graphics hardware 06

Compressed Lossless Texture Representation and Caching

Tetsugo Inada

University of Waterloo and Sony

Michael D. McCool

University of Waterloo and RapidMind

Motivation: Compression

+ Lowers memory requirements

+ Lowers bandwidth requirements

Lossy compression:

- Can result in poor image quality
- Can’t use for other kinds of data

Lossless compression:

Would not have these problems

Can substitute for other data structures

Issues: Lossless Compression

- ***Must support variable bit-rate coding***
- ***Must support random access***

- Block-based
- Low, predictable latency
- Multitexturing
- Renderable (optional)

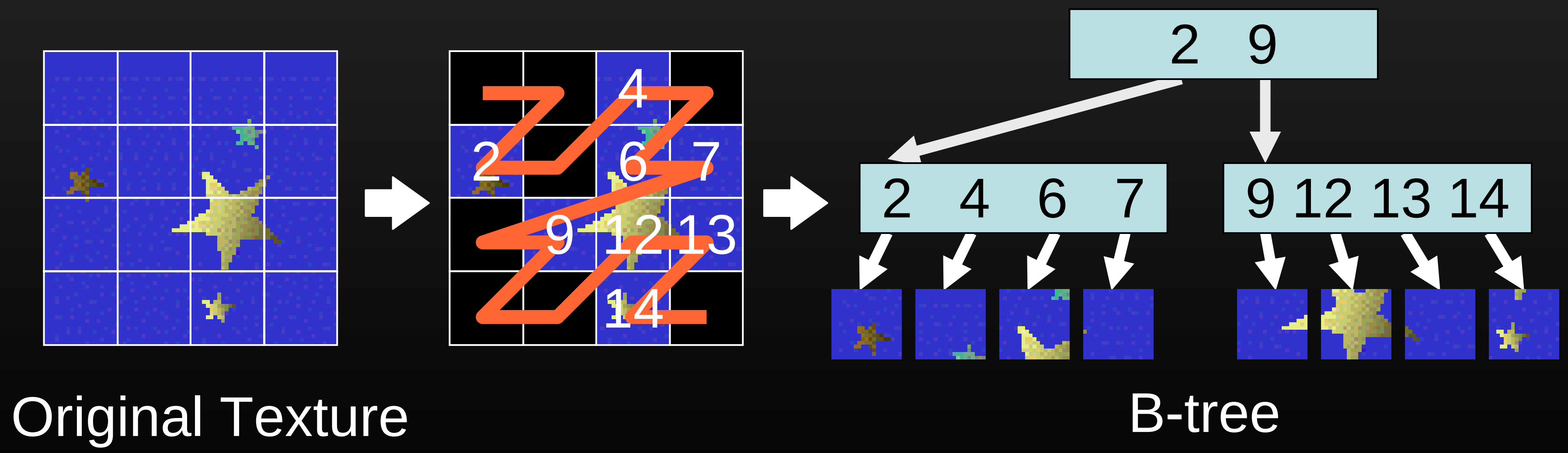
graphics hardware 06

Previous Work

- S3TC [Iourcha, 99], iPACKMAN [Strom, 05]
  - Lossy, fixed rate
- Talisman [Torborg, 96]
  - Lossy, fixed rate JPEG-like
  - Long latency
  - Two-level cache structure (similar to ours)
- B-tree indexing [Yee, 04]
  - Lossless
  - O(1) memory allocation, block oriented
  - Only based on exploiting sparsity

B-Tree Indexing [Yee, 04]

- Divide texture into pixel tiles
- Identify void (background) and occupied tiles
- Assign 1D keys to occupied tiles
- Insert into B-Tree



B-Tree Indexing [Yee, 04]

- Lossless
- No external fragmentation
  - Blocks are connected by pointers
- Random access to **uniform** sized blocks
- Exploits only sparsity
  - Narrows its application area

graphics hardware 06

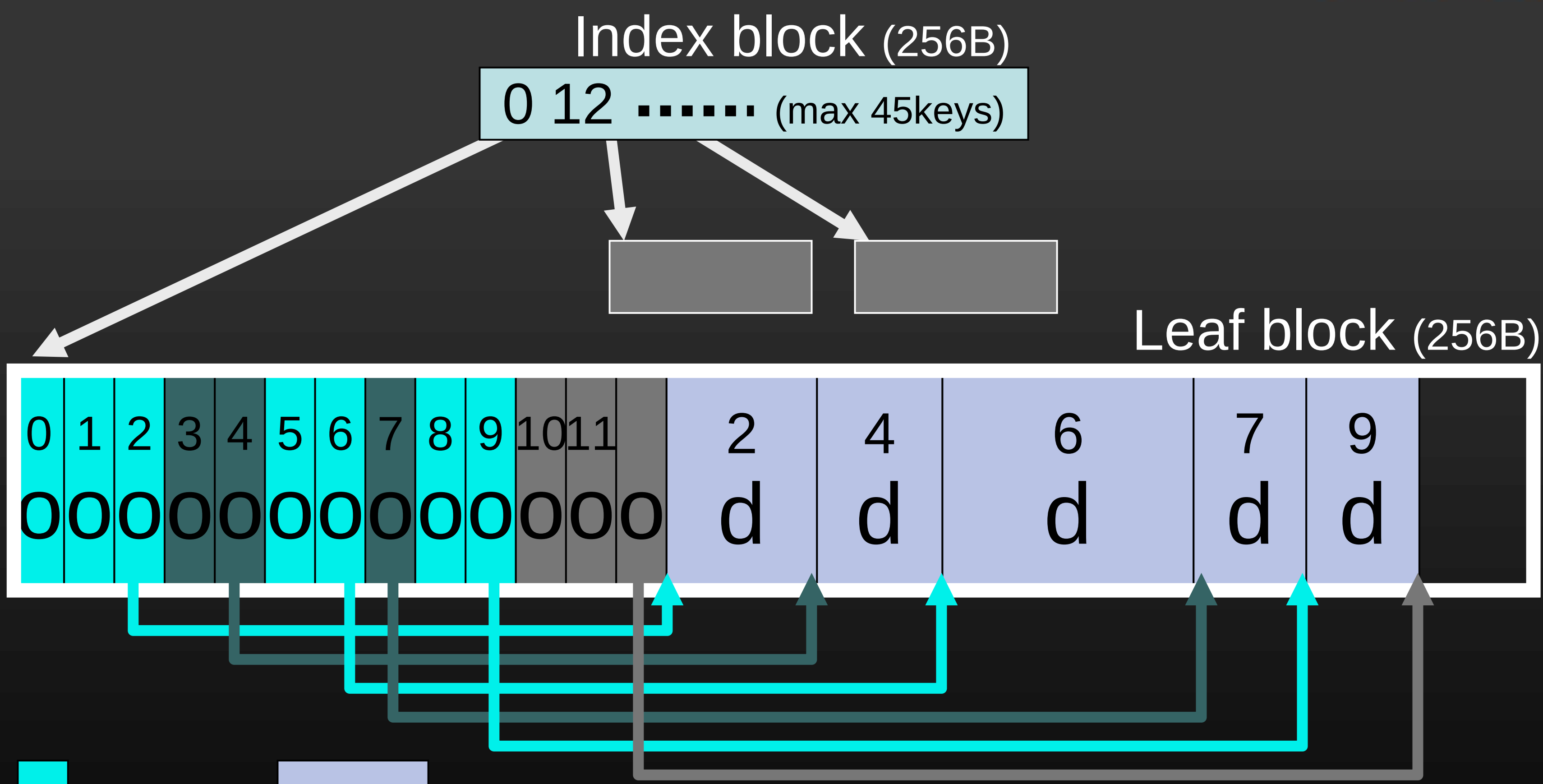
Proposed Method

- Based on Yee's B-Tree
  - Lossless
  - No external fragmentation
- Random access to **variable** sized blocks
- Exploits sparsity ***and*** variable bit-rate compression

