
Random-Accessible Compressed Triangle Meshes

Sung-eui Yoon

Korea Advanced Institute of Sci. and Tech. (KAIST)

Peter Lindstrom

Lawrence Livermore National Laboratory

KAIST



Challenges

- **Massive models**
 - Takes high disk/memory spaces
 - Long data access time and low I/O performance



Boeing 777 model (470M triangles)

Excerpted from SIGGRAPH tutorial



**ST. Matthew model
(372M triangles)**

Goal

- Provide a compressed mesh representation of a massive model for various applications
 - Supports random access
 - Provides various mesh access including connectivity (adjacency and incidence) queries
 - Possibly, improves the performance of applications



Existing Mesh Representations

- Uncompressed indexed file format (e.g., binary ply or obj files)
 - Provides *random access* for various applications
 - But, does not support connectivity (adjacency and incidence) queries
 - Requires large data space (e.g., 1.8GB for 100M triangles)
 - Have low I/O performance



External Mesh Representations

- Provides random and *connectivity queries*
 - [Silva et al. 02, Cignoni et al. 03, Isenburg and Gumhold 03]
- But, takes more disk space (e.g., 3.2GB for 100M triangles) and higher data access time



Cache-Coherent Mesh Layouts

- Store a mesh in a layout that closely matches runtime access patterns of applications [Yoon et al. 05 and 06]
 - Reduces the number of cache misses during random access
 - Achieves high I/O performance
 - Does not decrease nor increase data size



Mesh Compression

- Apply compression techniques to reduce the data size [Alliez and Gotsman 05]
- Mainly focus on efficient transmission or archival of data
 - *Limited to sequential access*
- Impose one particular data layout maximizing the compression ratio
 - Can lead to poor I/O performance



Compression Methods Supporting Random Access

- Quite common in video & audio encoding
 - E.g., MPEG video
- [Choe et al. 04, Kim et al. 06]
 - Support random access
 - Target rendering applications

Do not support general mesh access queries



Main Contributions

- Propose novel layout-preserving compression method and runtime data access framework
 - Serves as an compact out-of-core data access framework
 - Provides random access and various mesh access queries
 - Achieves high compression ratio and high I/O performance for massive models



Outline

- Overview
- Compression at preprocessing
- Runtime data access framework
- Results



Outline

- **Overview**
- Compression at preprocessing
- Runtime data access framework
- Results



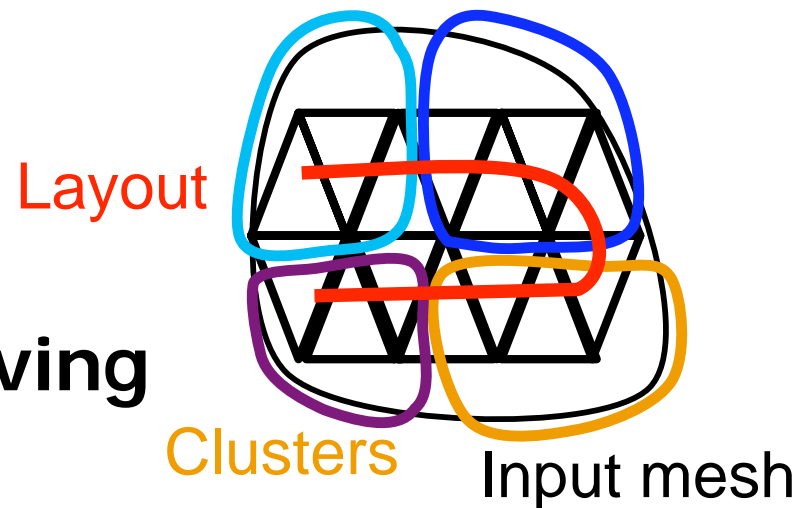
Overview – Compression at Preprocessing

- Decompose the mesh into spatially coherent clusters

- Efficiently performed by decomposing an original layout

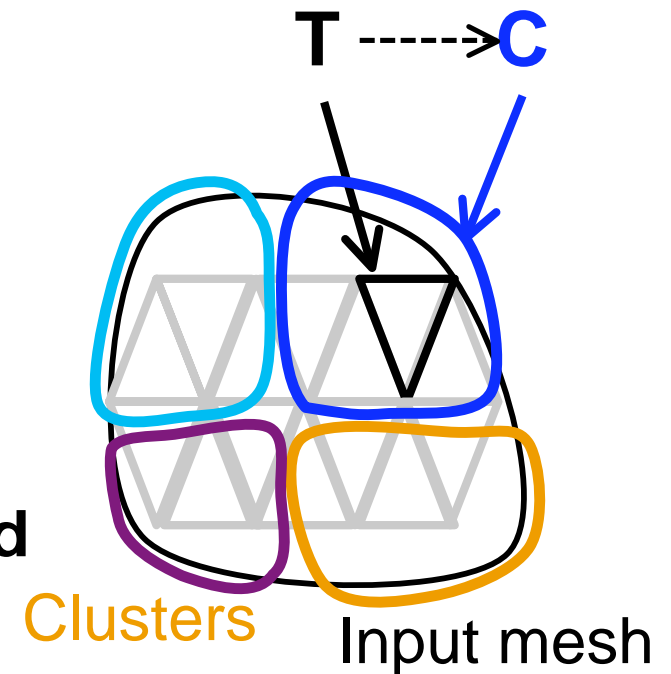
- Compress each cluster separately while preserving the mesh layout

- Each cluster serves as an access point



Overview – Runtime Data Access Framework

- Decompress the cluster containing a data required by an application
- Dynamically construct connectivity information
 - Provide correct connectivity information for the requested data even with partially loaded data



Outline

- Overview
- **Compression at preprocessing**
- Runtime data access framework
- Results

