tree

one global tree constructed & shared by all servers

that

minimizes creation of aggregates
Global Aggregation Tree

distributed bounding volume hierarchy
Global Aggregation Tree

Top Level Server

replaces all combined trees

distributed bounding volume hierarchy
Global Aggregation Tree

servers don’t answer queries

instead

clients answer queries, servers replicate data
multi-level replication
multi-level replication

Replication Commands

Client

Server

Top Level Server

Server

Server
multi-level replication

Client

Replication Commands

Server

Replication Commands

Top Level Server

Server

Server
multi-level replication

Client → Server

Replication Commands

Top Level Server

Server

Server
multi-level replication

Client

Replication Commands

Server

Replication Commands

Top Level Server

Server

Server

Replicated Data
multi-level replication

Replication
Commands

Client

Server

Replication
Commands

Top Level
Server

Server

Server

Replicated
Data
multi-level replication

Replication Commands

Client

Replicated Data

Server

Replicated Data

Top Level Server

Server

Server

Replication Commands
multi-level replication

Replication Commands

Client

Replicated Data

Server

Replicated Data

Replication Commands

Top Level Server

Server

Server
multi-level replication

Replication Commands

Client

Replicated Data

Server

Replicated Data

Top Level Server

Server

Server
multi-level replication

Client

Replication Commands

Server

Replication Commands

Top Level Server

Server

Server

Replicated Data

Replicated Data

Replicated Data
multi-level replication
multi-level replication

Client

Replication Commands

Replicated Data

Server

Replication Commands

Replicated Data

Top Level Server

Server

Server
multi-level replication
multi-level replication

Replication Commands

Client

Replicated Data

Server

Replicated Data

Replication Commands

Top Level Server

Server

Server
multi-level replication

Replication Commands

Client

Replicated Data

Server

Replicated Data

Top Level Server

Server

Server
multi-level replication

Replication Commands

Client

Replicated Data

Server

Replication Commands

Top Level Server

Server

Server

Replicated Data

Replicated Data
benefits
benefits

no new aggregates created when evaluating queries
benefits

no new aggregates created when evaluating queries
benefits

no new aggregates created when evaluating queries

simple, fine-grained replication of Global Aggregation Tree
replication protocol

commands are simple
(init, refine, coarsen, destroy)

but updates are tricky
(maintain consistent & connected tree structure, efficiently encode complex changes, e.g. move entire subtree)
evaluation:

does it scale?

how does it affect clients?
Workloads

We don’t actually have millions of users...
Workloads

We don’t actually have millions of users...

Second Life traces
(object layout, density x 256, avatar/object movement)

+ 

Procedurally generated scenes

+ 

Tiling
Comparison

PIntO
naïve hybrid shipping
first system design

Global Aggregation Tree
distributed query index
new system design
# of aggregates generated per server remains constant
Global Aggregation Tree creates less expensive aggregates
migration has minimal effect on GAT and client
Global Aggregation Tree

distributed query index
(single global data structure)
minimizes aggregates
Global Aggregation Tree

- distributed query index
  (single global data structure)
- minimizes aggregates
- weak consistency requirements
- permit efficient pair-wise replication
Global Aggregation Tree

distributed query index
(single global data structure)
minimizes aggregates

weak consistency requirements
permit efficient pair-wise replication

multi-level replication
pushes query evaluation to clients
and allows flexible queries
Global Aggregation Tree

distributed query index
(single global data structure)
minimizes aggregates

weak consistency requirements
permit efficient pair-wise replication

multi-level replication
pushes query evaluation to clients
and allows flexible queries

prefetching policy
reduces query latency caused by chatty protocol
Motivation: Metaverses & Immersion
Efficient Local Queries
Scaling & Distribution
Applications & Extensions
  Rendering
  Object Scripting
  Beyond Metaverses
Conclusion
object scripts often interact with nearby objects
distance queries are easily implemented
Customize

record format
(mesh, tweet)

aggregate data and algorithm
(mesh, trending topic)

query format and algorithm
(solid angle, search term)

index data
(largest object, Bloom filter)
good for

large, dynamic geometric data
multi-resolution aggregate display
online exploration
contributions

**LBVH**
ensures complete view for a single server

**GAT**
scalable & flexible geometric queries with aggregates

**Generalization**
multi-resolution online exploration of large geometric datasets
THANKS