Proposed Method

- Index structure
- Variable bit-rate coding
- Specialized cache architecture

Index Structure

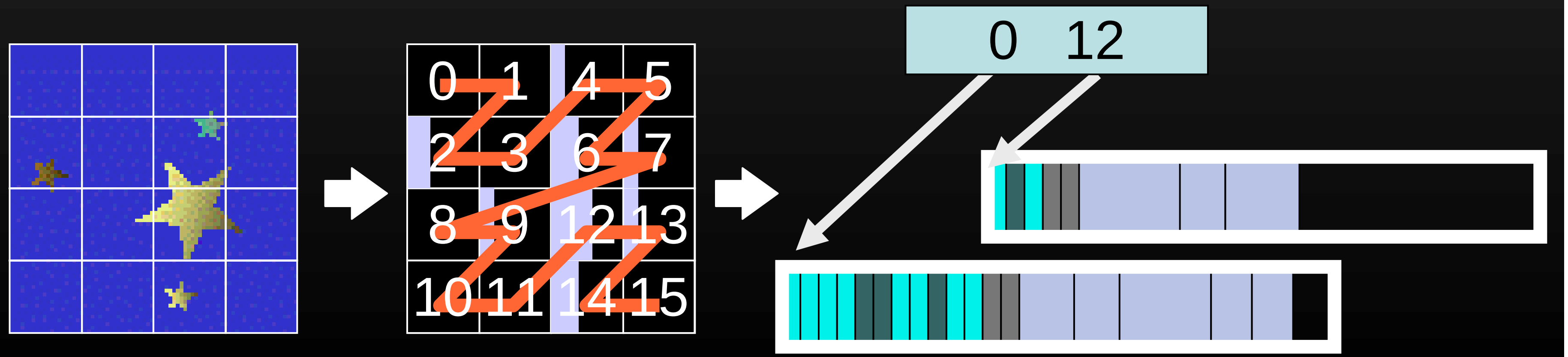


tile size = offset difference

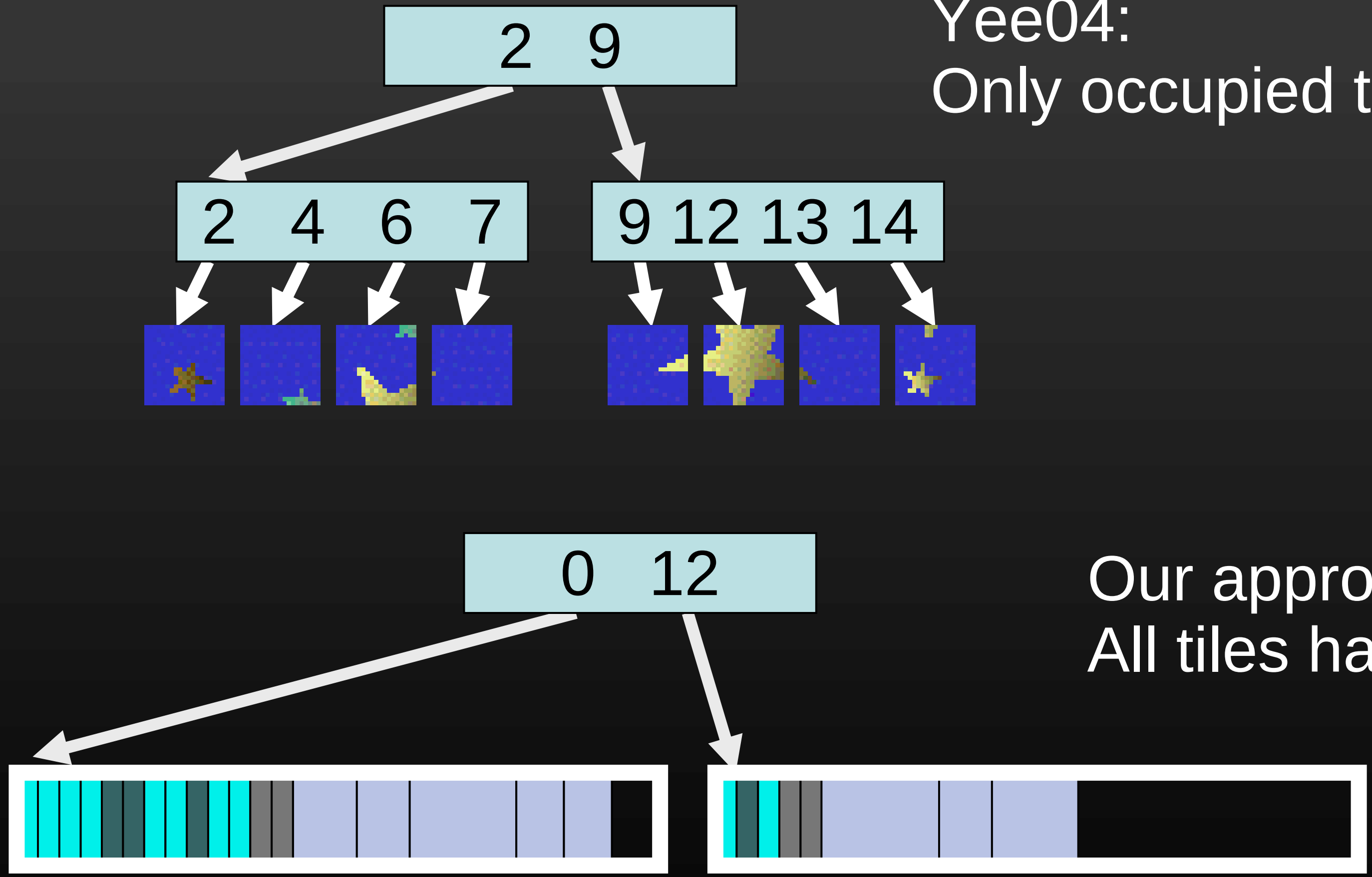
difference = 0 -> void tile

Index Structure

- Divide texture into pixel tiles
- Identify void (background) and occupied tiles
- Assign 1D keys to ***ALL*** tiles
- ***Compress occupied tiles***
- ***Pack occupied tiles into leaf blocks***
- Insert into B-Tree