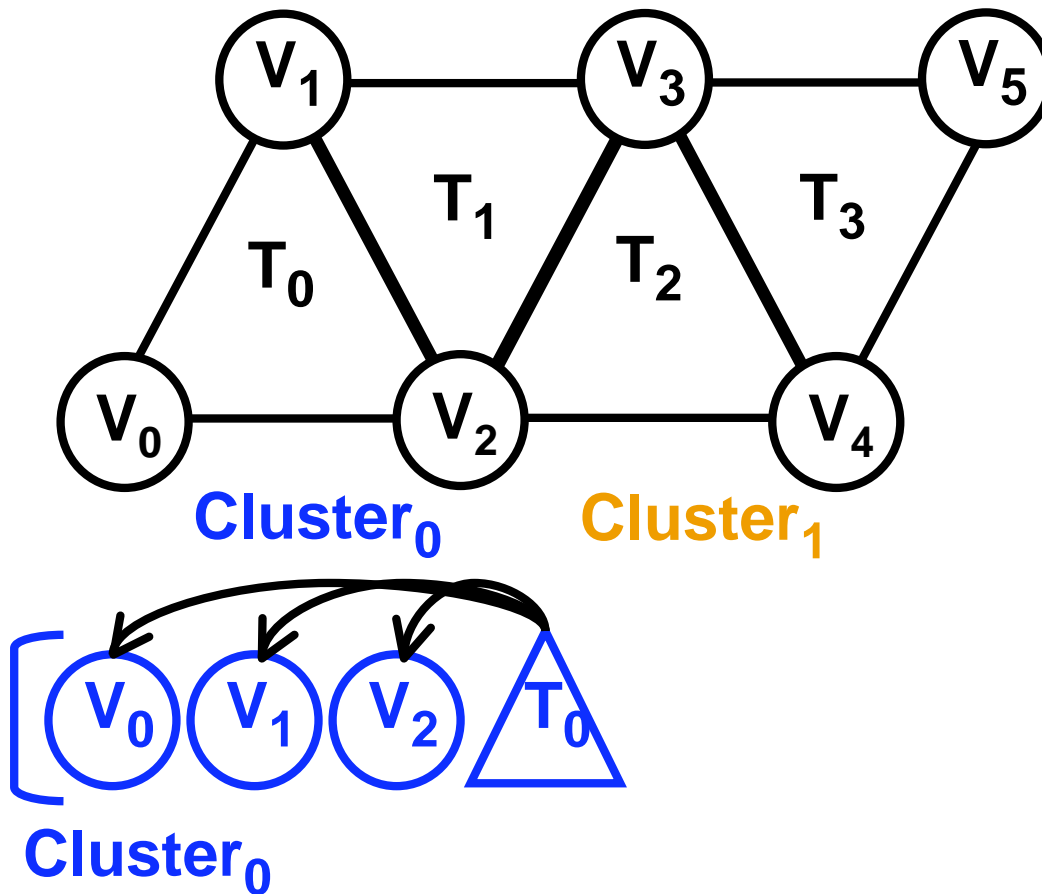


Layout-Preserving Cluster-Based Compression

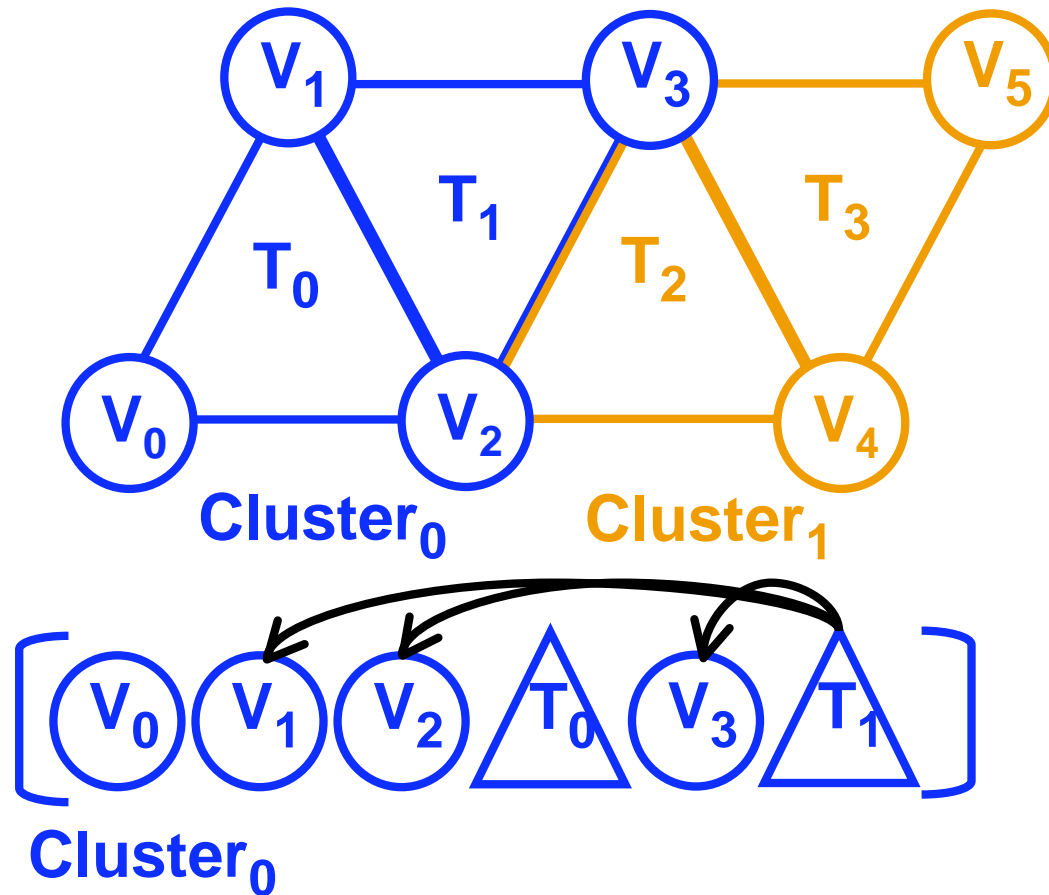
- Built on top of streaming mesh compression [Isenburg et al. 05]
- Cluster has:
 - Fixed number of consecutive triangles in the layout (e.g., 4k triangles)
 - Vertices introduced by those triangles
- Our method encodes an triangle layout of the mesh
 - Also, encodes cluster information (e.g., neighboring clusters)



