Maximizing Parallelism in the Construction of BVHs, Octrees, and *k*-d Trees

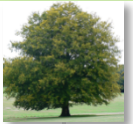
Tero Karras

NVIDIA Research

HIGH-PERFORMANCE GRAPHICS
PARIS, FRANCE JUNE 25-27, 2012

Trees

Trees



Better

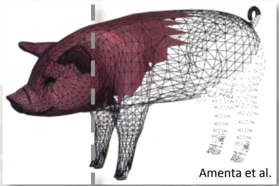
Faster

Pharr & Humphreys

Path tracing

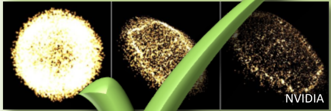
Photon mapping

Real-time ray tracing



Surface reconstruction

Collision detection



Particle simulation

Crassin et al.

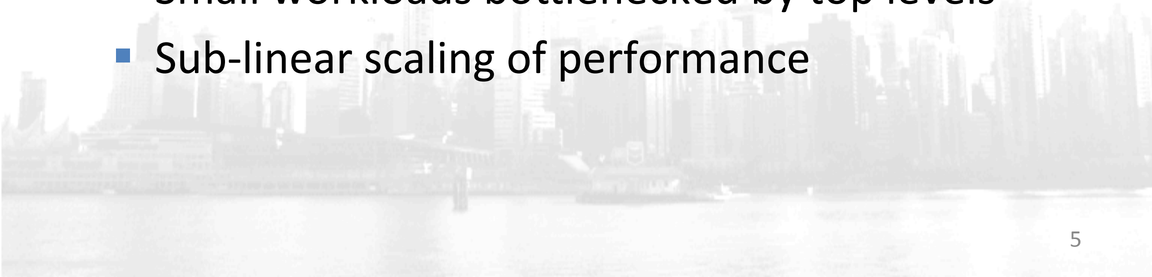
Voxel-based global illumination

Outline

- Fastest existing methods are sequential
  - Parallelize within each hierarchy level
  - But not between levels

Outline

- Fastest existing methods are sequential
  - Parallelize within each hierarchy level
  - But not between levels
- Lack of parallelism
  - Small workloads bottlenecked by top levels
  - Sub-linear scaling of performance



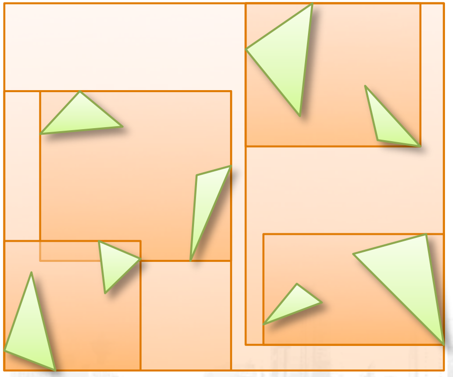
Outline

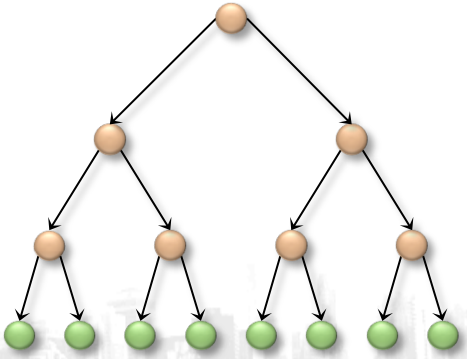
- Novel way to build the entire tree in parallel
  - Two algorithmic "building blocks"
  - Fast, scalable

Outline

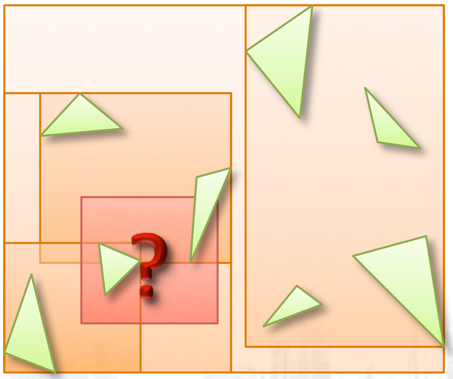
- Novel way to build the entire tree in parallel
  - Two algorithmic "building blocks"
  - Fast, scalable
- Main focus: BVHs
  - Point-based octrees and *k*-d trees also covered in the paper

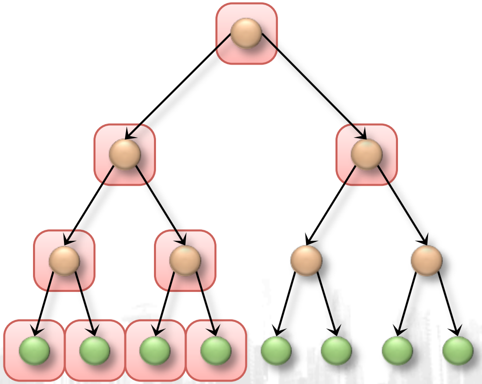
Bounding volume hierarchy





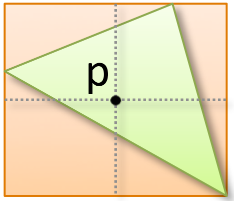
Bounding volume hierarchy





LBVH - Lauterbach et al. [2009]

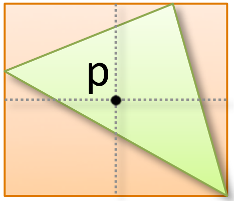
1. Assign Morton codes
2. Sort primitives
3. Generate hierarchy
4. Fit bounding boxes



$p_x = 0.1010$
$p_y = 0.0111$
$p_z = 0.1100$

LBVH - Lauterbach et al. [2009]

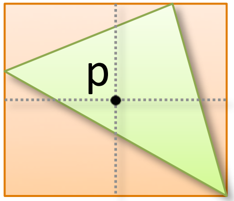
1. Assign Morton codes
2. Sort primitives
3. Generate hierarchy
4. Fit bounding boxes



p_x = 0.1 0 1 0 0 1 0
p_y = 0.0 1 0 1 1 1 1
p_z = 0.1 1 0 0 1 0 0

LBVH - Lauterbach et al. [2009]

1. Assign Morton codes
2. Sort primitives
3. Generate hierarchy
4. Fit bounding boxes

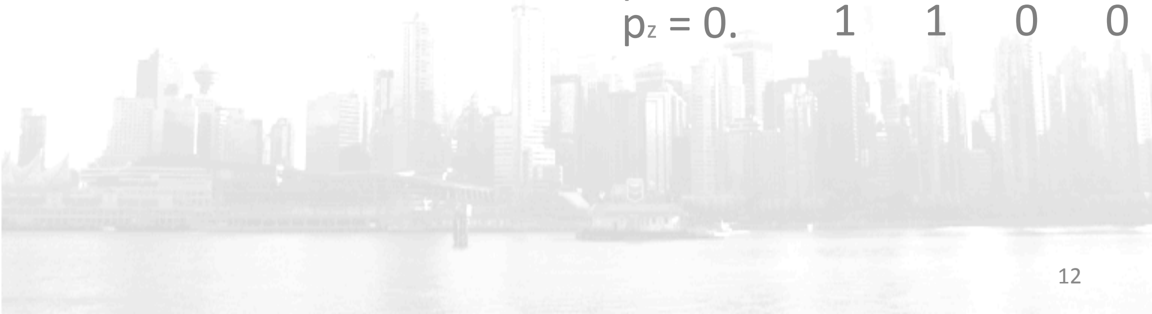


$p_x = 0.\ 1\ \ 0\ \ 1\ \ 0$

$p_y = 0.$

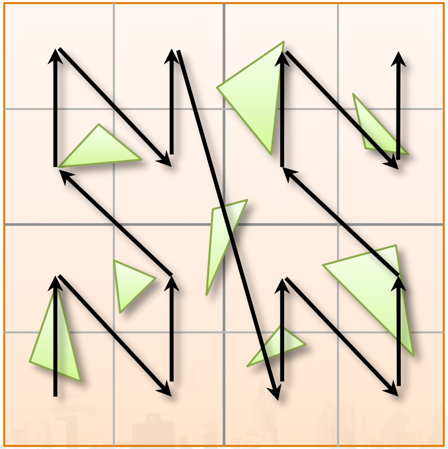
code = 1 0 1 0 1 1 1 1 0 0 1 0

$p_z = 0.\ \ 1\ \ 1\ \ 0\ \ 0$



LBVH - Lauterbach et al. [2009]

