# LightDB

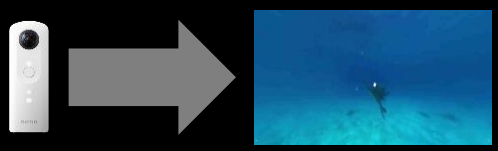
A Database System for
Virtual, Augmented, & Mixed Reality
Video Applications

Brandon Haynes, Amrita Mazumdar, Armin Alaghi,
Magdalena Balazinska, Luis Ceze, & Alvin Cheung

## Spherical Panoramic Images

2

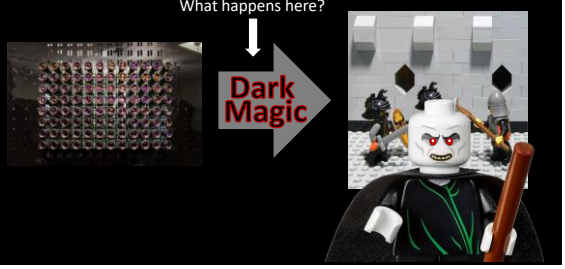
## Spherical Panoramic (360°) Videos



3

## Light Fields

What happens here?




4

## LightDB

Key Features:

- Data management system for VR/AR/MR video applications
- Unified data model for panoramic (360°) and light field video
- Declarative queries with automatic optimization
- Full stack: data ingest, processing, and real-time streaming

Key Results:

- Decreased development complexity (~ $^{1}/_{10}$ LOC)
- Increased performance (up to 4× for real-world workloads)
- Reduced client bandwidth & power requirements

5

## LightDB

Today:

1. LightDB Data Model
2. Physical & Logical Algebra
3. Architecture & Optimizer
4. Application: Predictive 360° Streaming