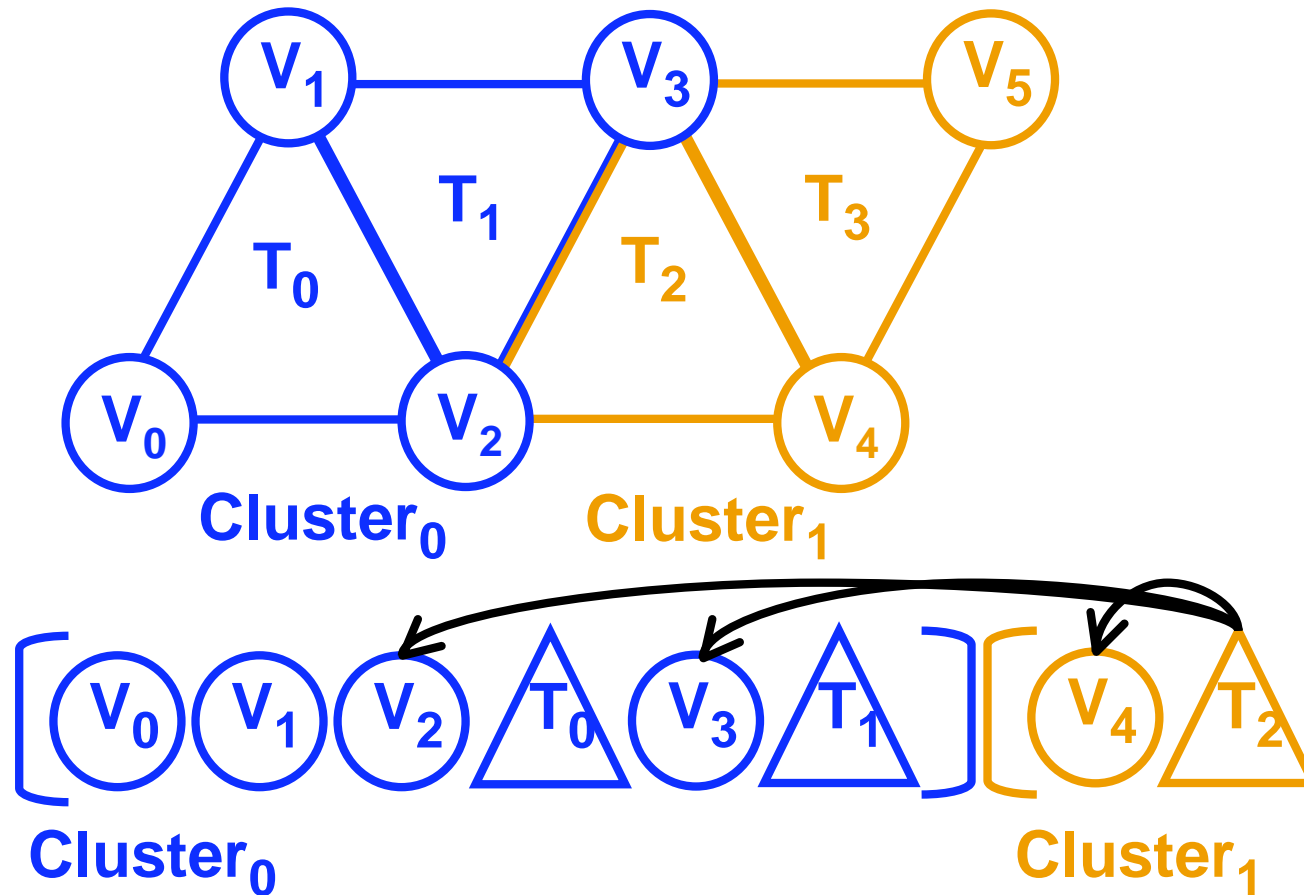
Layout-Preserving Cluster-Based Compression



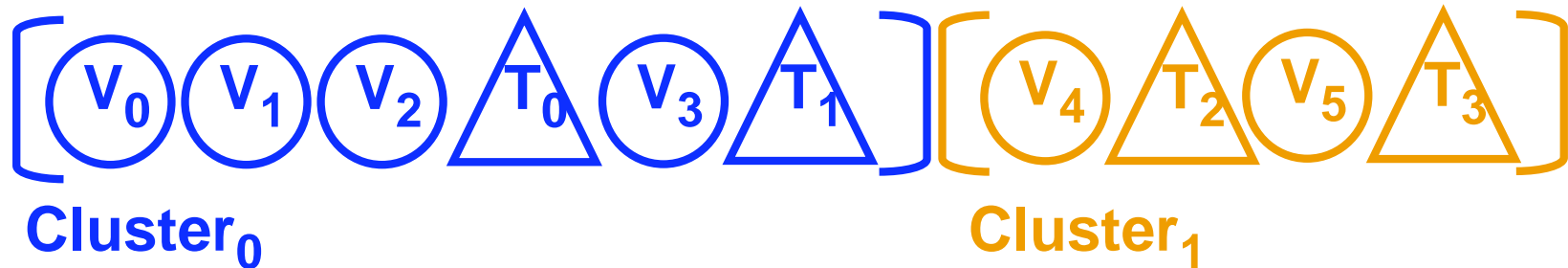
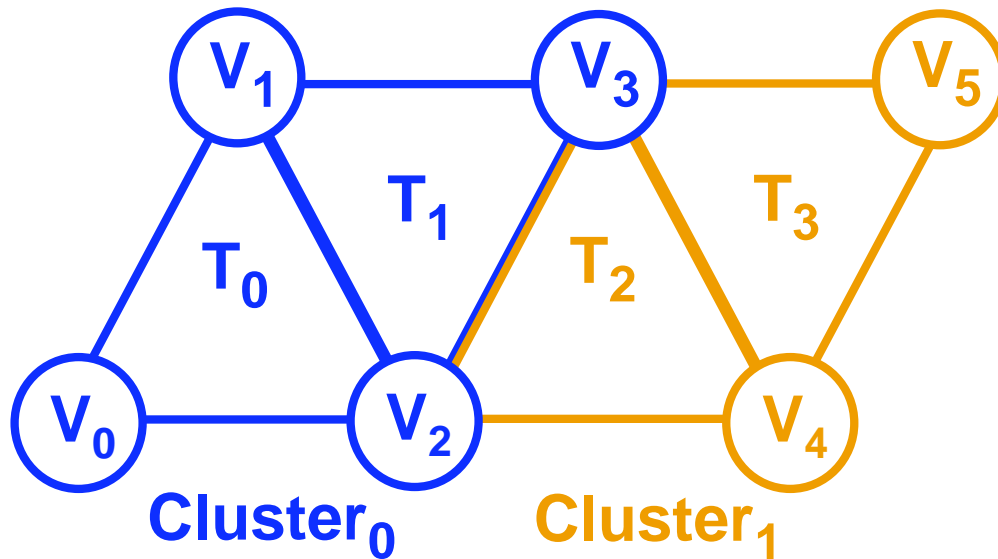
Layout-Preserving Cluster-Based Compression