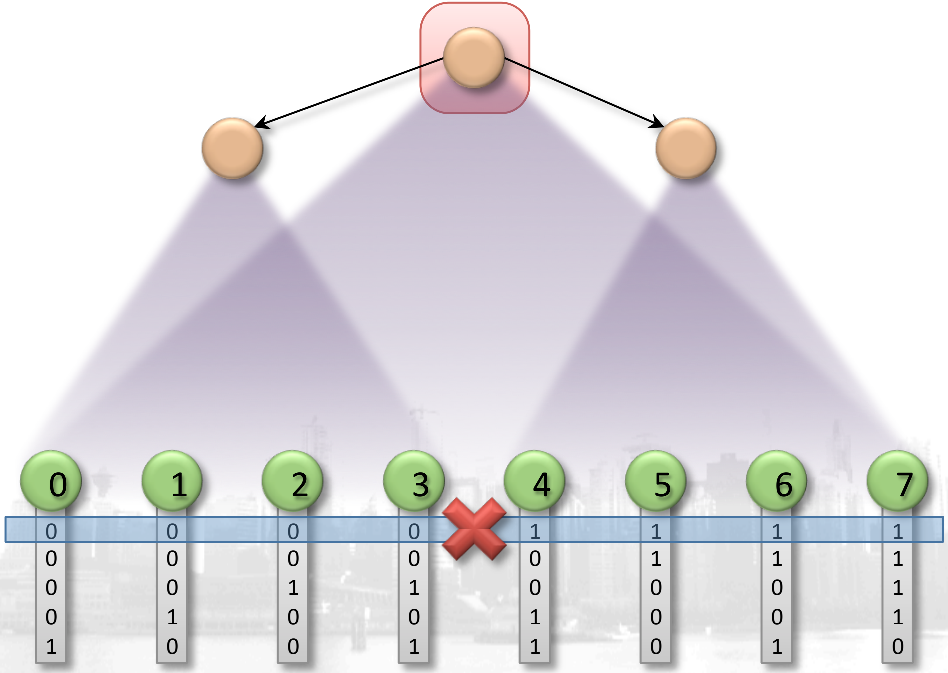
1. Assign Morton codes
2. Sort primitives
3. Generate hierarchy
4. Fit bounding boxes



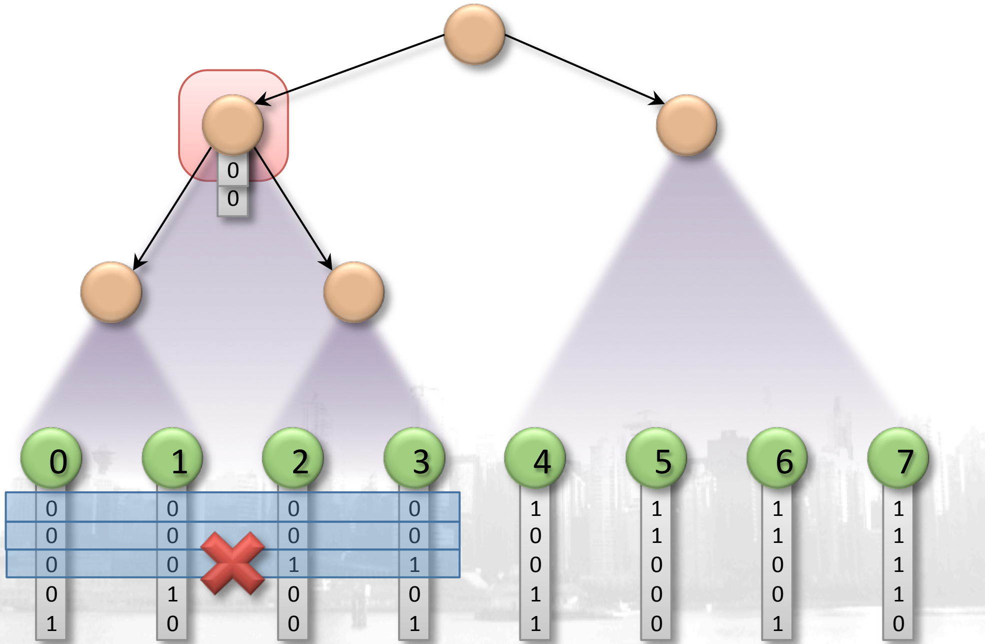
LBVH - Lauterbach et al. [2009]

