What Is It?

- Software platform for 3D virtual worlds
- Princeton/Stanford collaboration

- Goal: A user should be able to interact with the whole virtual world.
- Goal: Easy application development.
Sirikata Architecture
Sirikata Architecture

Computation

Object

Hosts
Sirikata Architecture

Communication

Space Servers

Computation

Object Hosts
Sirikata Architecture

Communication

Space Servers

Computation

Object Hosts

Storage

Content Distribution Network
A Meru world is administered by one or more space servers. Each space server is the authority of what objects are in a region of the world and their positions. It is also responsible for simulation, physics, object discovery queries, and forwarding messages between objects: objects must inhabit the same world to communicate.

Meru object hosts run object code. Each object has an associated “home” space server for its current position. Objects can migrate between object hosts in order to be closer to their home space server.
What Is It?

- Software platform for 3D virtual worlds
- Princeton/Stanford collaboration

- Goal: A user should be able to interact with the whole virtual world.
- Goal: Easy application development.
What Is It?

- Software platform for 3D virtual worlds
- Princeton/Stanford collaboration

- Goal: A user should be able to interact with the whole virtual world.
- Goal: Easy application development.
Sirikata Architecture

Communication

Space
Servers
Solid Angles

distance

solid angle

\[ \text{angle} \]

\[ \text{distance} \]
The potentially interesting object (Pinto) service answers queries on what objects are nearby. Unlike virtual worlds today, which use a distance metric, Pinto uses solid angles, which define how much screen real estate an object may occupy. To quickly answer such queries, Pinto uses a modified bounding volume hierarchy data structure that stores not only bounding regions but also the largest object within them. To operate across space servers, Pinto aggregates its local object queries into a single server query, which it exports to other space servers.
Query Comparison

- For same # of objects, better quality
- For same quality, fewer objects
LBVH: Largest Bounding Volume Hierarchy

bounding volume hierarchy (BVH)

BVH geometry
PlntO Conceptual Model
Space Server

- **Location (Loc)**
  - query → objects
  - position, physics

- **Coordinate Segmentation (CSeg)**
  - object → server

- **Object Segmentation (OSeg)**
  - geometry → servers

**Potentially Interesting Objects (Pinto)**

**Forwarder**

- **OH1**
- **OH2**
- **OH3**
- **OH4**
Message from #f813 to #17c1
Obj Host 1

Obj Host 2

SS_A

SS_B

SS_C

Msg
From: #f813
To: #17c1

#d775 #f813

#17c1 #6ffa
The Forwarder routes messages between virtual world objects. In a large, distributed system, message load may exceed capacity. This means the forwarder must decide how to distribute limited throughput among competing object pairs.

The Forwarder assigns capacity using a function that models light - larger and closer objects receive more capacity than smaller and further ones, yet every object pair can communicate and none starve.

Mathematically:

\[ w_i = F(R_s, R_d) = \int \int f(p_s, p_d) \, dp_s \, dp_d \]

\[ r = d(p_s, p_d) \]

\[ f(r) = \frac{1}{(sr + \rho)^2 \cdot \log^2(sr + \rho)} \]

The functions above are how the Forwarder weights a pair \( w_i \) in terms of a pointwise function \( f(r) \). The plot shows the system enforcing these weights. Every object pair tries to communicate at a rate of approximately 8kbps (ideal). Pairs which have a falloff higher than this value can send 8kbps. Pairs with a lower falloff share the excess capacity the underutilizing pairs leave unused.
What Is It?

• Software platform for 3D virtual worlds
• Princeton/Stanford collaboration

• Goal: A user should be able to interact with the whole virtual world.
• Goal: Easy application development.
Sirikata Architecture

Communication
- Space
- Servers

Storage

Computation
- Object
- Hosts

Content Distribution Network
Sirikata Architecture

Computation

Object
Hosts
Emerson, a language for virtual worlds

Emerson has two mechanisms to simplify writing and deploying new applications. The first is sandboxing. Meru applications work very much like web ones: an entity downloads executable code to interact with an application entity, such as a chess board. Sandboxing makes it easy to do this safely.