Layout-Preserving Cluster-Based Compression



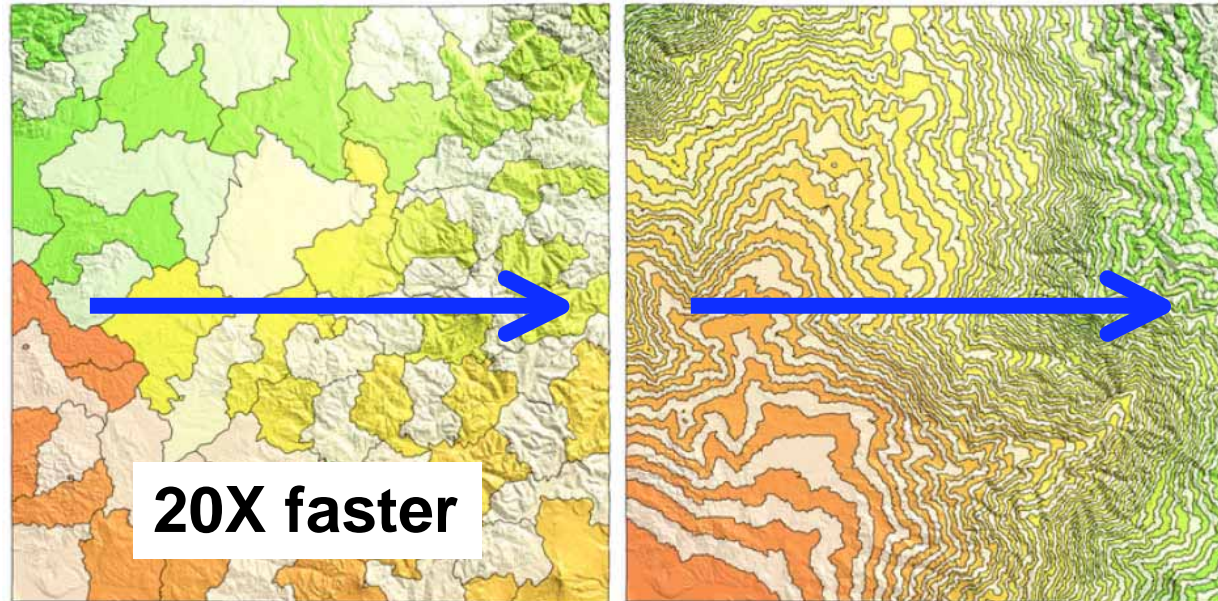
Layout-Preserving Cluster-Based Compression



Original Mesh Layout

- Defines clusters
 - Layout should provide spatially coherent clusters

Runtime
access
pattern



Cache-oblivious
mesh layout

Breadth-first
layout

[Yoon et al. 05,
Yoon and Lindstrom 06

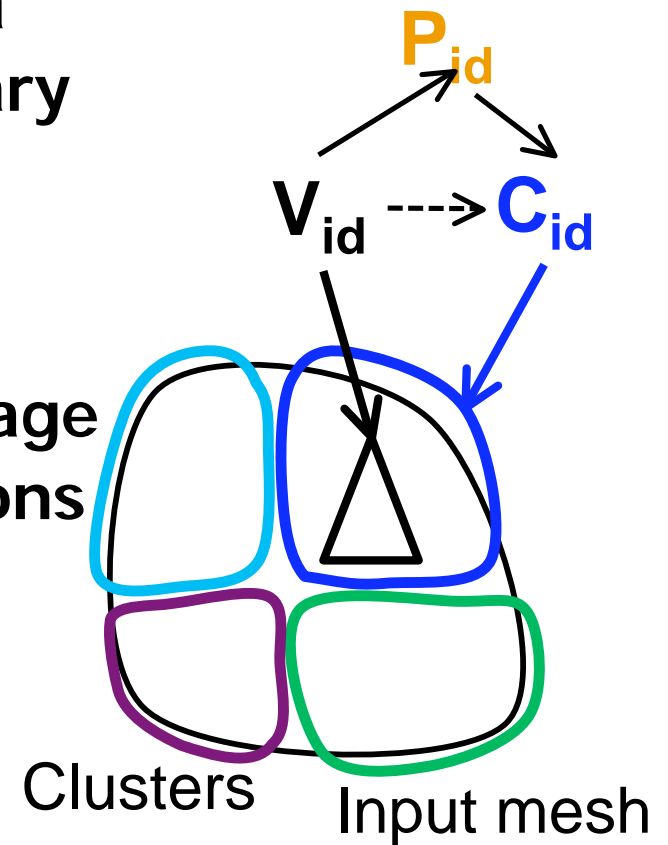
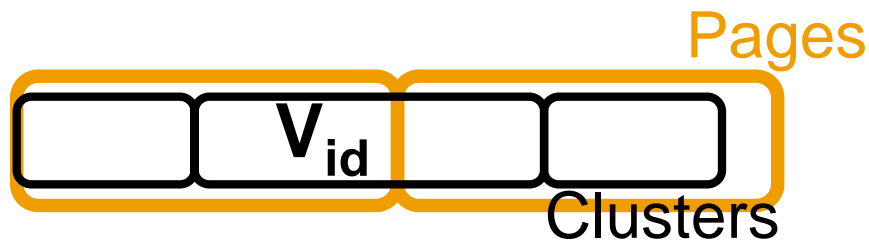
Outline