1. Assign Morton codes
2. Sort primitives
3. Generate hierarchy
4. Fit bounding boxes

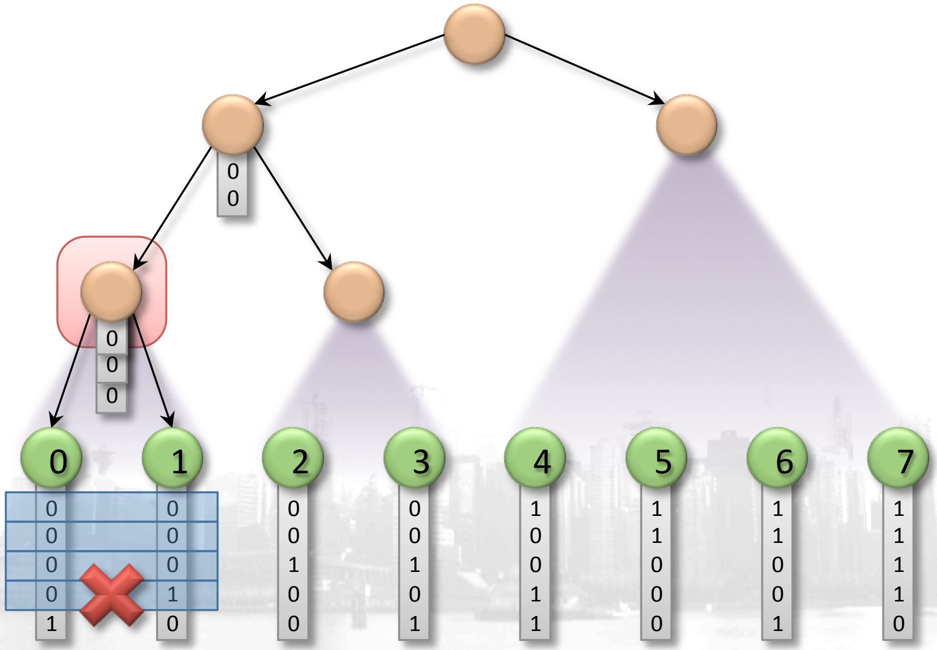
Binary radix tree



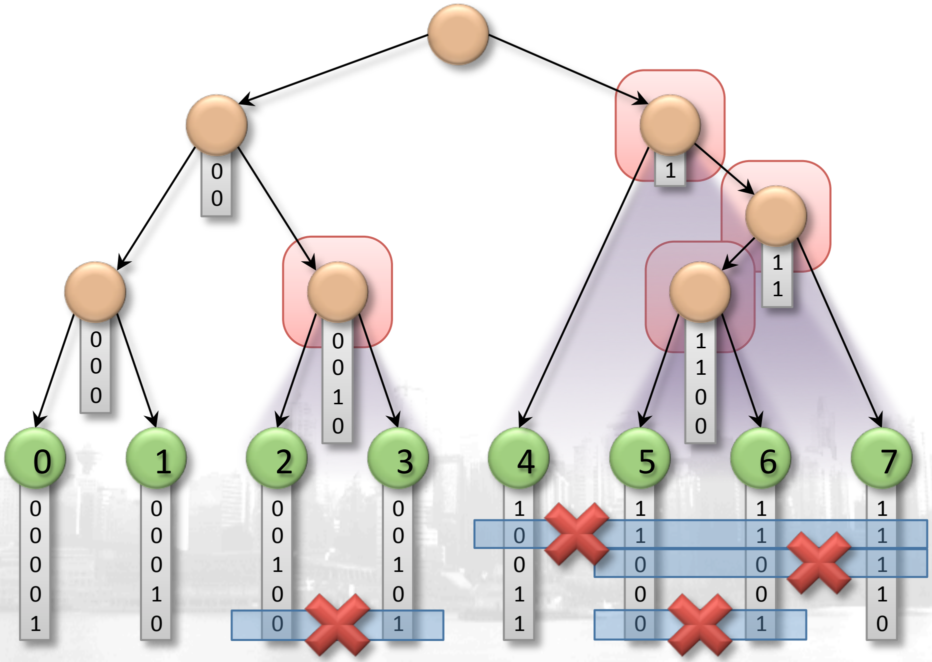
Binary radix tree



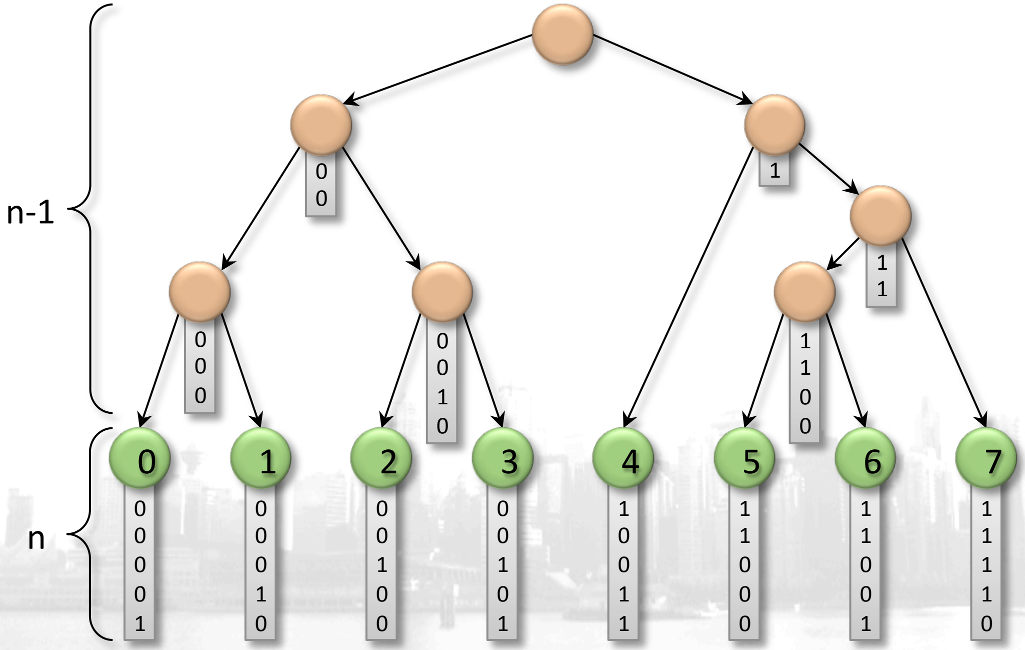
Binary radix tree



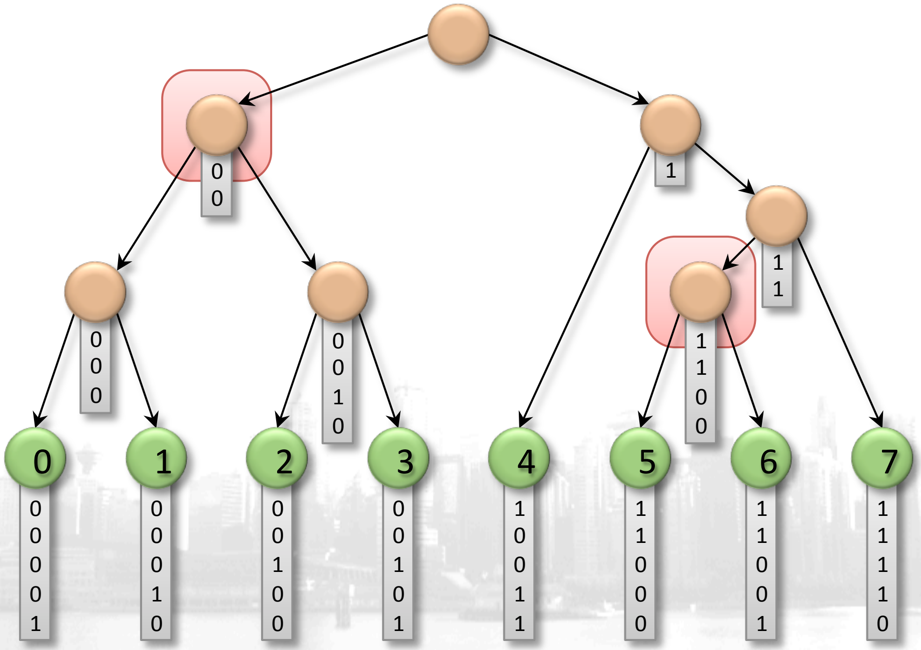
Binary radix tree



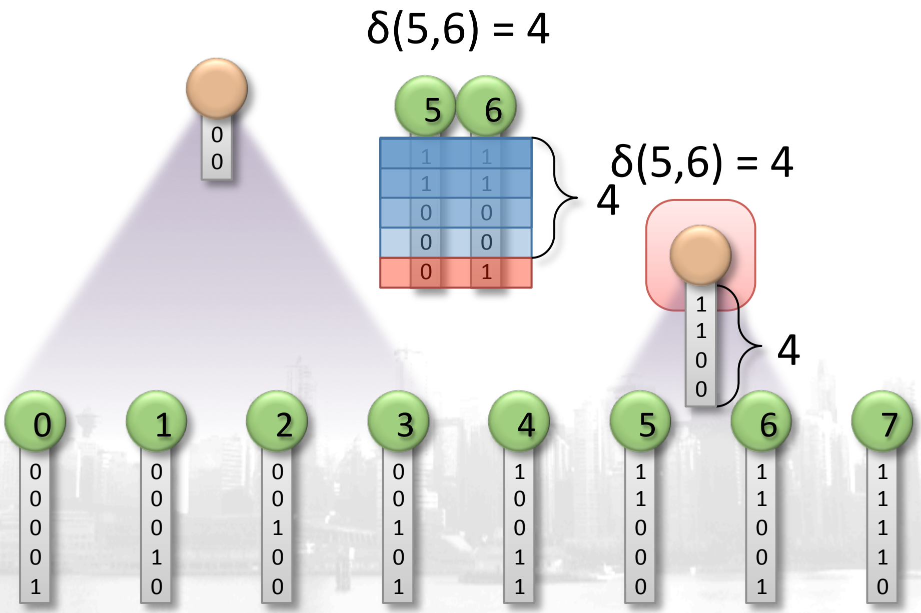
Binary radix tree



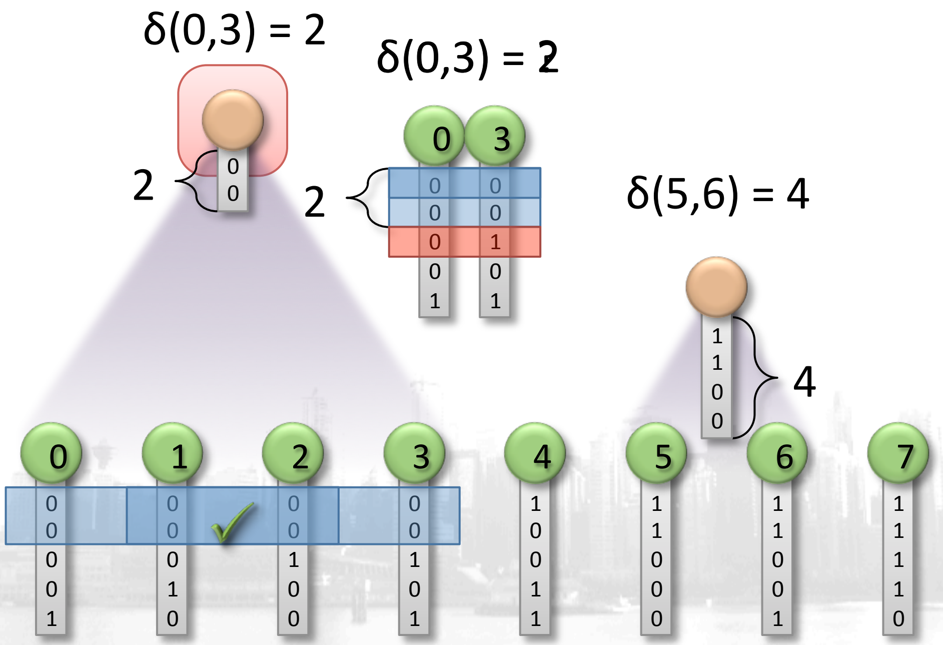
Longest common prefix



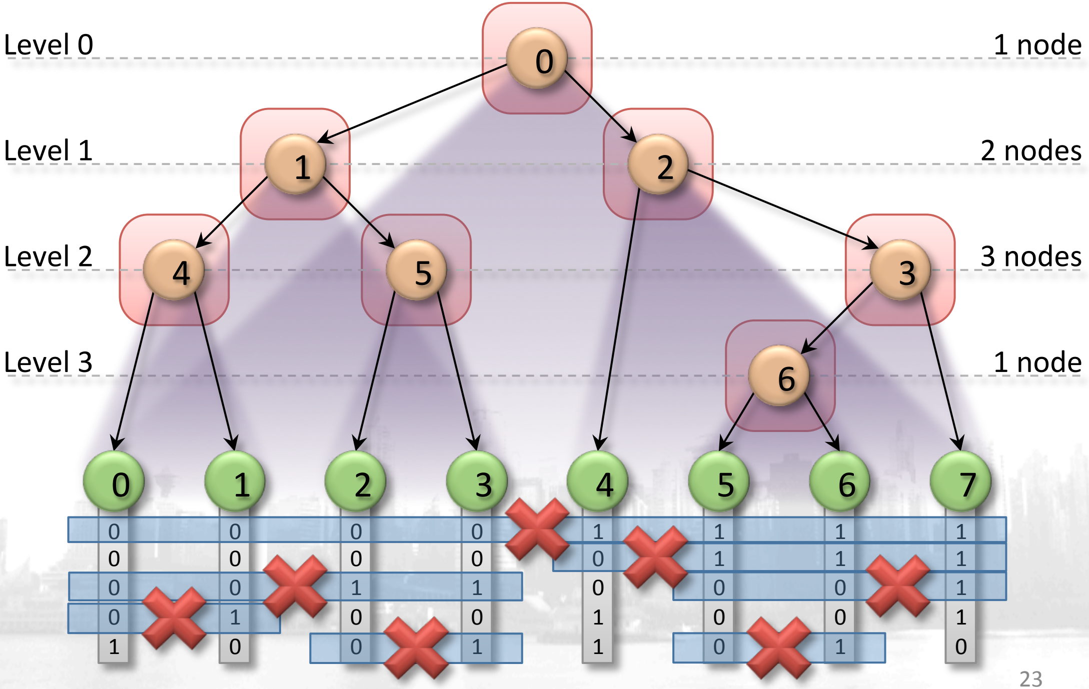
Longest common prefix



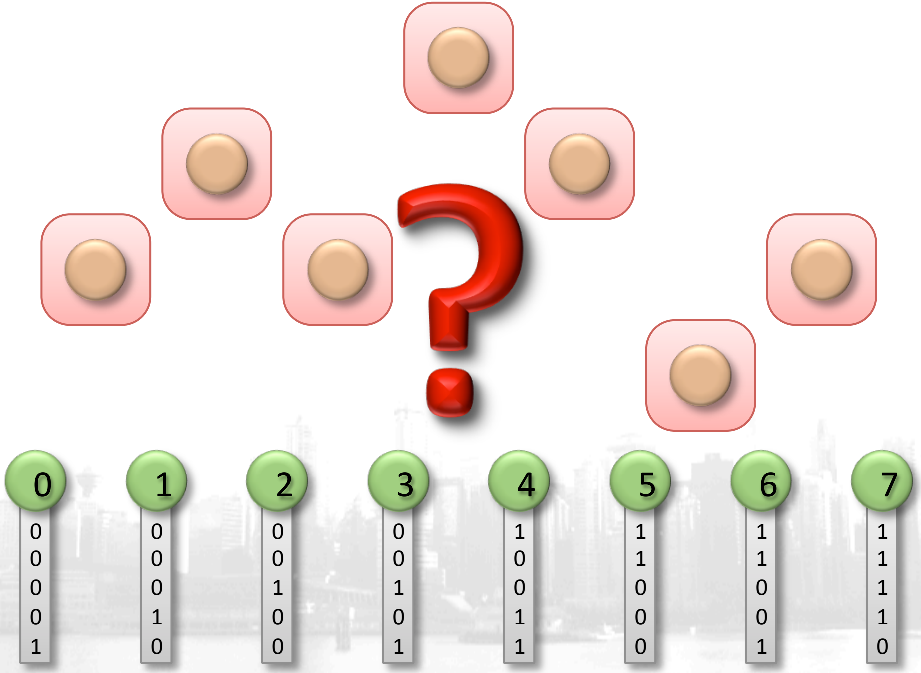
Longest common prefix



Garanzha et al. [2011]