Index Structure



Yee04:
Only occupied tiles have keys

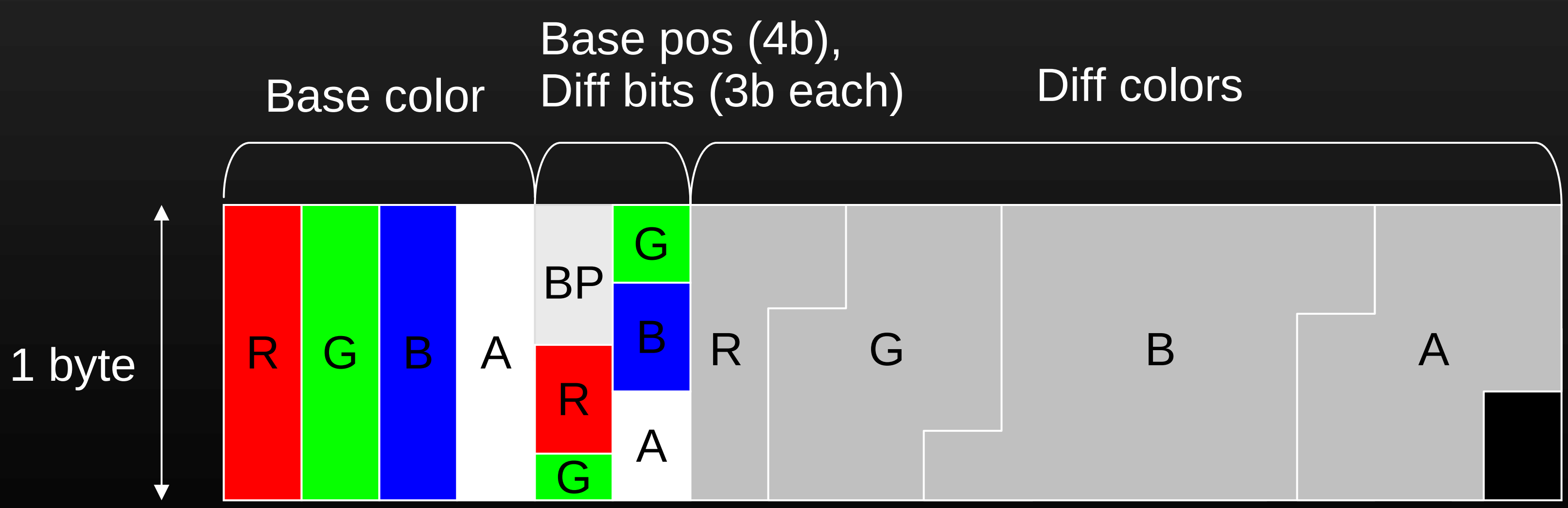
Our approach:
All tiles have keys

Variable Bit-rate Compression

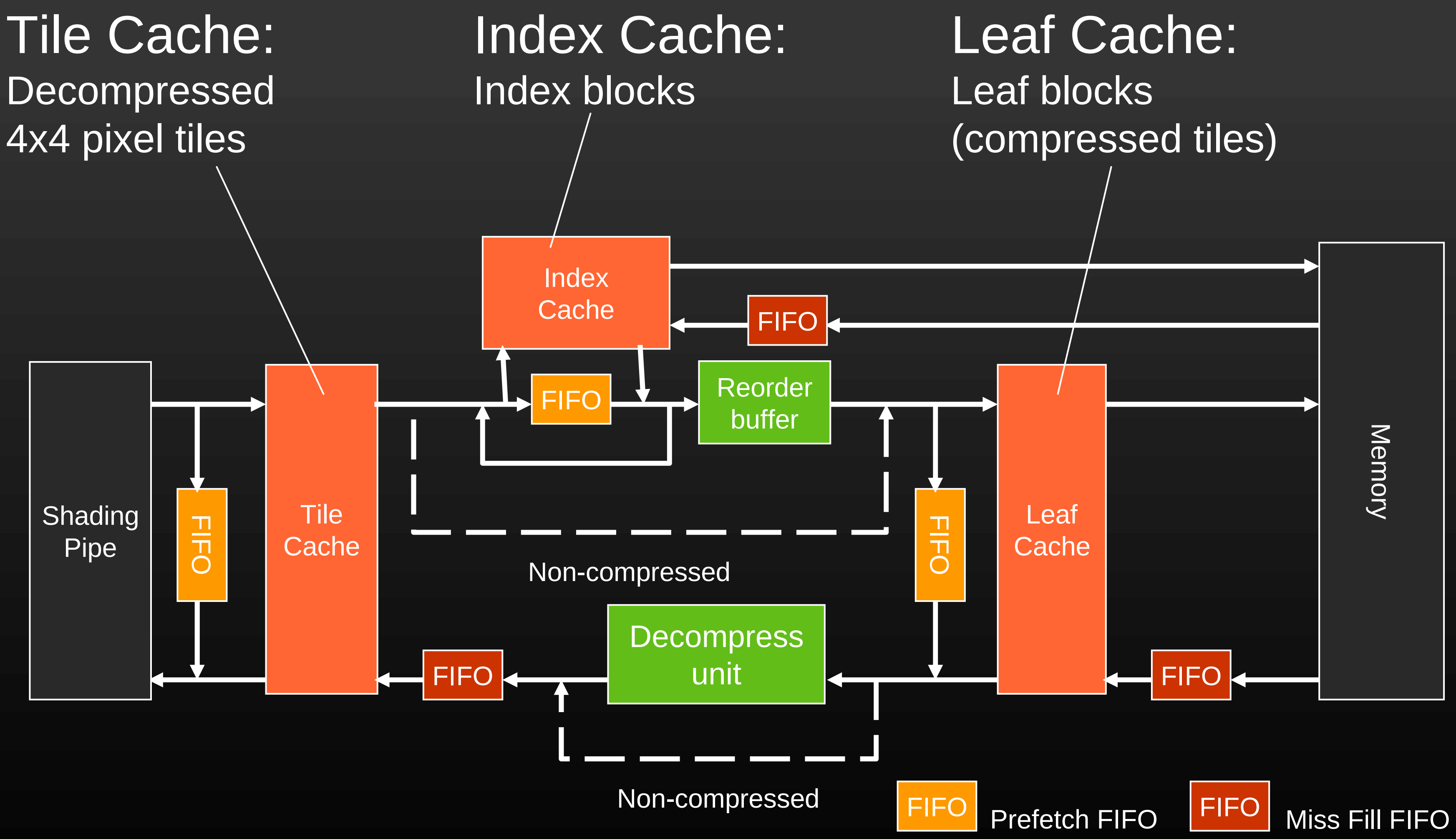
- **Independent** from our cache and index
  - DCT & Huffman coding (JPEG)
  - Wavelet & Arithmetic coding (JPEG2000)
- ”Difference Packing” (our approach)
  - Packing color differences from the base color in minimum bit length
  - Ease of hardware implementation
  - Low latency

Difference Packing

- Select a base pixel from 16 (4x4 tile)
- Pack differences to the 15 other pixels
  - Ex. If all differences are within -4 to 3, they are packed into 3 bits each.



Cache Architecture



Results

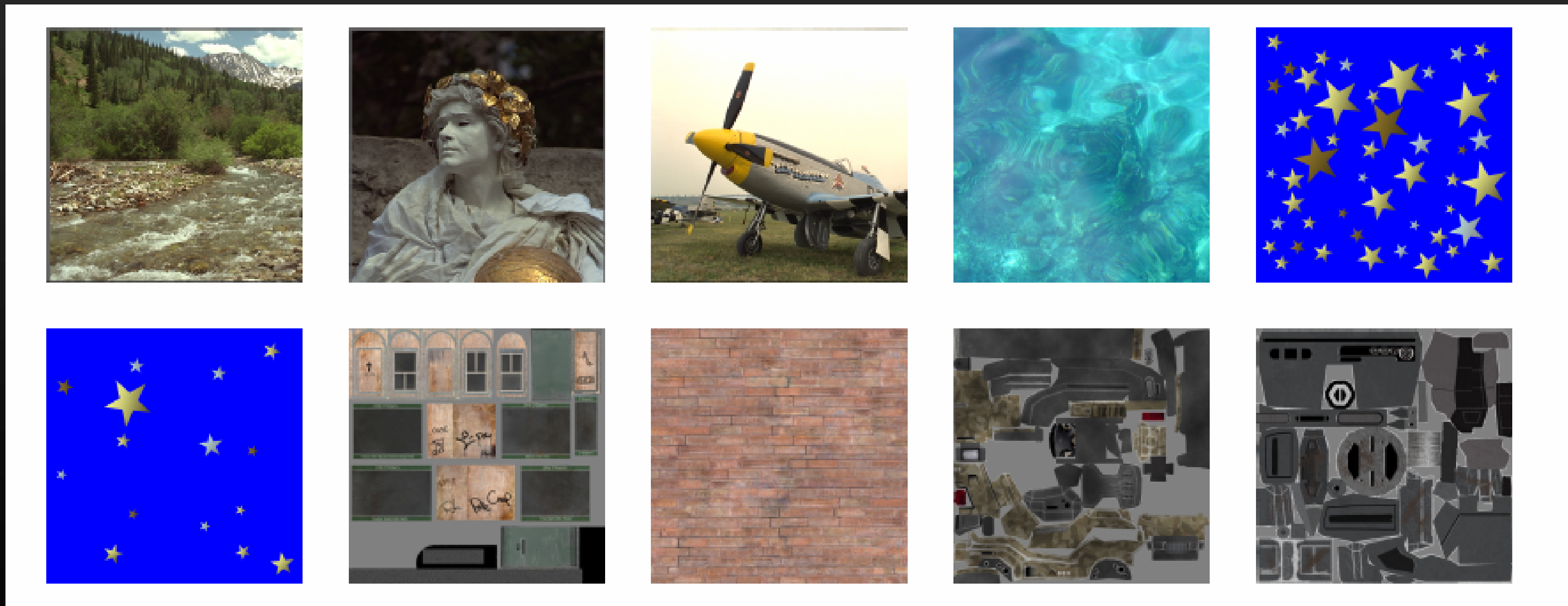
- Compression ratios
- Hardware simulation results
  - Bandwidth consumption
  - Latency
- Cycle accurate simulator
  - Workloads generated by OpenGL apps
  - Modified Mesa to generate traces
  - Morton curve rasterization order

Test Suite: Images

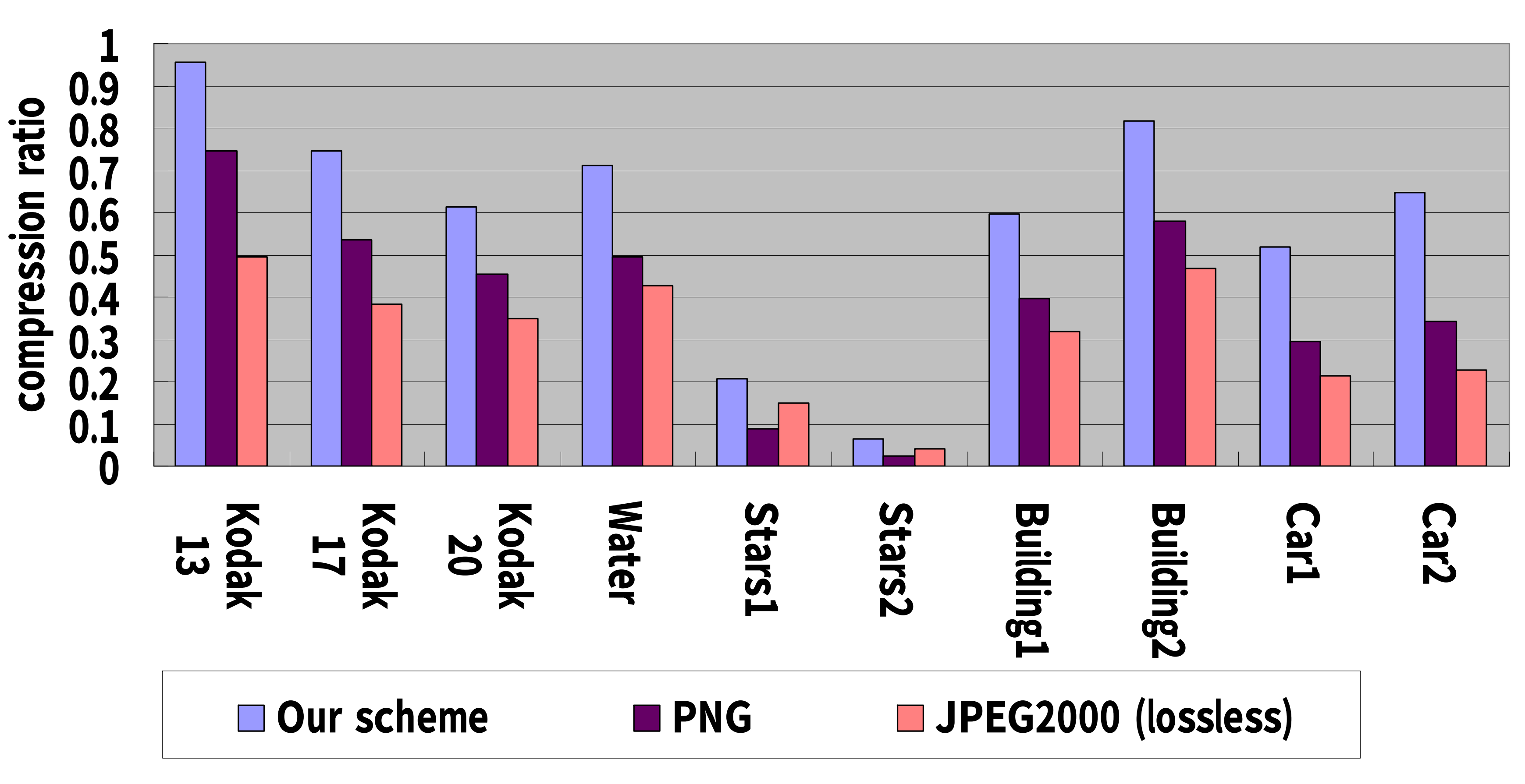
Kodak13, Kodak17, Kodak20, (natural images)