- Overview
- Compression at preprocessing
- **Runtime mesh access framework**
- Results



Runtime Mesh Access Framework

- First identify a cluster containing the requested data
 - Slow since clusters have arbitrary vertex elements
- Pages
 - Have power-of-two elements & clusters overlapping with the page
 - Requires only a few bit operations to find the cluster

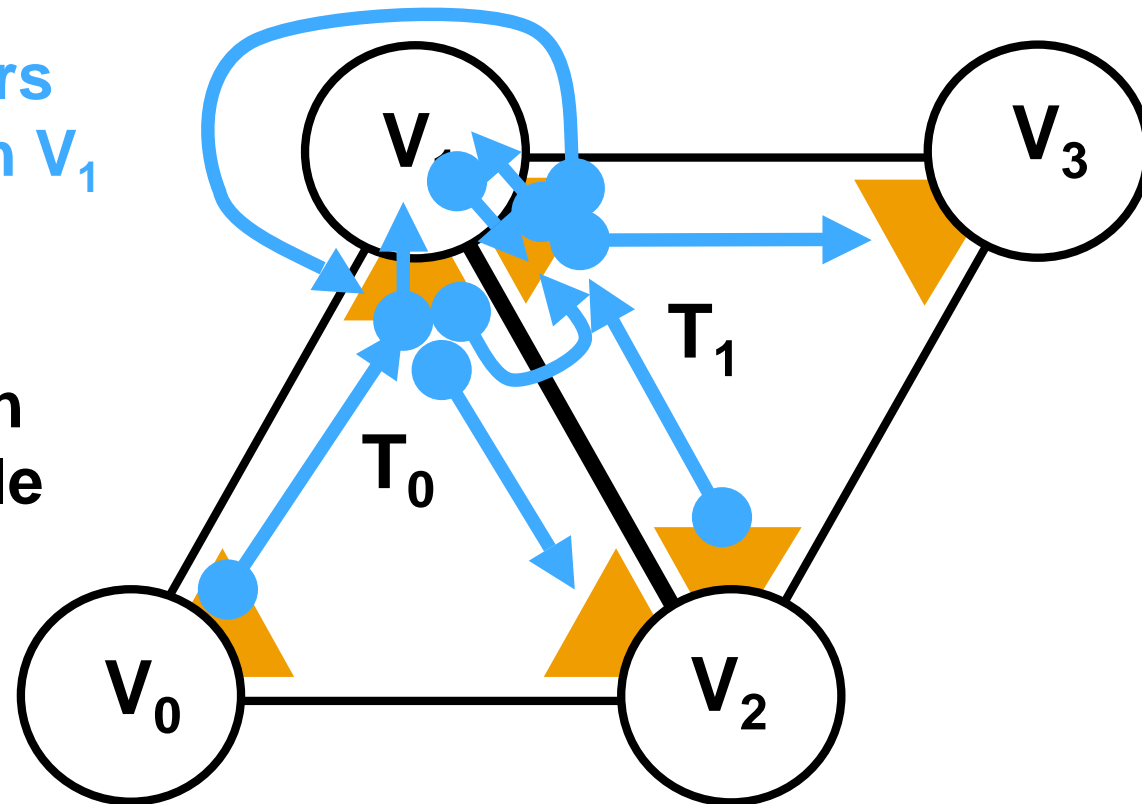


In-Core Mesh Representation

- Similar to the corner tables [Rossignac 01, Joy et al. 03]
 - Supports connectivity queries