Our method



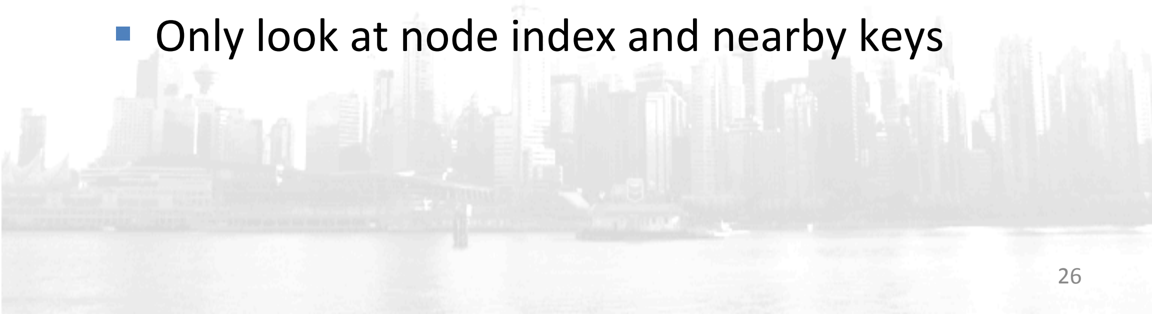
Our method

- Define a numbering scheme for the nodes
  - Gain some knowledge of their identity
  - Establish a connection with the keys

Our method

- Define a numbering scheme for the nodes
  - Gain some knowledge of their identity
  - Establish a connection with the keys

- Find the children of a given node
  - Only look at node index and nearby keys

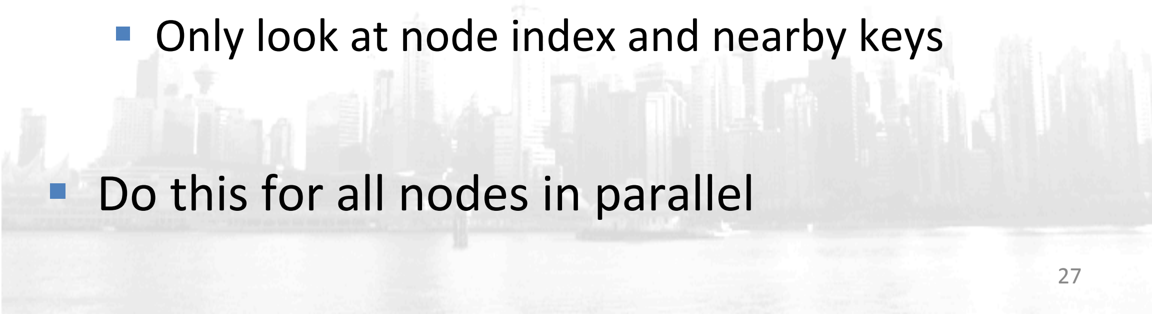


Our method

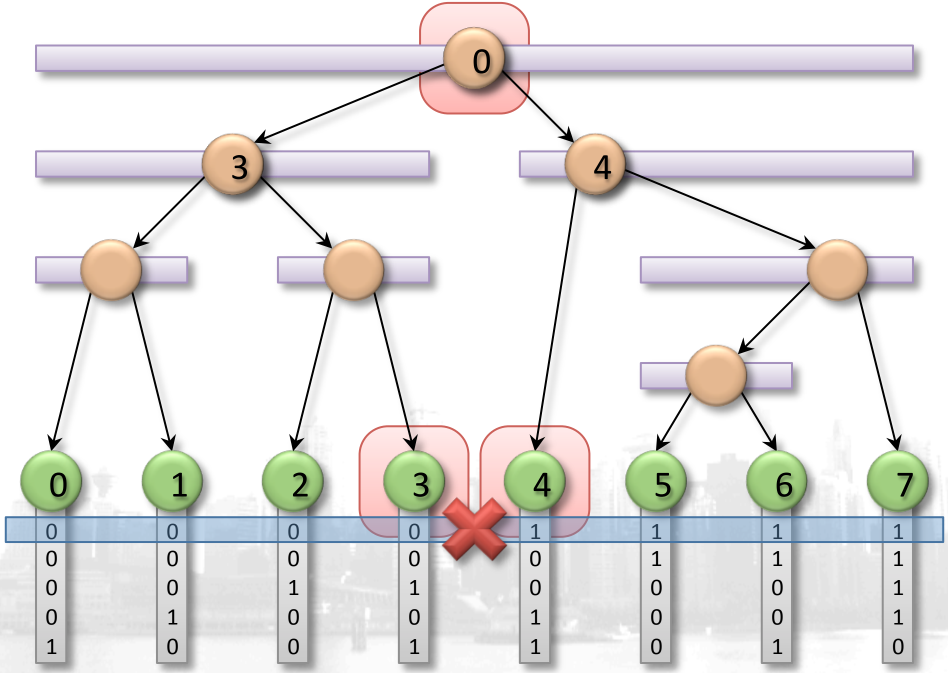
- Define a numbering scheme for the nodes
  - Gain some knowledge of their identity
  - Establish a connection with the keys