Water, Stars1, Stars2, (tileable textures)

Building1, Building2, Car1, Car2 (models)

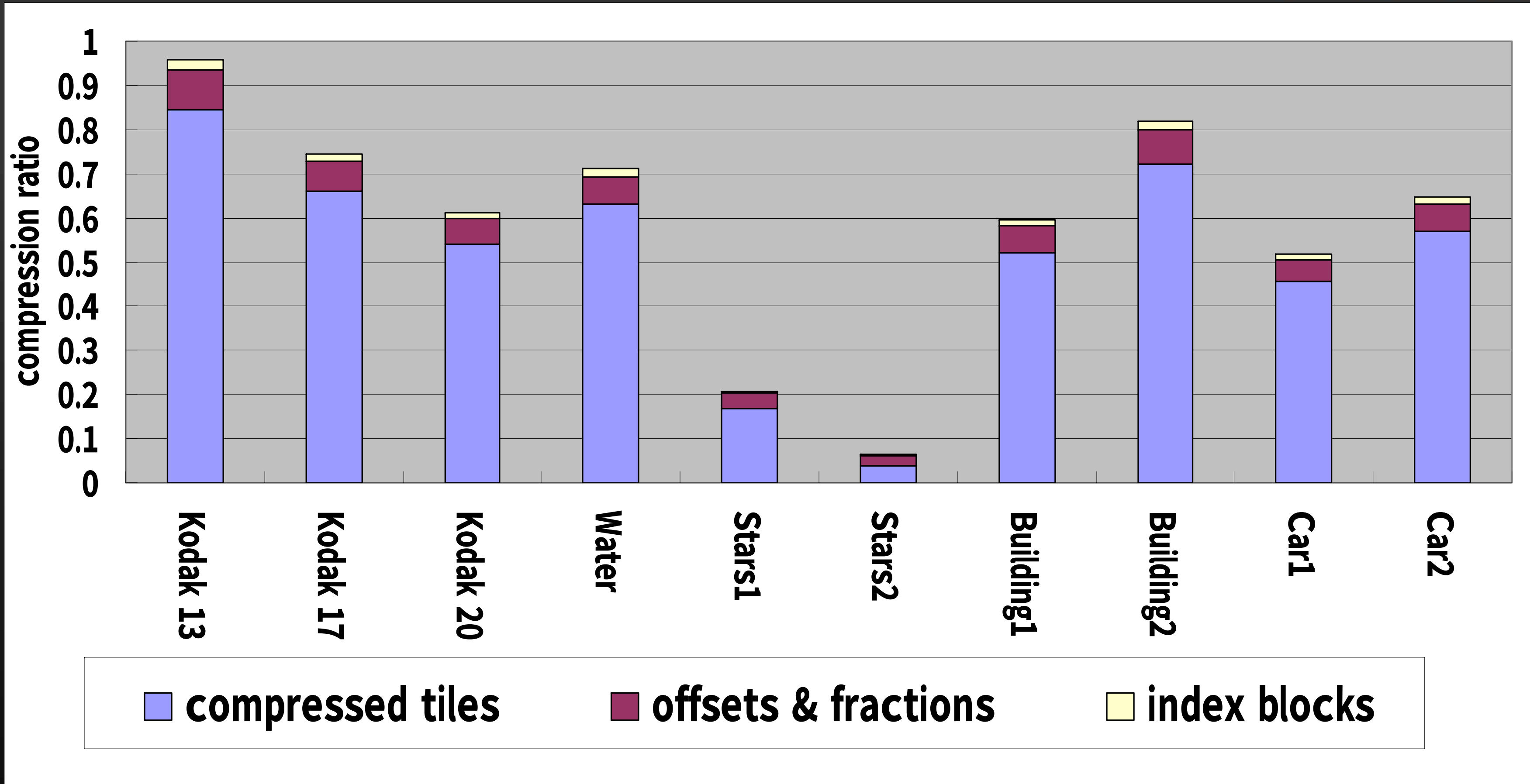


Compression Ratios



B-Tree heights: 3 to 4

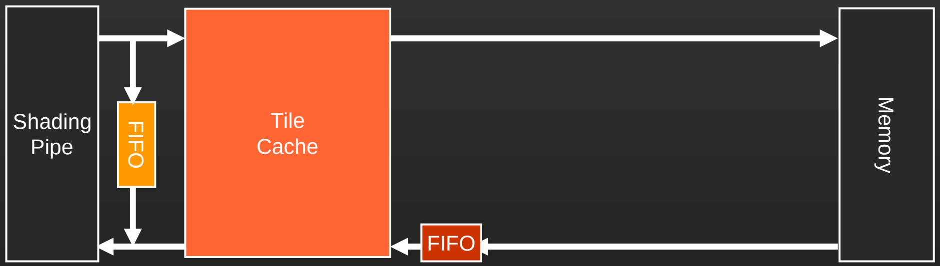
Compression Ratios



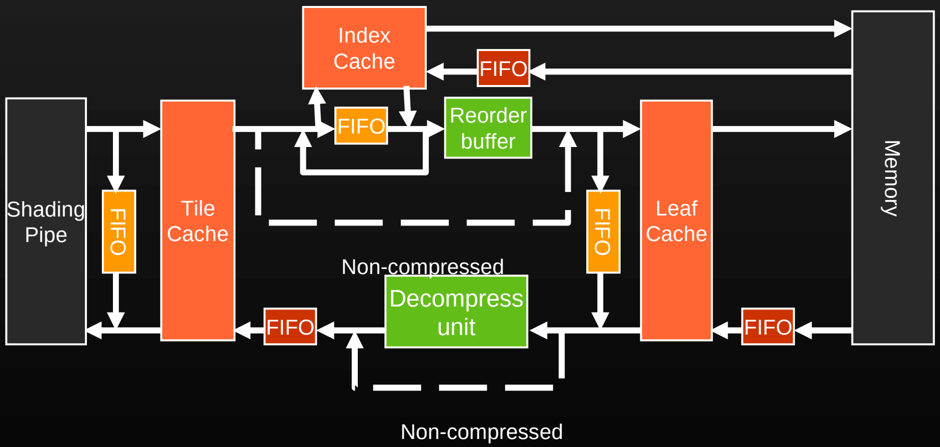
Overhead for random access: less than 11%

Comparison

- Conventional Architecture
  - 32.0KB
  - Same *area*



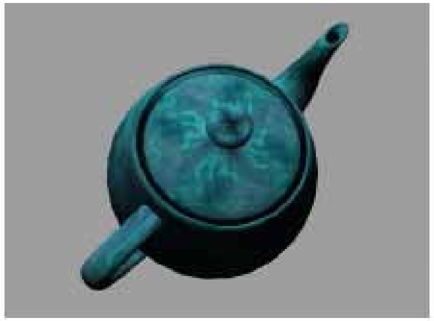
- Our Architecture
  - Tile Cache: 2.0KB
  - Leaf Cache: 16.0KB
  - Index Cache: 4.0KB