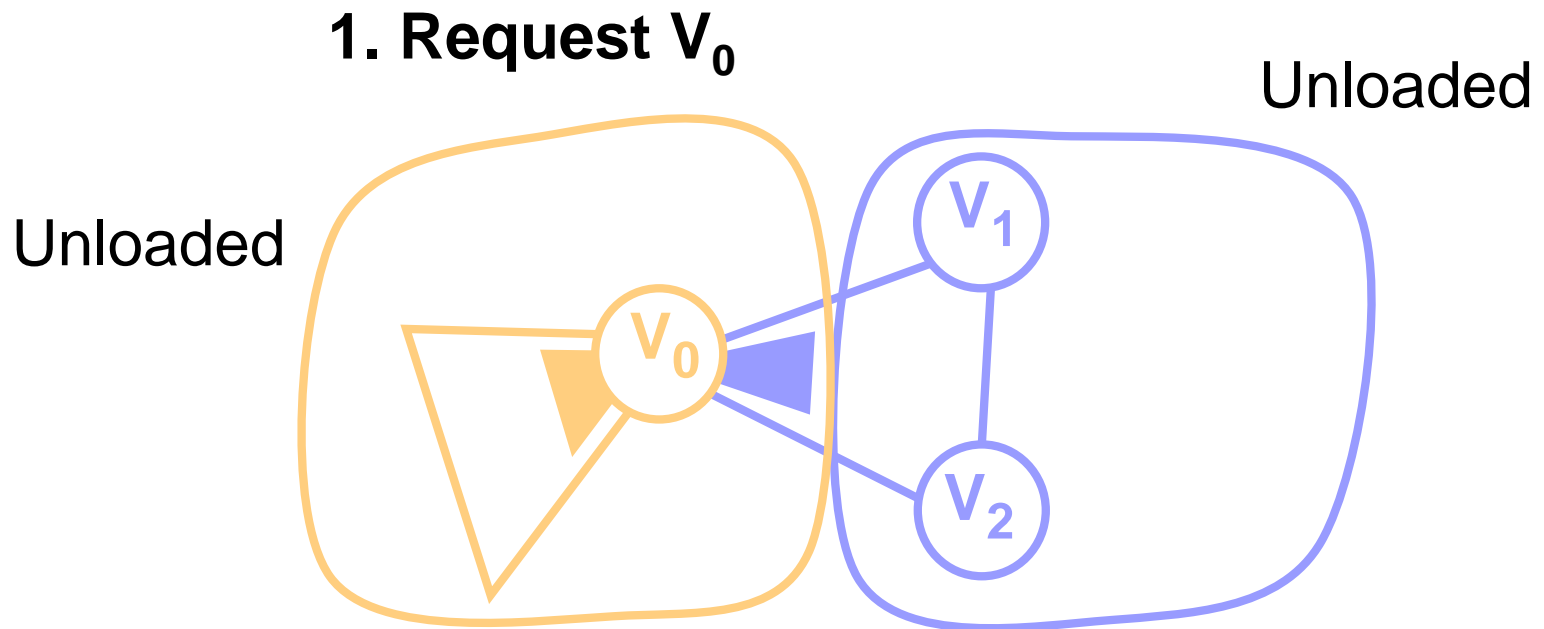
All the pointers
associated with V_1

Compactly
represented in
the corner table



On-Demand Connectivity Construction

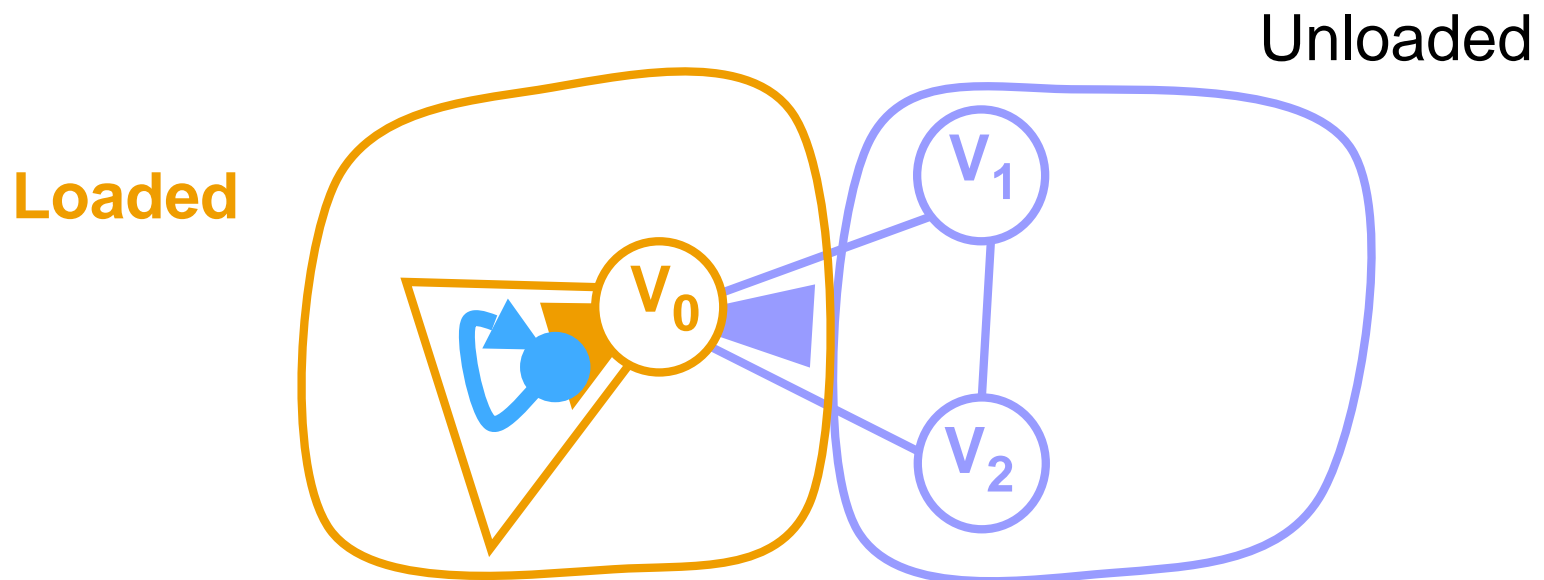
- Dynamically construct the corner information as we load clusters