- Find the children of a given node
  - Only look at node index and nearby keys

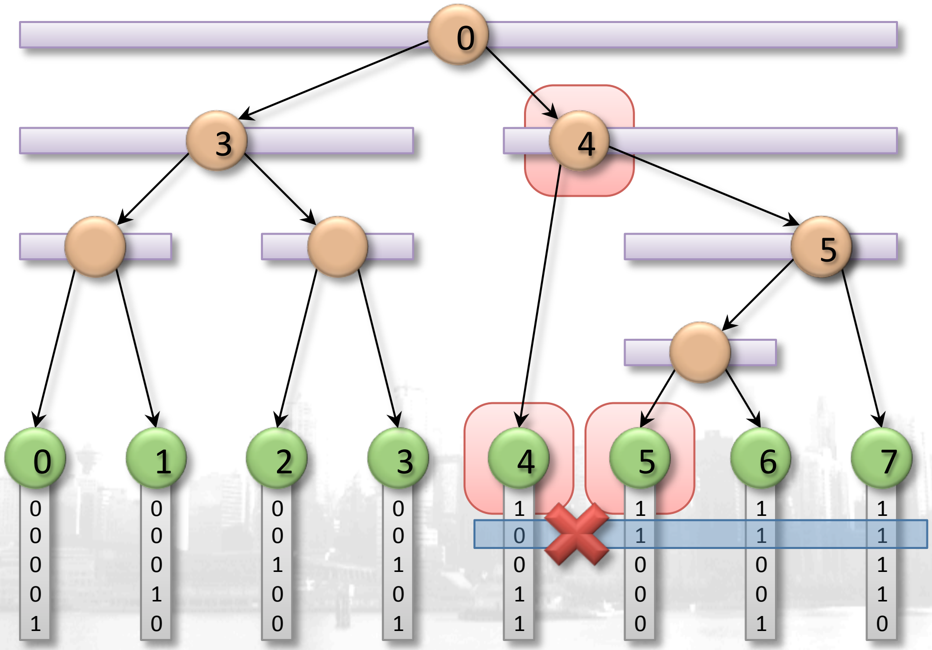
- Do this for all nodes in parallel



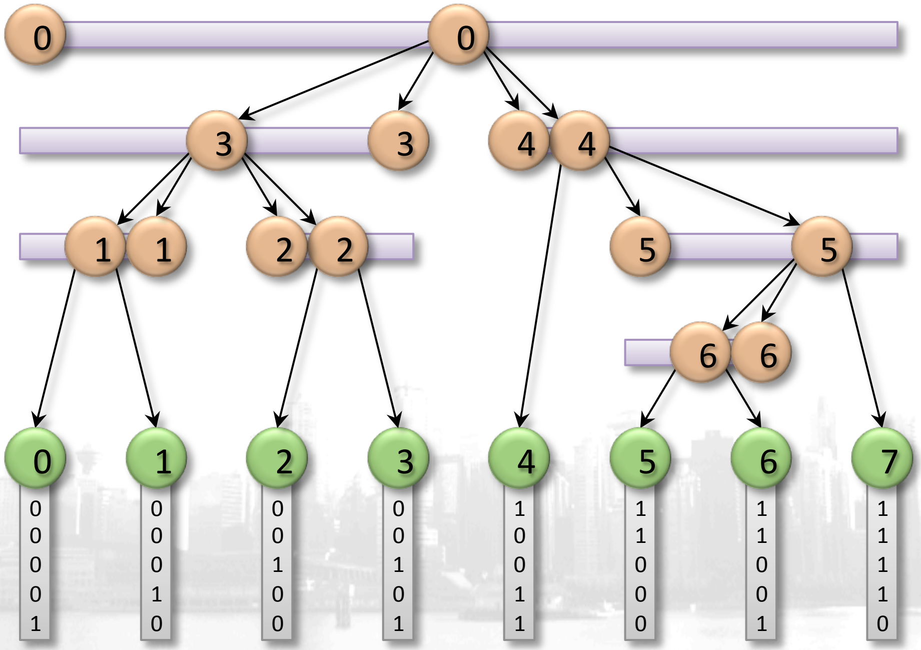
Numbering scheme



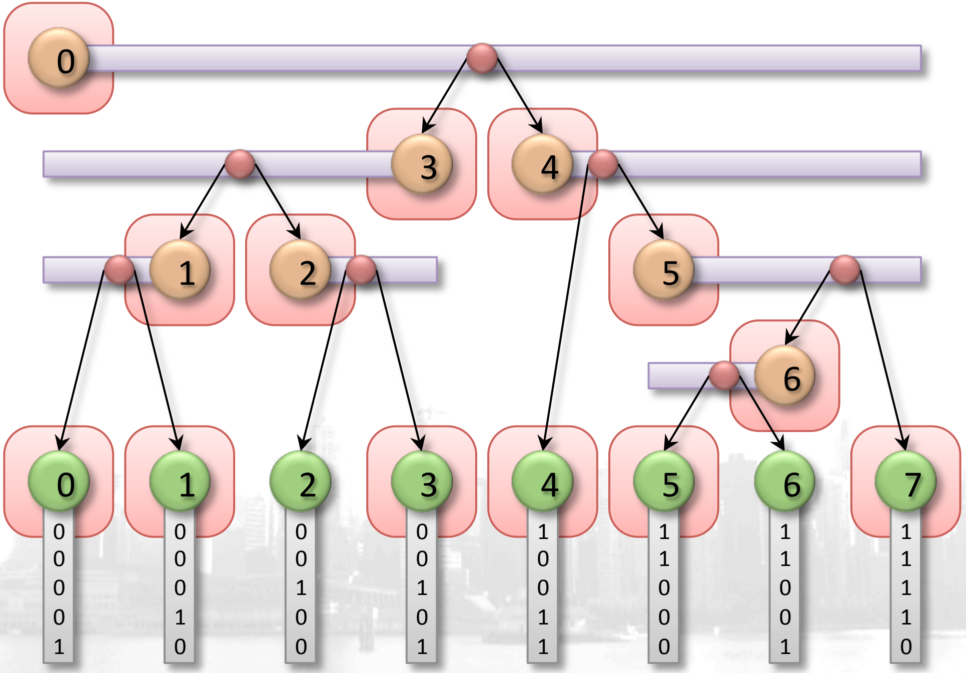
Numbering scheme



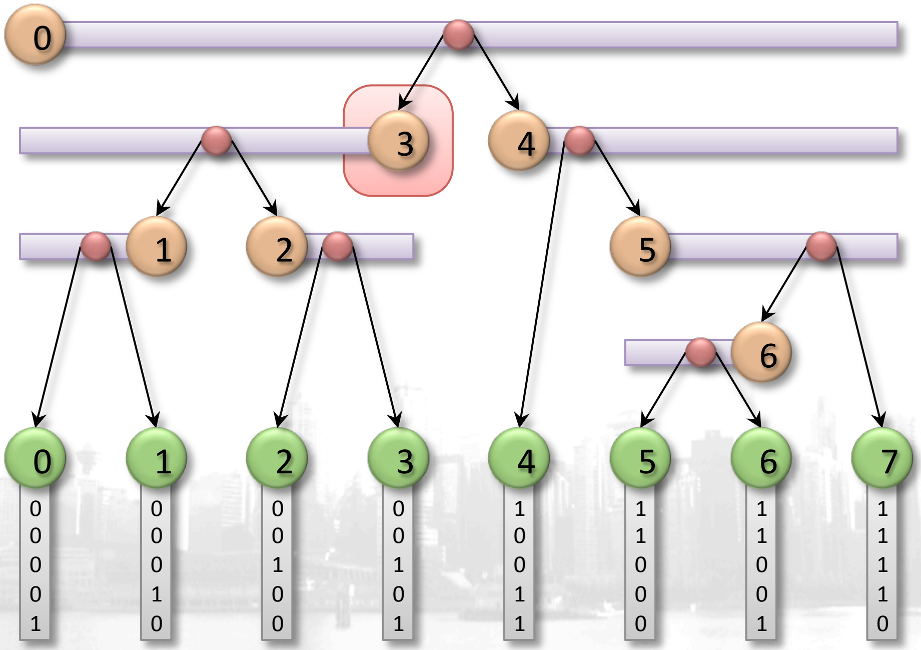
Numbering scheme



Numbering scheme

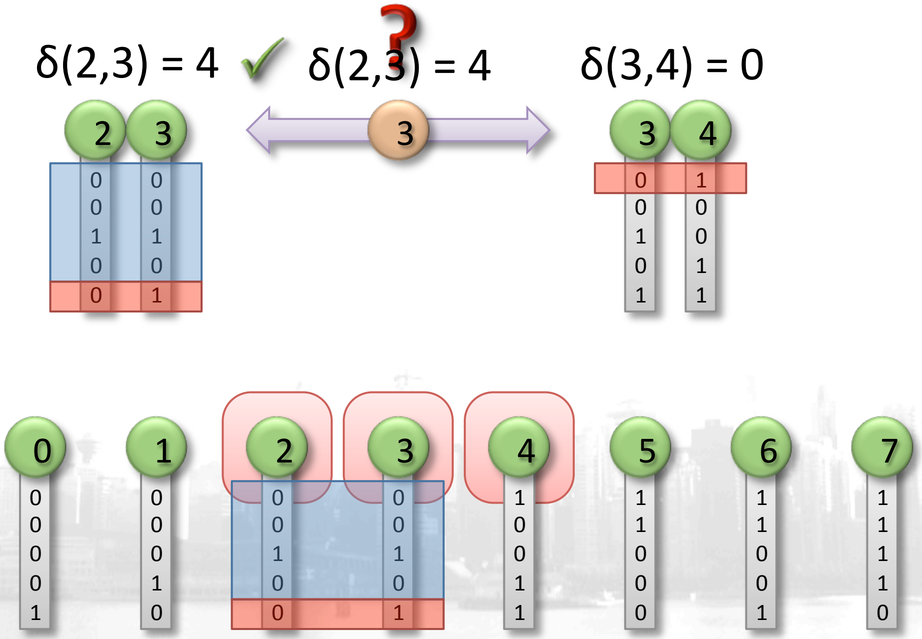


Algorithm



0
3
4
1
2
5
6
0
1
2
3
4
5
6
7
00001
00010
00100
00101
10011
11000
11001
11110

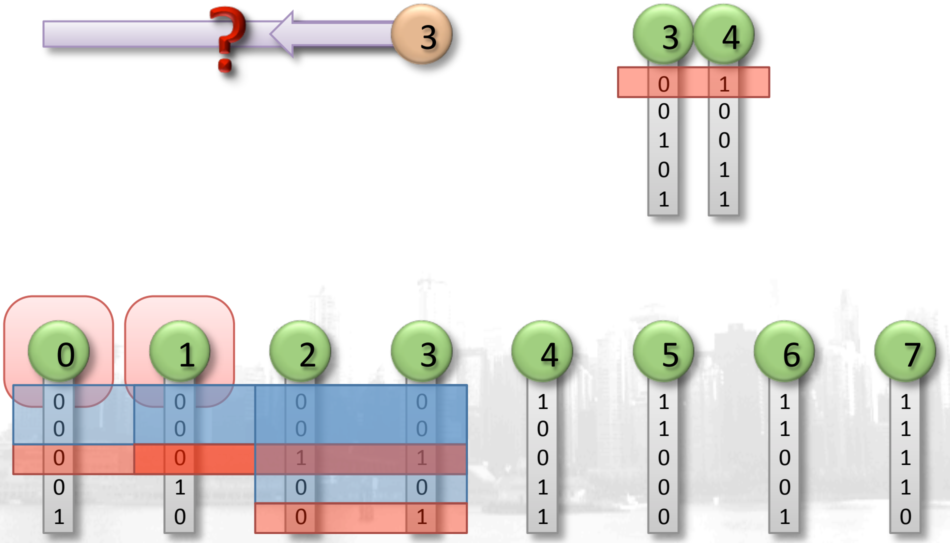
Algorithm



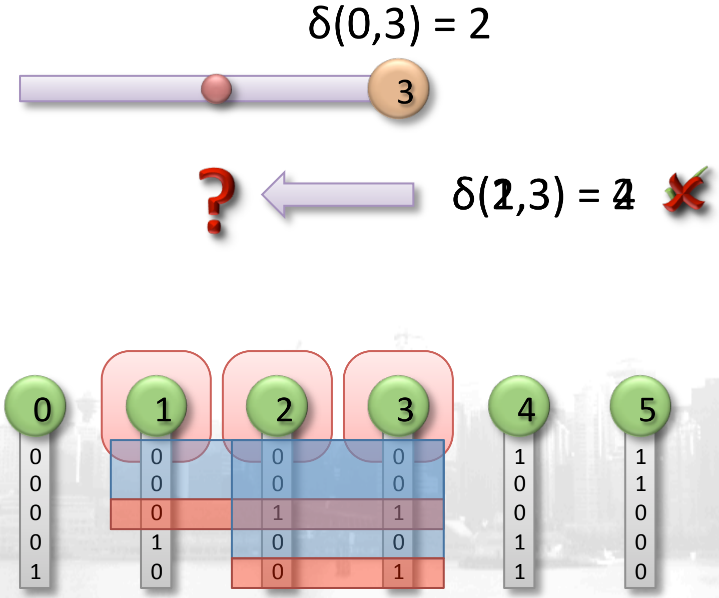
δ(2,3) = 4
δ(2,3) = 4
δ(3,4) = 0
2
3
0 0 1 0 0
0 0 1 0 1
3
3
4
0 0 1 0 1
1 0 0 1 1
0
1
2
3
4
5
6
7
0 0 0 0 1
0 0 0 1 0
0 0 1 0 0