Test Suite: Scenes

- Scenes

Quad
Screen: 512x512
Texture: 512x512
Unique texels/frag: 1.0



Teapot
Screen: 640x480
Texture: 512x512
Unique texels/frag: 0.475

Building
Screen: 640x480
Texture: 1024x1024 and 1024x1024
Unique texels/frag: 0.868

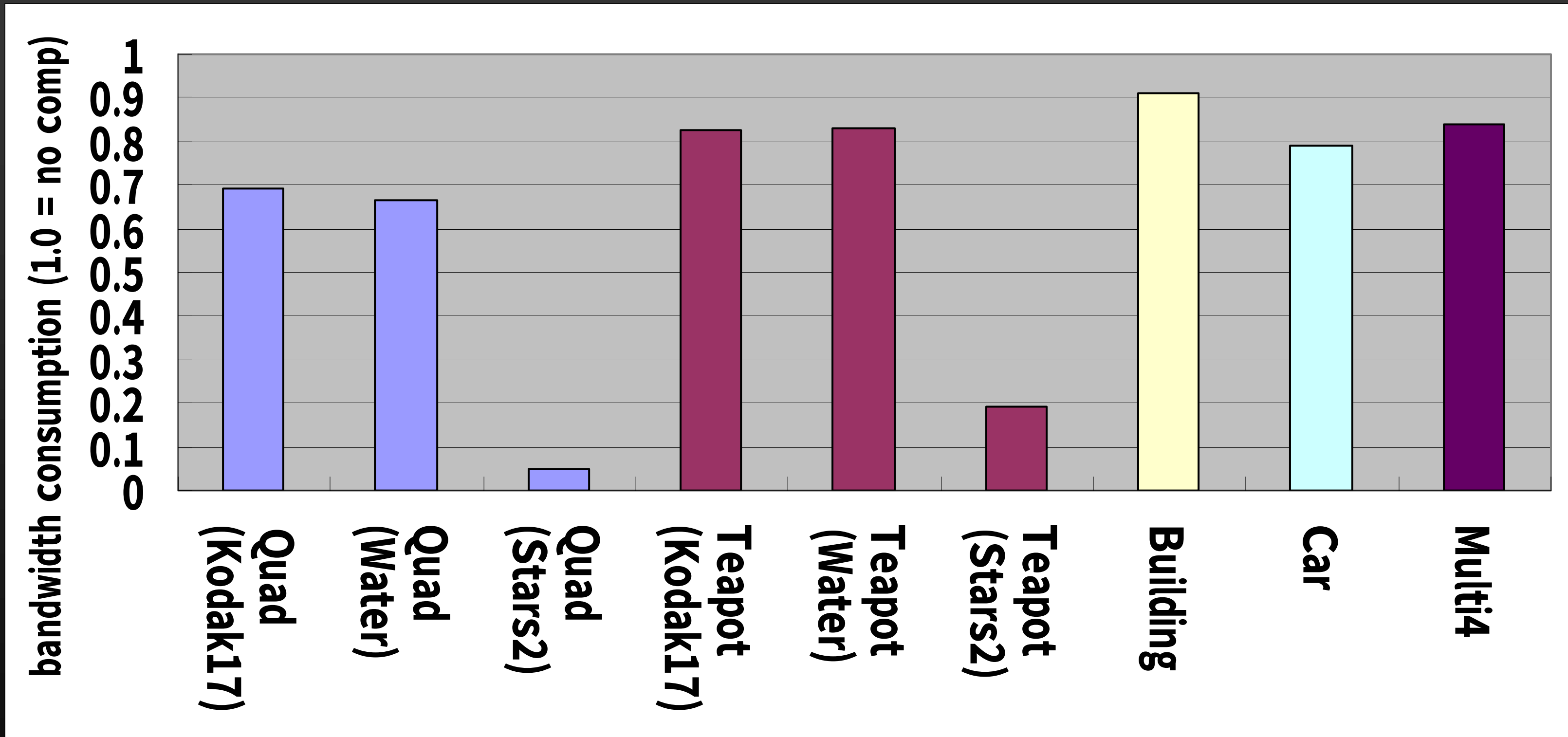
Car
Screen: 640x480
Texture: 1024x1024 and 512x512
Unique texels/frag: 1.635