On-Demand Connectivity Construction

- Dynamically construct the corner information as we load clusters

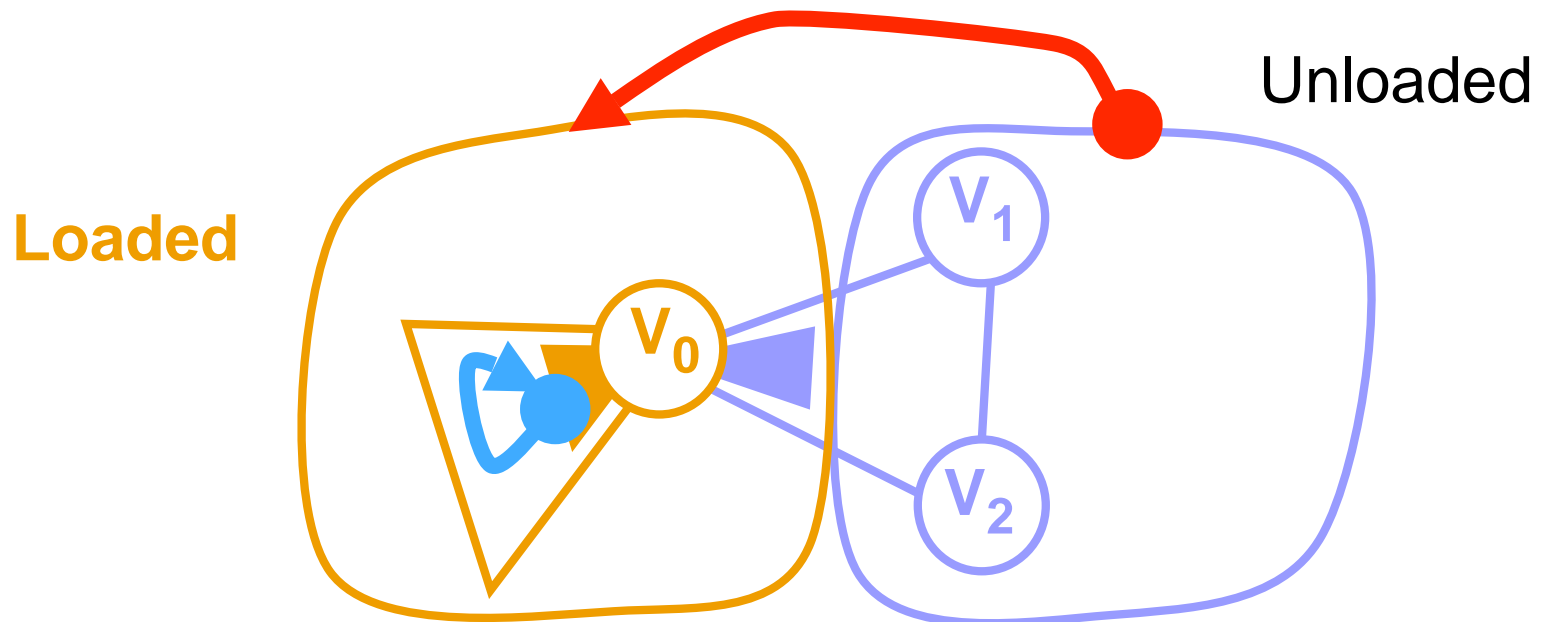
2. Load the cluster and build connectivity



On-Demand Connectivity Construction

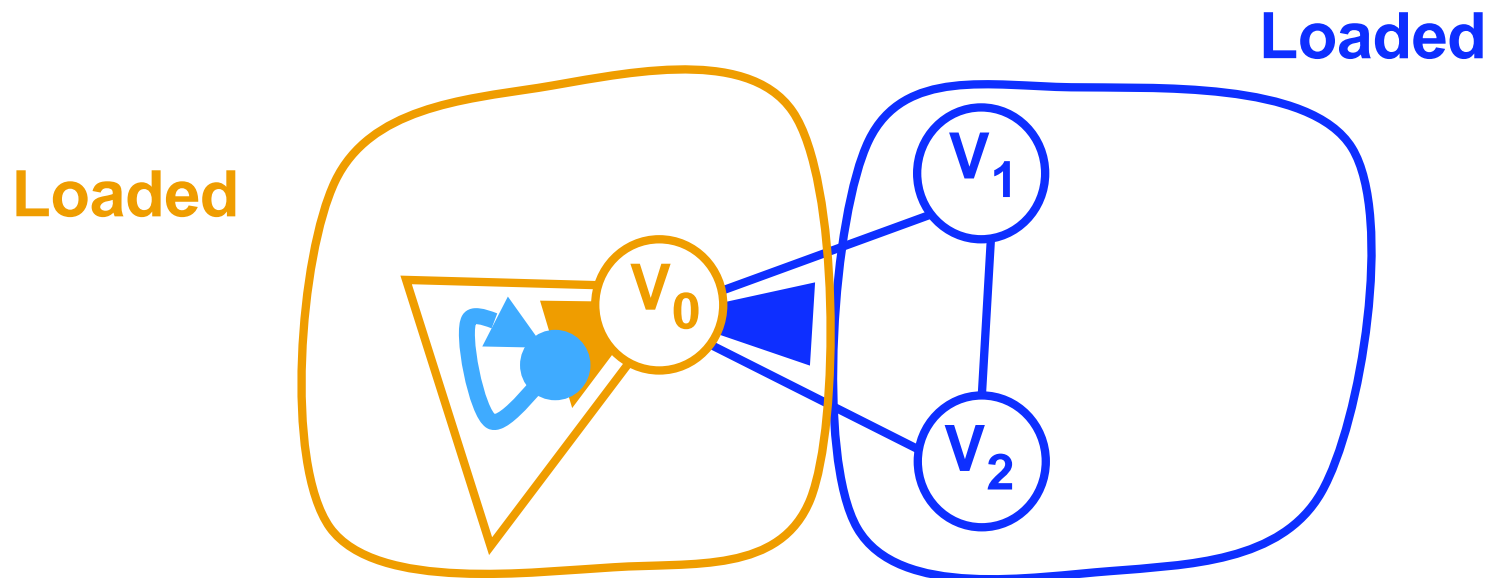
- Dynamically construct the corner information as we load clusters