0 0 1 0 1
1 0 0 1 1
1 1 0 0 0
1 1 0 0 1
1 1 1 1 0

Algorithm

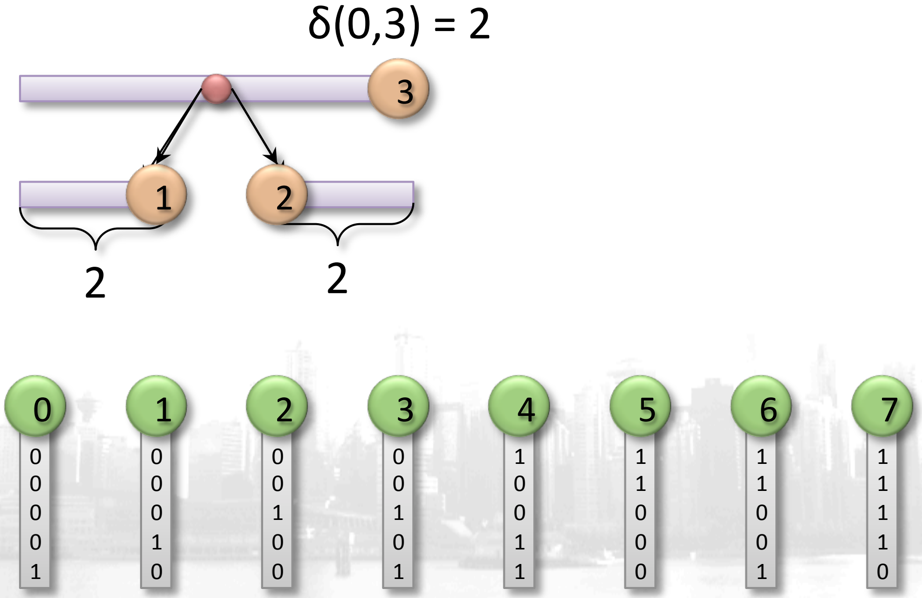
δ(2,3) = 4 ✓ δ(3,4) = 0



Algorithm


δ(0,3) = 2
3
?
δ(1,3) = 4
0
0
0
0
0
1
1
0
0
0
1
0
2
0
0
1
0
0
3
0
0
1
0
1
4
1
0
0
1
1
5
1
1
0
0
0
6
1
1
0
0
1
7
1
1
1
1
0

Algorithm


δ(0,3) = 2
3
1
2
2
2
0
00001
1
00010
2
00100
3
00101
4
10011
5
11000
6
11001
7
11110

Algorithm

For each node i=0..n-2 in parallel:

1. Determine direction of the range
2. Expand the range as far as possible
3. Find where to split the range
4. Identify children

Binary search

$$\boxed{\mathcal{O}(n \log h)}$$

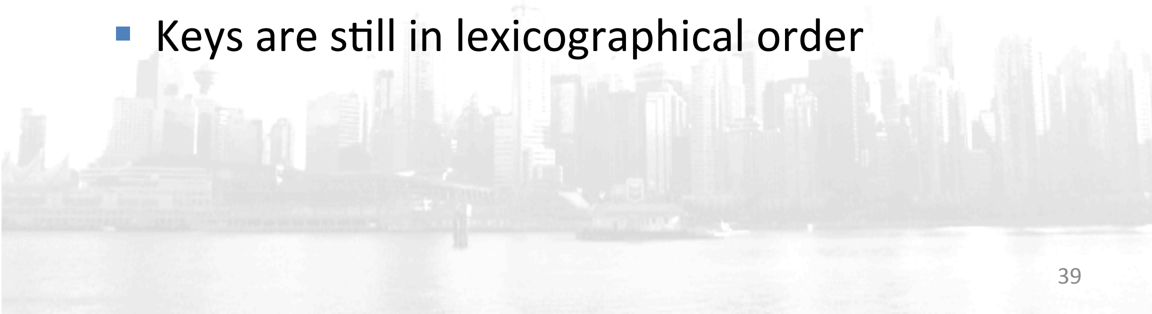
Duplicate keys

- The algorithm only works with unique keys
  - Duplicates are common in practice