The second mechanism is “multi-presencing.” Writing an application like a chess board normally requires complex message passing between each piece: Emerson allows a single script to control multiple virtual objects, or “presences.”
Survival Game: Zombie

Only one zombie attacks at a time
Zombie Game: Basic Actor Model

Zombie 1 script

Zombie 2 script

Zombie 3 script
Zombie Game: Basic Actor Model

Zombie 1 script
Zombie 2 script
Zombie 3 script
Zombie Game: Basic Actor Model

Lock Request

Zombie 1 script
Zombie 2 script
Zombie 3 script
Zombie Game: Basic Actor Model

Zombie 1
script

Zombie 2
script

Zombie 3
script

Lock Granted
Zombie Game: Basic Actor Model

Zombie 1 script

Zombie 2 script

Zombie 3 script

Zombie Attack!
Zombie Game: Multi-Presence

Zombies Script
Zombie Game: Multi-Presence

Zombies Script
Zombie A attack!
Sirikata Architecture

Communication

Space Servers

Computation

Object Hosts

Storage

Content Distribution Network
Sirikata Architecture

Storage

Content Distribution Network

42
Sirikata CDN

- [http://open3dhub.com](http://open3dhub.com)
  ‣ [meerkat:///jterrace/Seagull.dae/optimized/Seagull.dae](meerkat:///jterrace/Seagull.dae/optimized/Seagull.dae)
- Authentication through OAuth
- Wide-area replication through Cassandra
- DONAR for mapping to nearest replica
- Caching to reduce latency
Making It Easy

Model upload

Sirikata CDN
Making It Easy

Model upload

arrow

clean

Sirikata CDN
Making It Easy

Model upload → clean → original → Sirikata CDN
Making It Easy

Model upload

clean

original

optimized

Sirikata CDN
Making It Easy

Model upload → clean → original → optimized → progressive → Sirikata CDN

base mesh → stream
Content Pipeline

- Check COLLADA model is valid
- Asynchronously upload optimized model
  - Reduces materials, atlases textures, generates triangles
  - Average draw calls reduced from 13 to 5
- Asynchronously upload progressive model
  - Base mesh at low detail
  - Download progressive stream to increase detail
  - Allows fine-grained client control of system resources
Content Pipeline

• Check COLLADA model is valid
• Asynchronously upload optimized model
  ‣ Reduces materials,atlases textures, generates triangles
  ‣ Average draw calls reduced from 13 to 5
• Asynchronously upload progressive model
  ‣ Base mesh at low detail
  ‣ Download progressive stream to increase detail
  ‣ Allows fine-grained client control of system resources
Sirikata Architecture

Communication

Space Servers

Computation

Object Hosts

Storage

Content Distribution Network
Questions
Coordinate Segmentation

- Distributed kd-tree
- Hierarchically divides space into regions
- Tree “cut” into two levels
- Replicated, highly stable upper tree
- Much more dynamic lower trees managed by CSEG servers
- Load balancing done at CSEG servers based on number of avatars
Evaluation

- kd-tree is “split-axis”
- Not perfect, but still pretty good...
- Graph characterizes a kd-tree over world population from 1990-2015
  - Depth remains bounded by 41; deeper than a perfect tree by 50%
  - Average depth remains about the same
Upper Tree Stability

- With cut at a suitable depth, upper tree remains very stable
Performance

- Currently, latency for a query is around 2.5 ms
- Most of it is network latency
- With four CSEG servers, currently sustains ~5000 queries per second
Object Segmentation

- The space server forwarder must know which space server is authoritative for the recipient.
- OSeg maintains presence to space server mappings:

<table>
<thead>
<tr>
<th>Presence ID</th>
<th>Server connected to ID</th>
<th>Presence radius</th>
</tr>
</thead>
</table>

- Each record is ~50B: heavy caching is feasible.
- Stale entries are easy to handle (extra hop).
Object Segmentation (OSeg)

<table>
<thead>
<tr>
<th>Presence ID</th>
<th>Server connected to ID</th>
<th>Presence radius</th>
</tr>
</thead>
</table>

**Space Server A**

- Forwarder
- OSeg Cache

**Space Server B**

- Forwarder
- OSeg Cache

**OSeg Service**

(Authoritative records)
Object Segmentation (OSeg)

Presence ID  Server connected to ID  Presence radius

Space Server A
Forwarder
OSeg Cache

Space Server B
Forwarder
OSeg Cache

Msg
From: #f813
To: #17c1

OSeg Service
(Authoritative records)
Object Segmentation (OSeg)

<table>
<thead>
<tr>
<th>Presence ID</th>
<th>Server connected to ID</th>
<th>Presence radius</th>
</tr>
</thead>
</table>

**Space Server A**
- Forwarder
- OSeg Cache

**Space Server B**
- Forwarder
- OSeg Cache

Msg
From: #f813
To: #17c1

OSeg Service (Authoritative records)
Object Segmentation (OSeg)

<table>
<thead>
<tr>
<th>Presence ID</th>
<th>Server connected to ID</th>
<th>Presence radius</th>
</tr>
</thead>
</table>

Space Server A
- Forwarder
- OSeg Cache

Space Server B
- Forwarder
- OSeg Cache

OSeg Service (Authoritative records)