3. Load clusters referencing the cluster



On-Demand Connectivity Construction

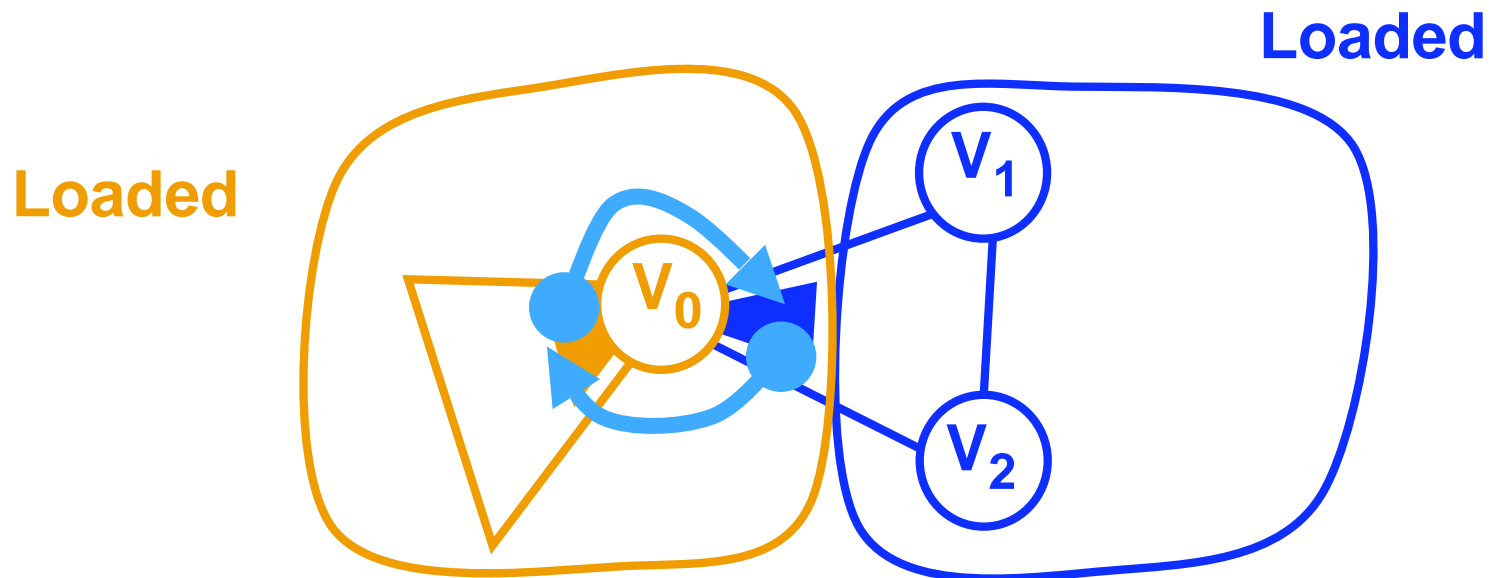
- Dynamically construct the corner information as we load clusters



On-Demand Connectivity Construction

- Construct the corner information as we load clusters

4. Provide correct connectivity information with partially loaded data



Outline

- Overview
- Compression at preprocessing
- Runtime data access framework
- **Results**



Compression Results

	Compression, bits per vertex (bpv) (= Geom + Conn)	Compression ratio
Puget Sound (134M tri.)	21.4 (13.5, 7.9)	21:1
RMI-Isosurface (102M tri.)	27.6 (19.4, 8.2)	16:1
St. Matthew (128M tri.)	22.9 (14.8, 8.1)	20:1

Used 16 bits quantization



Comparison Results

- Streaming compressor [Isenbourg et al. 05]
 - Ours has 15% overhead due to encoding cluster-related information
- [Touma and Gotsman 98]
 - Ours has 50% overhead

Runtime Benchmarks

- Iso-contouring using seed sets
 - Accesses sub-sets of the mesh
- Mesh re-ordering
 - Accesses the whole mesh

Require mesh traversal in a random order



Performance in Iso-Contouring

- Puget sound, 134M tri
 - Uses the cache-oblivious mesh layout
 - 178MB for the compressed mesh (21:1 compression ratio)
 - 2.4 overall runtime performance improvement over using the uncompressed mesh

Extracted iso-contour



Mesh Re-Ordering

- Re-compute the layout optimized for a specific runtime application (e.g, streaming computation)
- Re-order vertices and triangles of the mesh
 - Achieve up to 6.4 times performance improvement over using un-compressed meshes



Performance Improvement

- Mainly due to higher I/O performance
 - Reduce disk I/O time by using compression
 - Use CPU power to efficiently decompress the data



Conclusions

- Novel layout-preserving compression and runtime data access framework
 - Provide random access and connectivity queries without decompressing the whole mesh
 - **Source codes are available as [OpenRACM](#)**
 - Achieve high I/O performance and, thus, able to observe up to 6:1 performance improvement

Acknowledgements

- Martin Isenburg
- Anonymous reviewers
- Members of KAIST theory of computation lab. and LLNL data analysis

- Funding sources
 - Lawrence Livermore National Lab.
 - KAIST seed grant



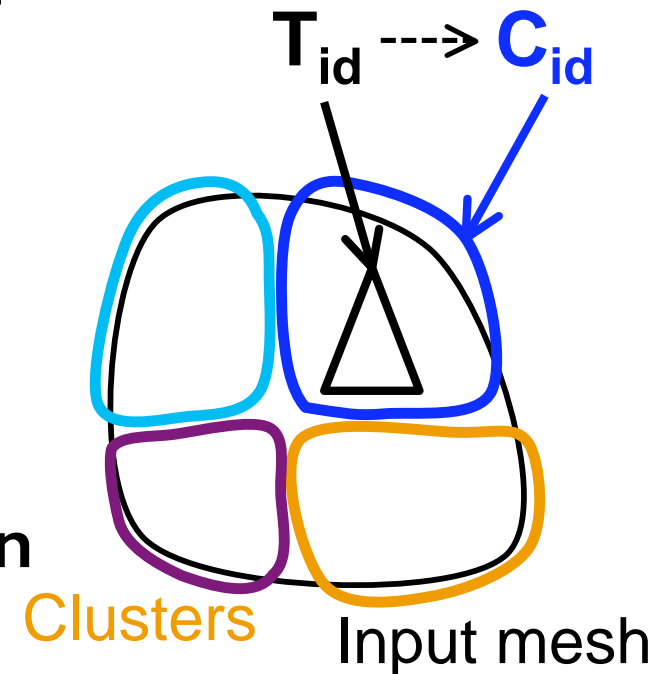
Any Questions?

● Thank you!



Benefits

- Provide random access and connectivity queries
 - Without decompressing the whole mesh
 - High I/O performance
- Preserve a cache-coherent layout
 - Achieve high cache utilization



Encoding Vertex Indices

- Provides three layers of compression contexts
 - 1) three vertices of a previous triangle
 - 2) active vertices in the current cluster
 - 3) vertices in the neighboring clusters: two indices (cluster index, local index)



Mesh API

- GetVertex (...)
- GetTriangle (...)
- GetCorner (...)
- GetNextVertexCorner (...)
- GetNextTriangleCorner (...), etc

Build more complex mesh access functions from the API



Ongoing and Future Work

- Efficiently handle modifications to the mesh
- Further optimize for parallel data access
- Extend it to hierarchies and apply to ray tracing and collision detection