Duplicate keys

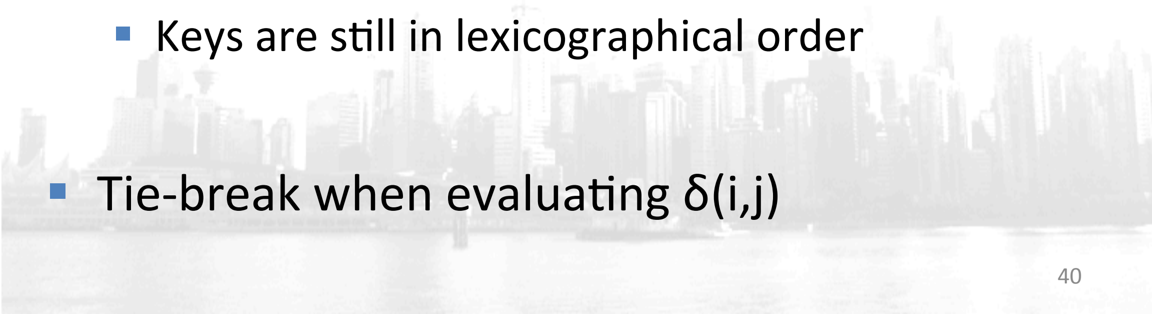
- The algorithm only works with unique keys
  - Duplicates are common in practice

- Trick: Augment each key with its index
  - Distinguishes between duplicates
  - Keys are still in lexicographical order



Duplicate keys

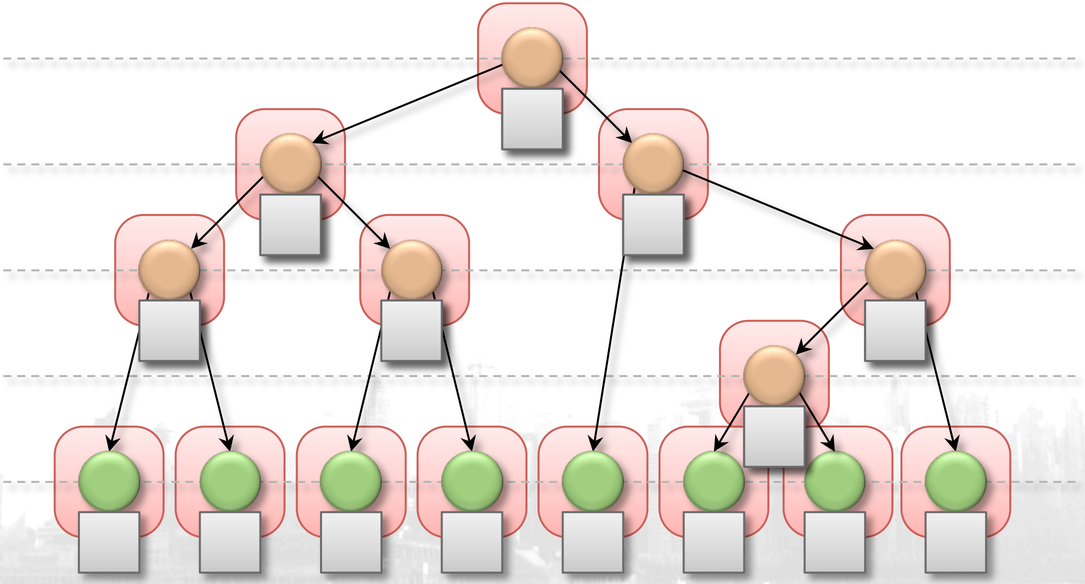
- The algorithm only works with unique keys
  - Duplicates are common in practice
- Trick: Augment each key with its index
  - Distinguishes between duplicates
  - Keys are still in lexicographical order
- Tie-break when evaluating δ(i,j)



LBVH

1. Assign Morton codes
2. Sort primitives
3. Generate hierarchy
4. Fit bounding boxes

Lauterbach et al. [2009]



Our method

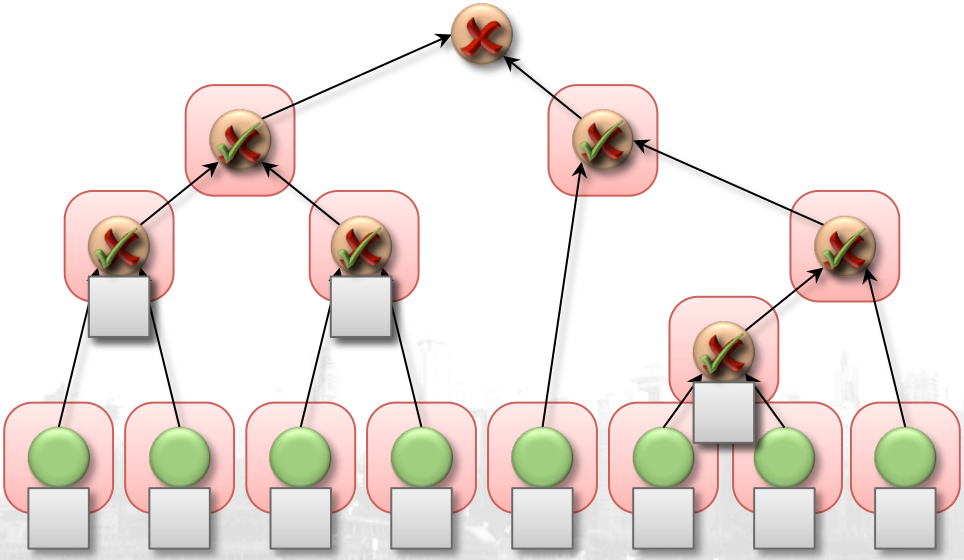
- Need a different approach
  - How many levels are there?
  - Which nodes are located on a given level?

Our method

- Need a different approach
  - How many levels are there?
  - Which nodes are located on a given level?

- Traverse paths in the tree in parallel
  - Start from leaves, advance toward the root
  - Terminate threads using per-node atomic flags

Our method



Results

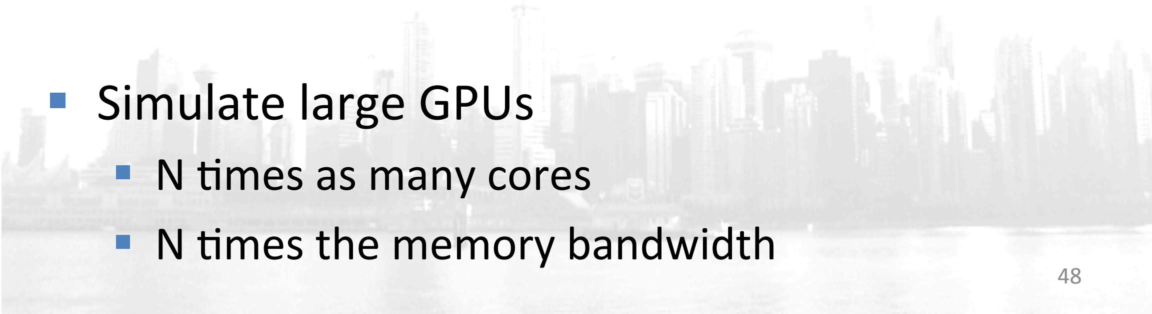
- Evaluate performance on GTX 480 (Fermi)
  - CUDA, 30-bit Morton codes

Results

- Evaluate performance on GTX 480 (Fermi)
  - CUDA, 30-bit Morton codes
- Compare against Garanzha et al. [2011]
  - Identical tree (top-level SAH splits disabled)

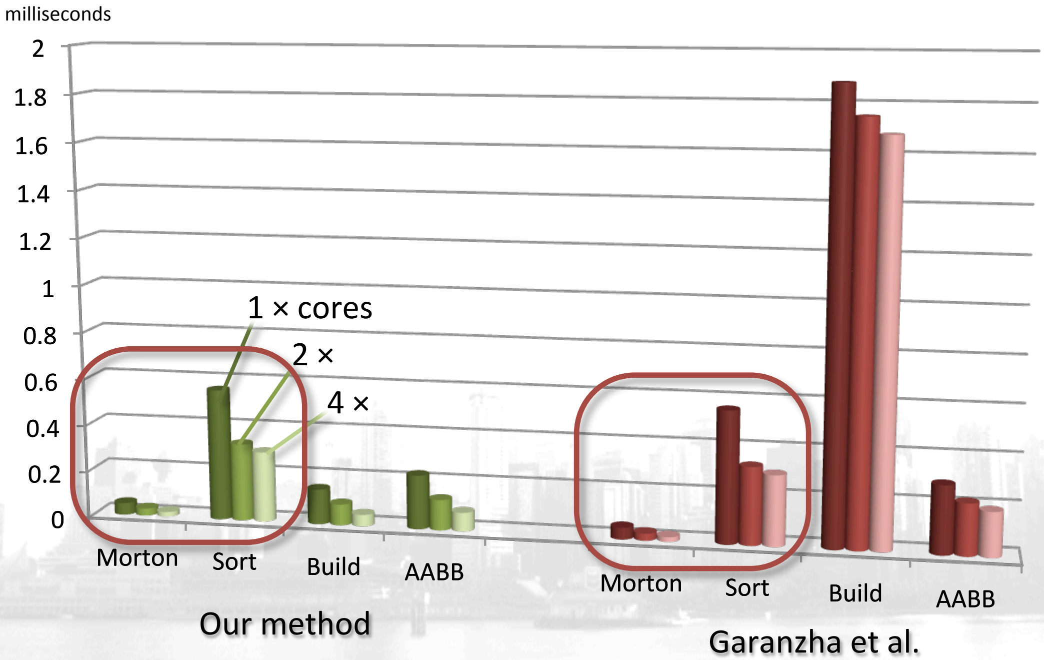
Results

- Evaluate performance on GTX 480 (Fermi)
  - CUDA, 30-bit Morton codes
- Compare against Garanzha et al. [2011]
  - Identical tree (top-level SAH splits disabled)
- Simulate large GPUs
  - N times as many cores
  - N times the memory bandwidth