6

# The Light Field Data Model


7

Spherical Images

Angle $(\theta, \phi)$

360° Videos

Angle $(\theta, \phi)$, Time $(t)$

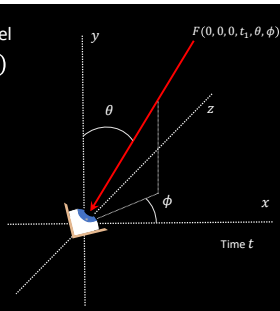
Light Fields

Angle $(\theta, \phi)$, Time $(t)$, Position $(x, y, z)$

8

Light field data model

$F(x, y, z, t, \theta, \phi)$

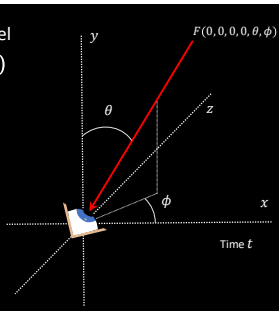



9

Light field data model

$F(x, y, z, t, \theta, \phi)$

Panoramic Image

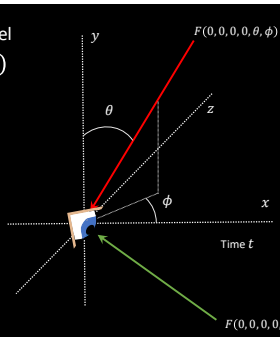



10

Light field data model

$F(x, y, z, t, \theta, \phi)$

Panoramic Image

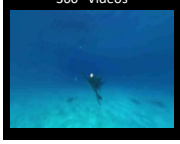


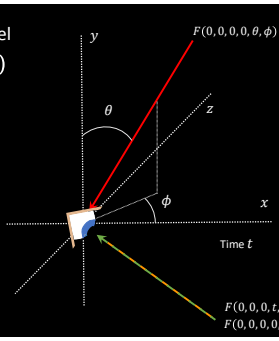

11

Light field data model

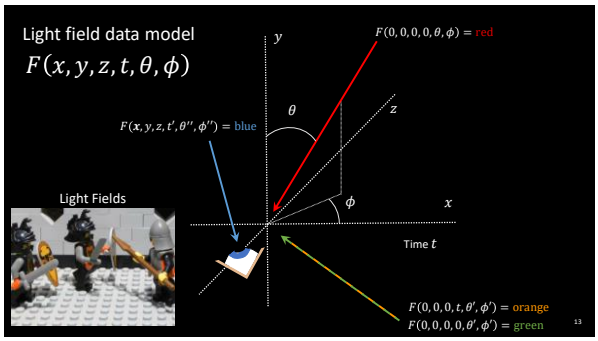
$F(x, y, z, t, \theta, \phi)$

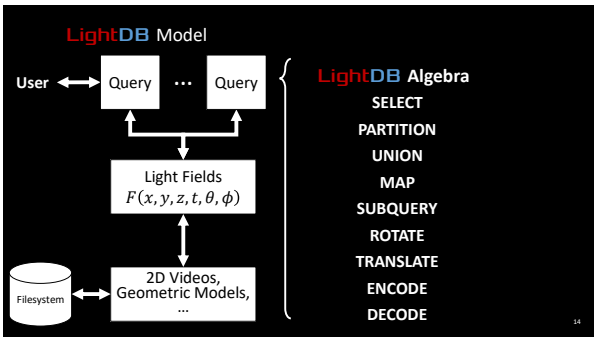
360° Videos

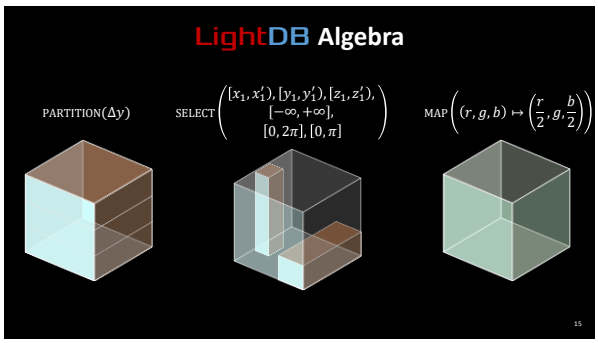


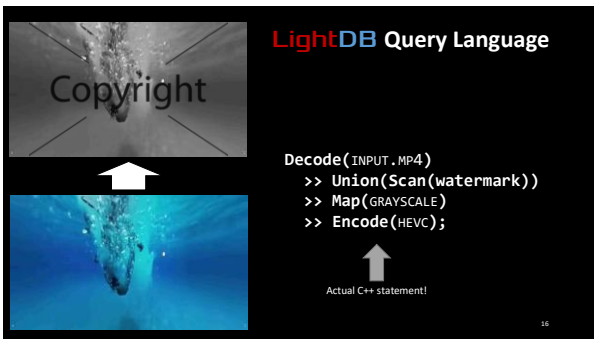


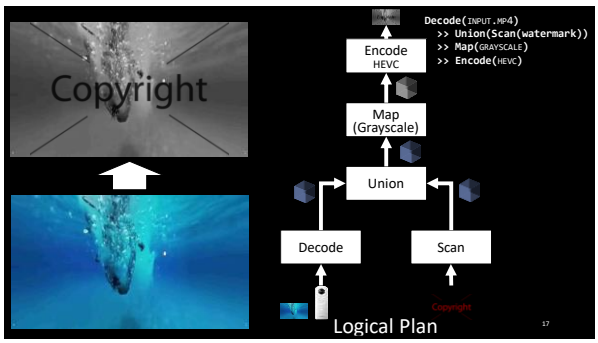

12

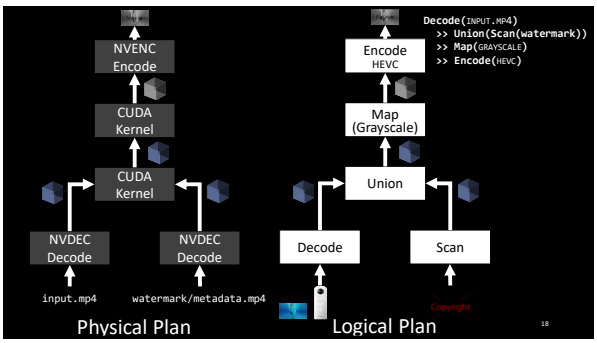

Light field data model
$F(x, y, z, t, \theta, \phi)$
$F(0,0,0,0,\theta,\phi)$ = red
$F(x,y,z,t',\theta'',\phi'')$ = blue
Light Fields
Time $t$
$F(0,0,0,t,\theta',\phi')$ = orange
$F(0,0,0,0,\theta',\phi')$ = green
13