Multi4
Screen: 640x480
Unique texels/frag: 5.068966

Multitexturing scene with
- multiple shaders
- multiple textures per shader

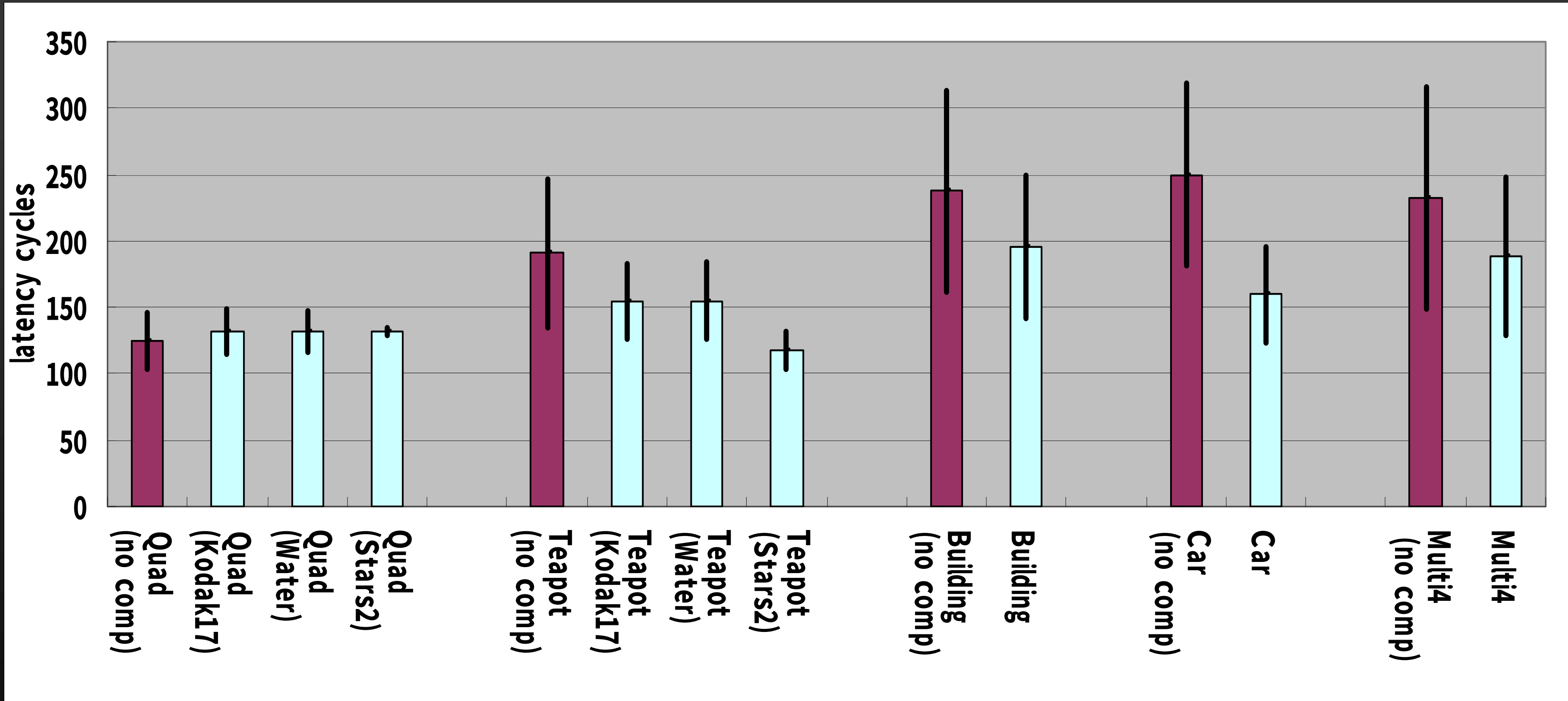
graphics hardware 06

Bandwidth Consumption



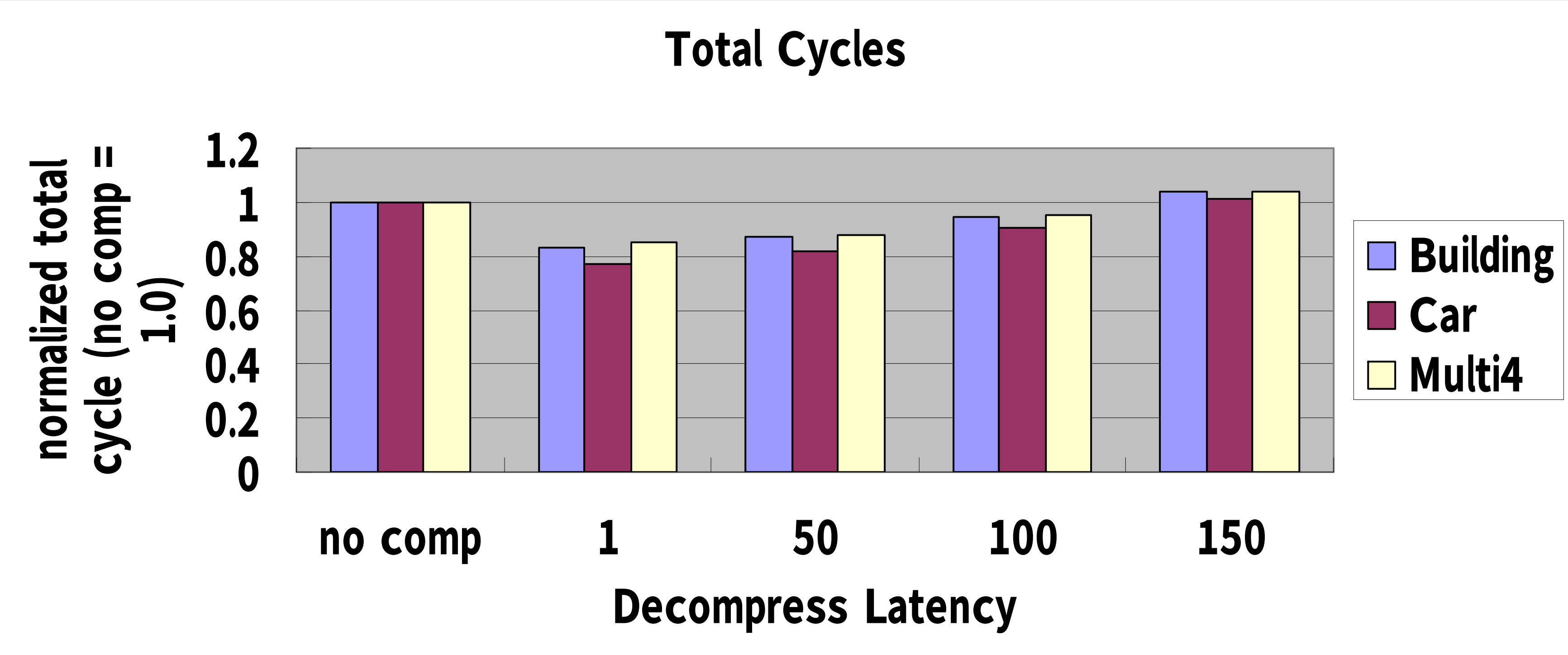
- Stars2: 5 to 19%
- Others: 66 to 91%

Overall Latency



- Compression ***lowered*** averages and standard deviations of latency

Effects of Decompression Latency



- Decompression latency could actually be increased significantly without impacting performance!

Summary

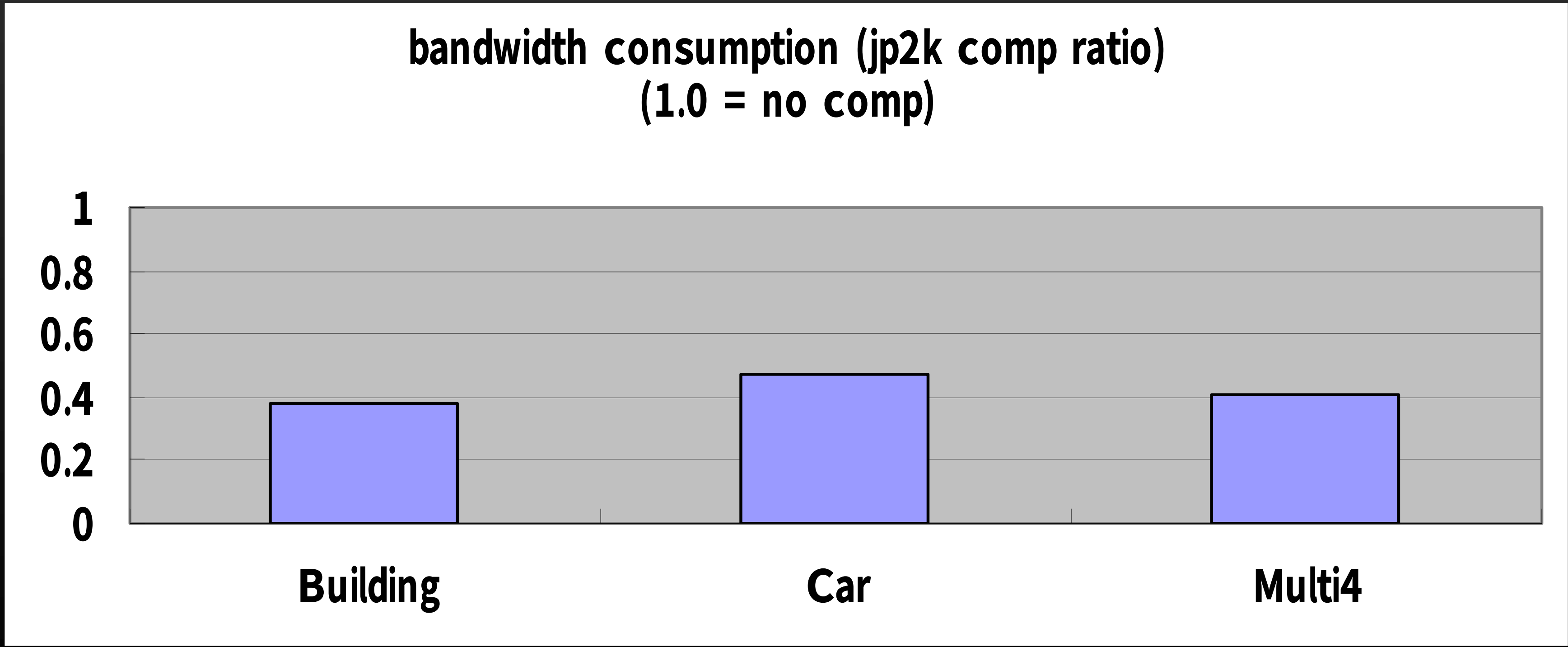
- Architectural support for variable bit-rate compression and random access
  - Index structure is independent from variable bit-rate compression
- "Difference packing" compression
  - Low latency
  - Moderate compression ratio
- Higher latency can be tolerated
  - ***Better compression schemes?***