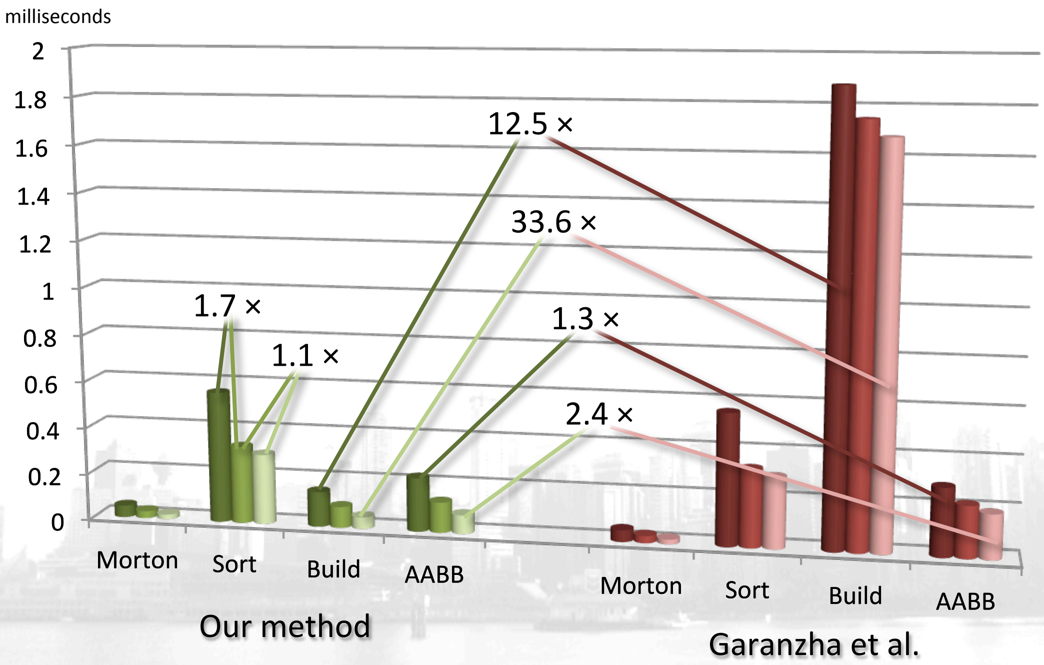
Results

Fairy Forest
174K triangles



Results

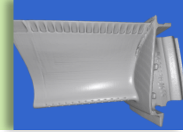
Fairy Forest
174K triangles

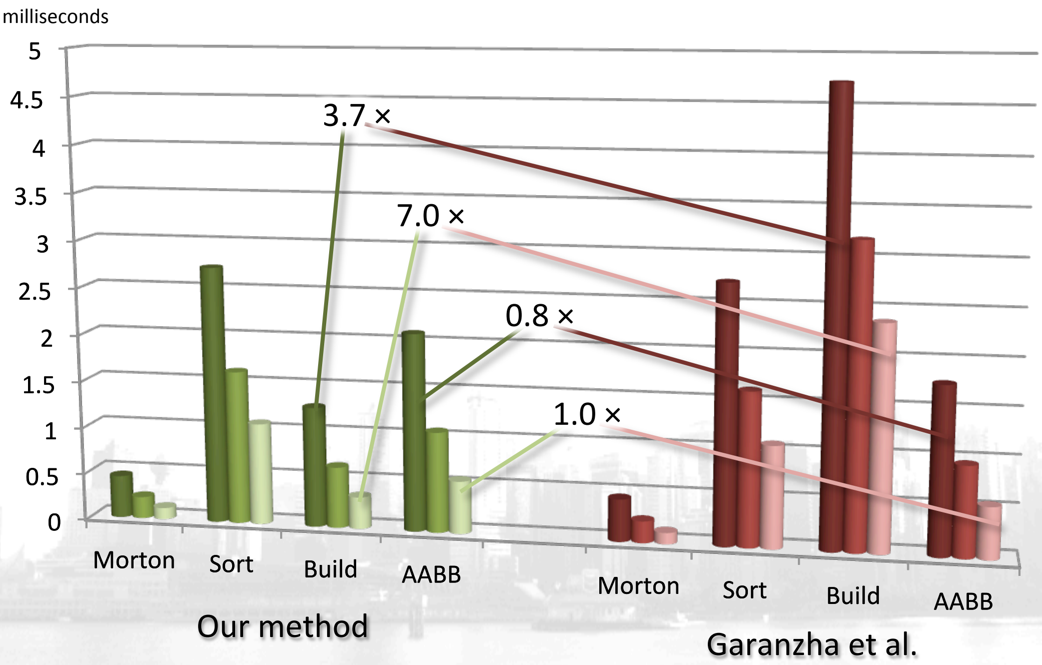


Results

Turbine Blade

1.77M triangles

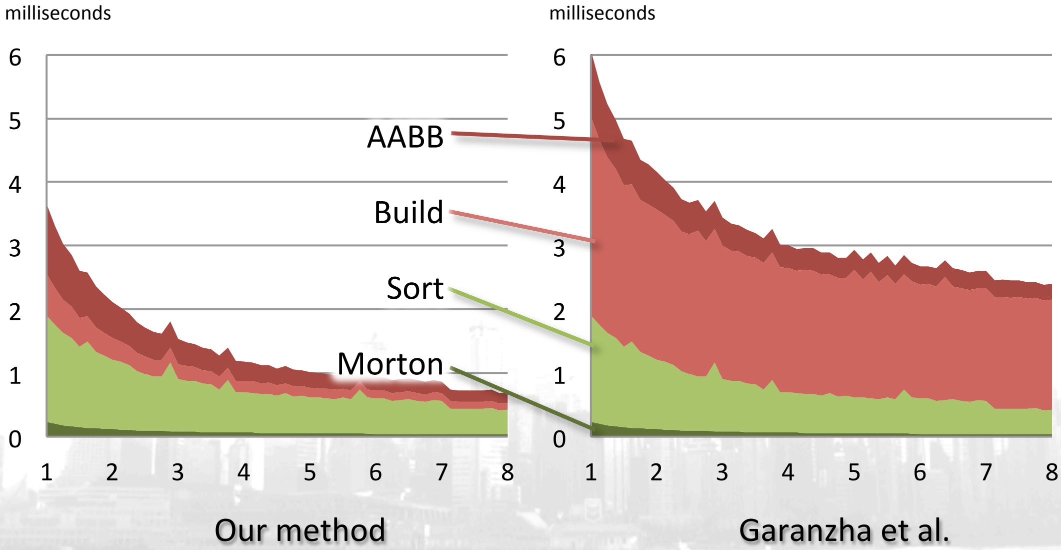




Results

Stanford Dragon

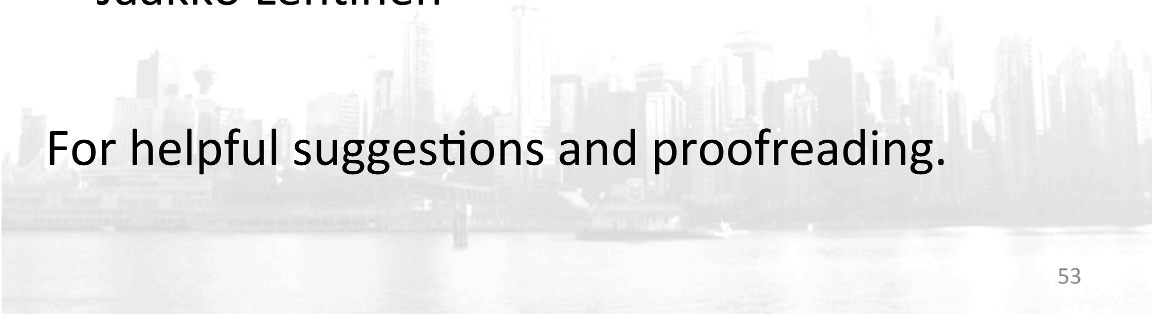
871K triangles



Acknowledgements

- Timo Aila
- Samuli Laine
- David Luebke
- Jacopo Pantaleoni
- Jaakko Lehtinen

For helpful suggestions and proofreading.



Thank You

- Questions