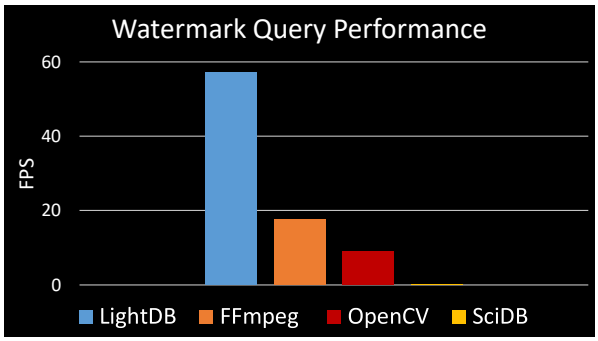

LightDB Model
User
Query
Query
Light Fields
$F(x, y, z, t, \theta, \phi)$
Filesystem
2D Videos, Geometric Models, ...
LightDB Algebra
SELECT
PARTITION
UNION
MAP
SUBQUERY
ROTATE
TRANSLATE
ENCODE
DECODE
14


LightDB Algebra
PARTITION($\Delta y$)
SELECT
MAP
15

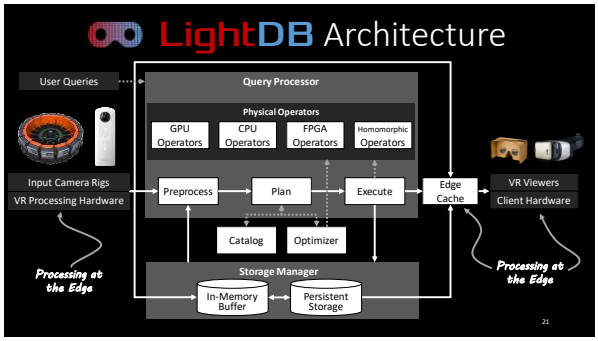

Copyright
LightDB Query Language
Decode(INPUT.MP4)
>> Union(Scan(watermark))
>> Map(GRAYSCALE)
>> Encode(HEVC);
Actual C++ statement!
16

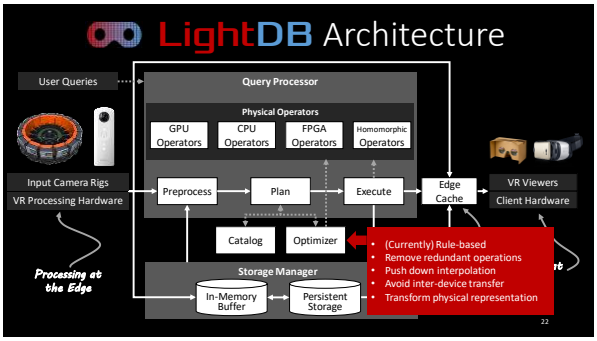

Copyright
Decode(INPUT.MP4)
>> Union(Scan(watermark))
>> Map(GRAYSCALE)
>> Encode(HEVC)
Encode HEVC
Map (Grayscale)
Union
Decode
Scan
Logical Plan
17

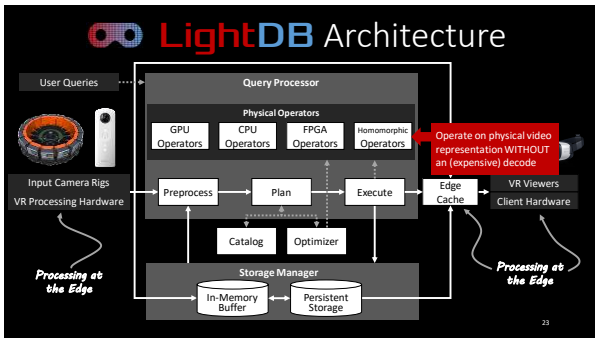

NVENC Encode
CUDA Kernel
CUDA Kernel
NVDEC Decode
NVDEC Decode
input.mp4
watermark/metadata.mp4
Physical Plan
Decode(INPUT.MP4)
>> Union(Scan(watermark))
>> Map(GRAYSCALE)
>> Encode(HEVC)
Encode HEVC
Map (Grayscale)
Union
Decode
Scan
Logical Plan
18