Extensions

- Other compression methods
  - JPEG2000 (lossless)
  - Lossy compression

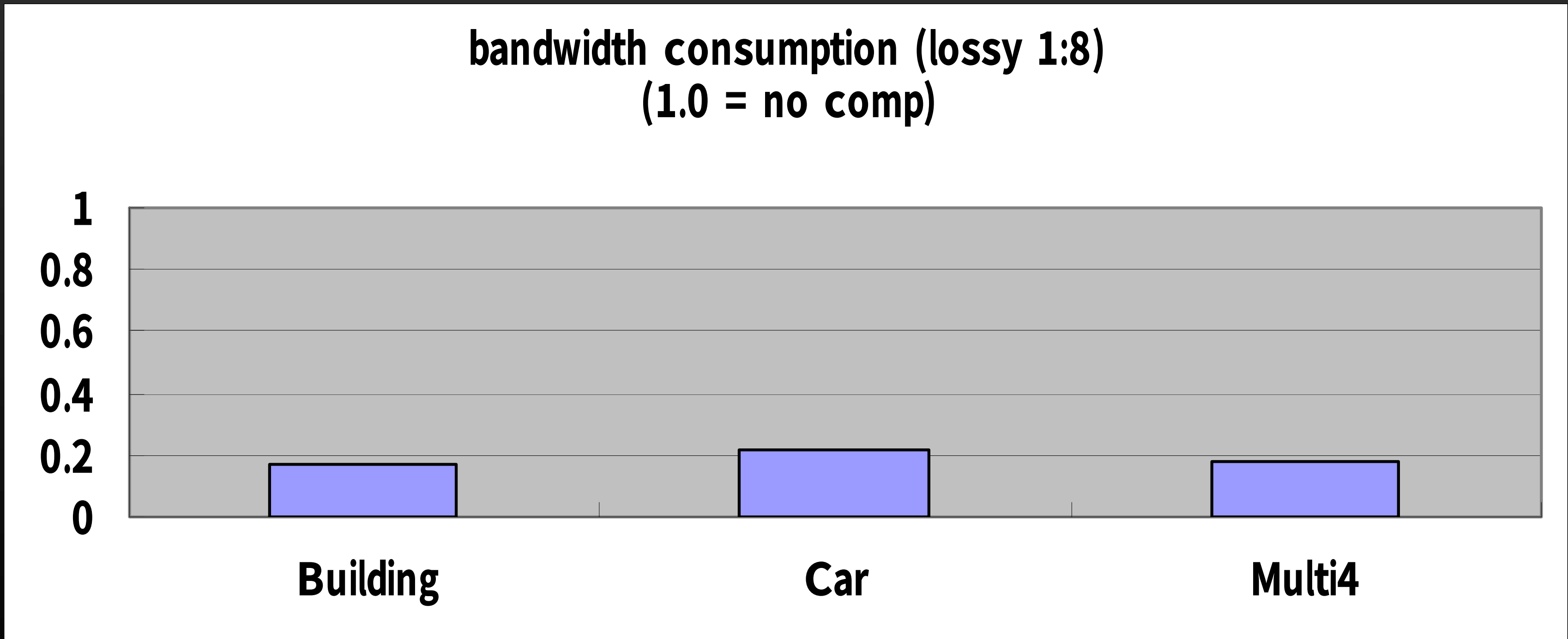
JPEG2000: Bandwidth

- Assume: 150 cycle decompression latency
- Applied JPEG2000's compression ratio



JPEG2000: Lossy Compression

- Assume: fixed compression ratio of 1:8



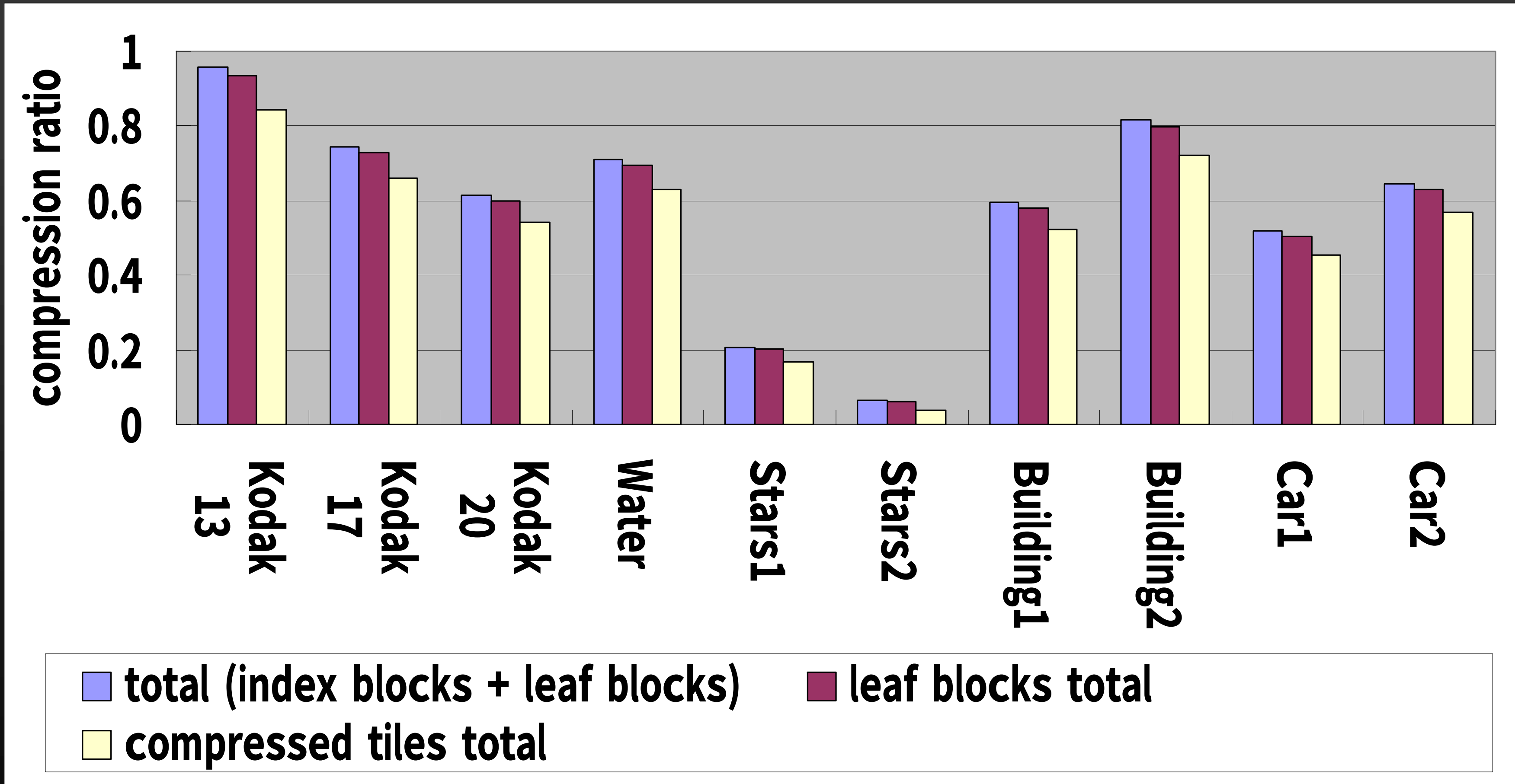
Conclusion

- We have presented an index structure which supports variable bit-rate compression and random access
- High decompression latency can be tolerated
- Compression is feasible and can result in significant bandwidth savings
- Indexing simplifies memory allocation

- Future work includes variable bit-rate lossy compression as well as better lossless compression

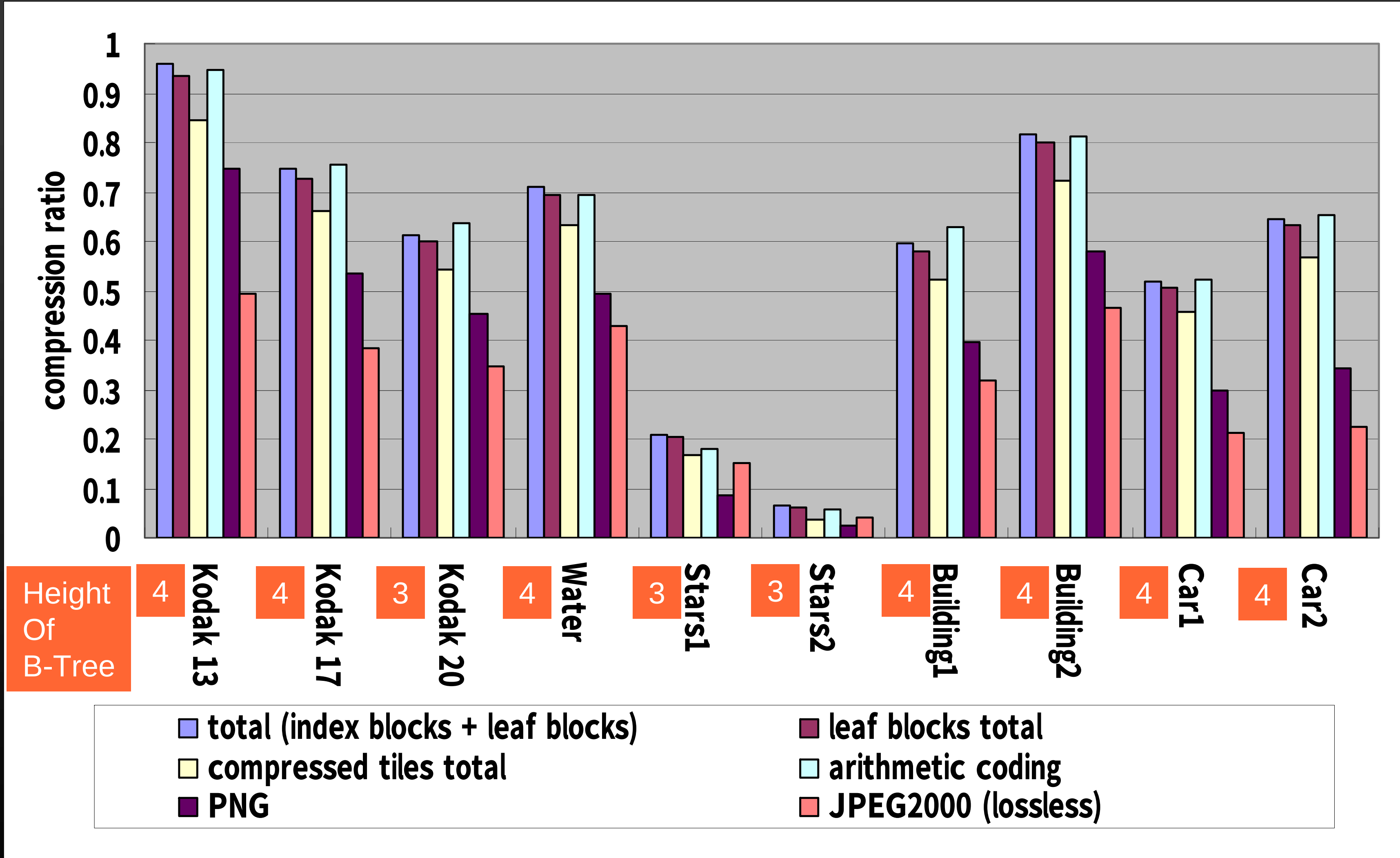
Extra Results

Compression Ratios

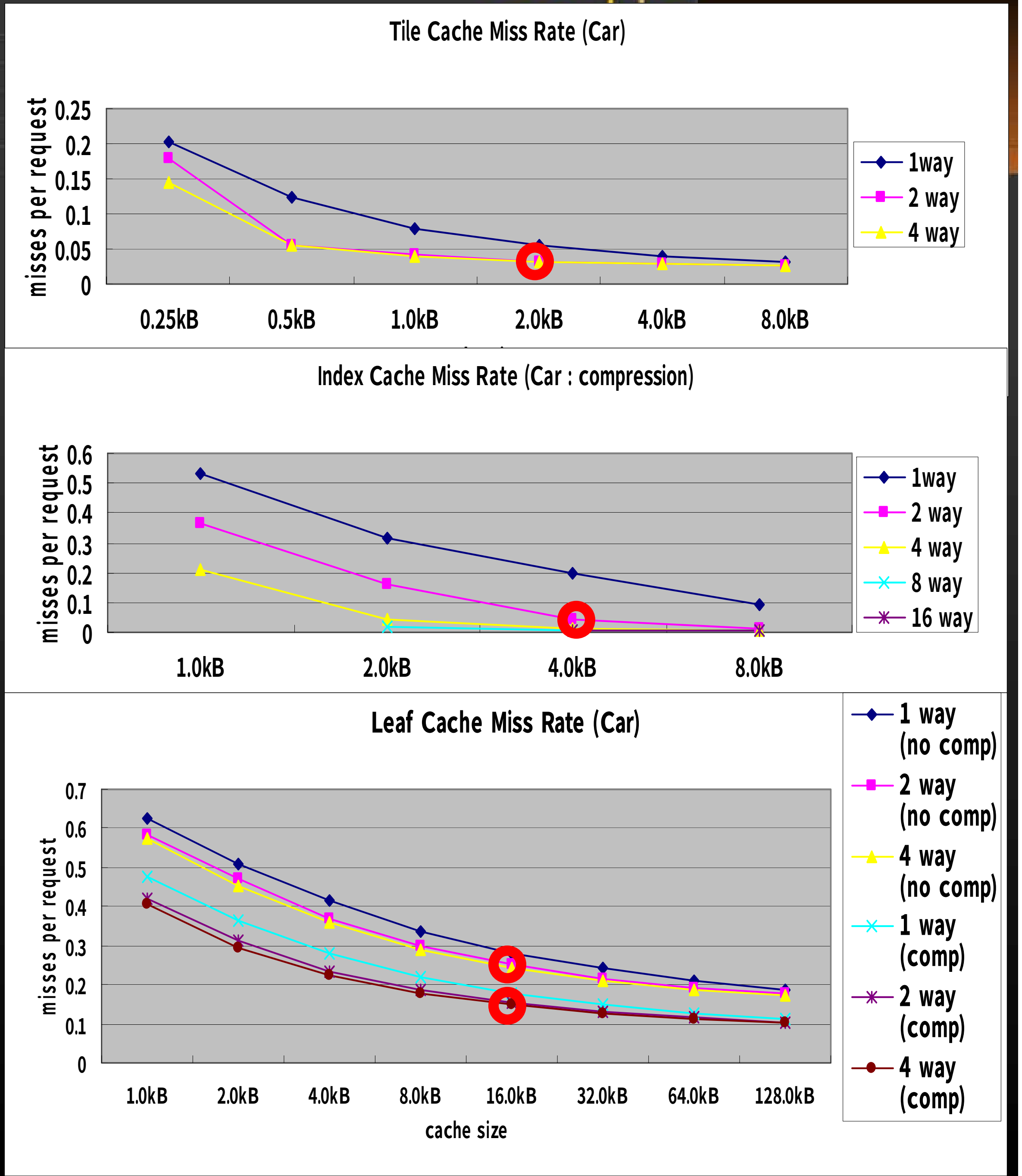


Overhead for random access: less than 10%

Compression Ratios

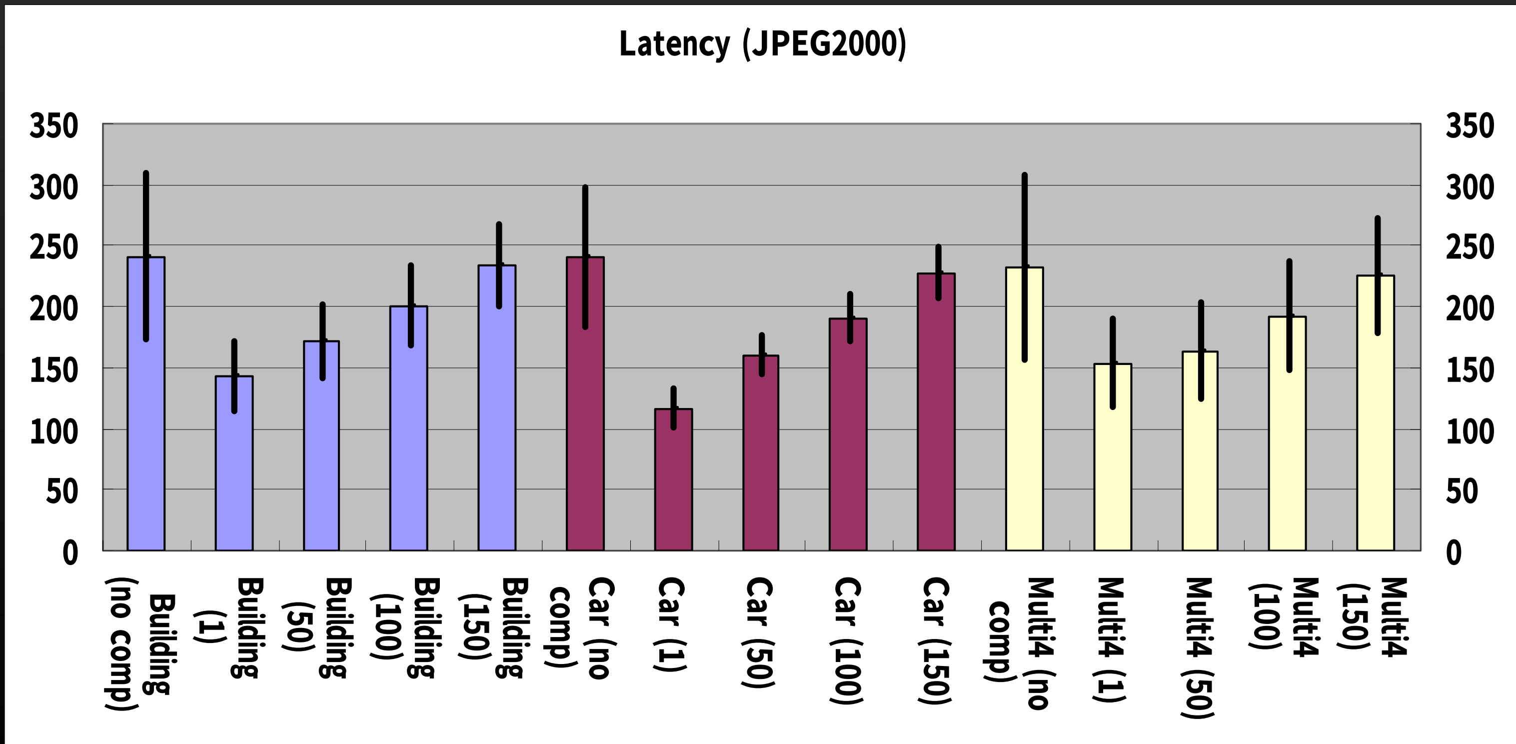


- Tile cache
  - 2-way/2kB
    - (Multi4: 8-way/8kB)
  - Prefetch FIFO : 128
  - Miss Fill FIFO : 2
- Index cache
  - 4-way/4kB
    - (Multi4: 16-way/16kB)
  - **Prefetch FIFO : 1**
    - **Reorder buffer is unnecessary**
  - Miss Fill FIFO : 1
- Leaf cache
  - 2-way/16kB
    - (Multi4: 8-way/64kB)
  - Prefetch FIFO : 32
  - Miss Fill FIFO : 2



JPEG2000: Latency

- Use JPEG2000 lossless compression ratio
- Change decompression latency



Index Cache: Working Set

- Multi4, Atlas textures
  - 1k x 1k textures
    - B-Tree: 3 index blocks + 1 leaf block (level 0)
    - 3x2 index blocks / texture (tri-linear)
  - Working sets
    - 1.5KB, 3.0KB, 6.0KB, 12.0KB (1,2,4,8 tex)