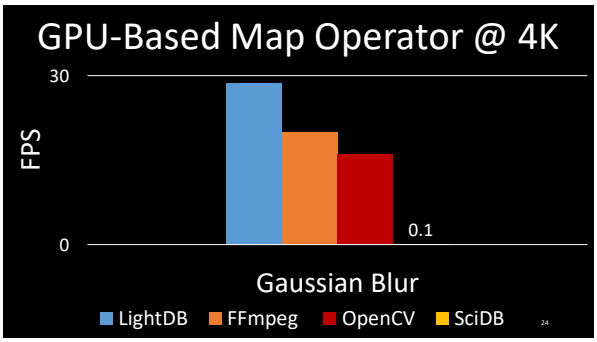

Watermark Query Performance
60
40
20
0
FPS
LightDB
FFmpeg
OpenCV
SciDB

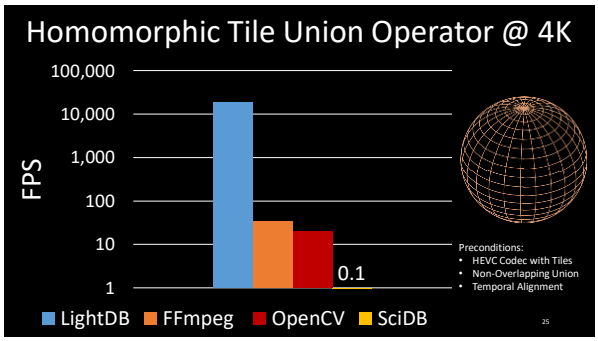

LightDB Architecture
20

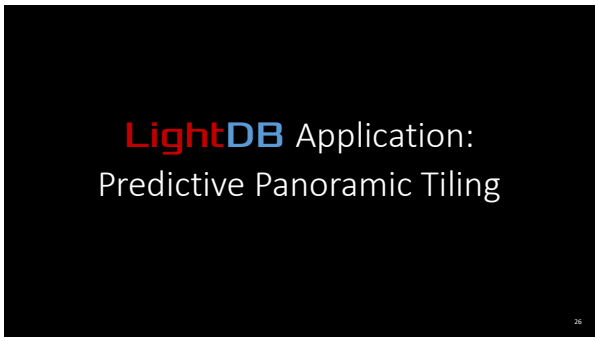

LightDB Architecture
User Queries
Query Processor
Physical Operators
GPU Operators
CPU Operators
FPGA Operators
Homomorphic Operators
Input Camera Rigs
VR Processing Hardware
Preprocess
Plan
Execute
Edge Cache
VR Viewers
Client Hardware
Catalog
Optimizer
Storage Manager
In-Memory Buffer
Persistent Storage
Processing at the Edge
Processing at the Edge
21

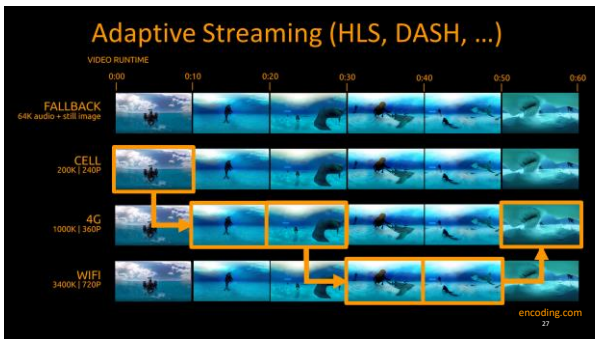

LightDB Architecture
User Queries
Query Processor
Physical Operators
GPU Operators
CPU Operators
FPGA Operators
Homomorphic Operators
Input Camera Rigs
VR Processing Hardware
Preprocess
Plan
Execute
Edge Cache
VR Viewers
Client Hardware
Catalog
Optimizer
(Currently) Rule-based
Remove redundant operations
Push down interpolation
Avoid inter-device transfer
Transform physical representation
Storage Manager
In-Memory Buffer
Persistent Storage
Processing at the Edge
22

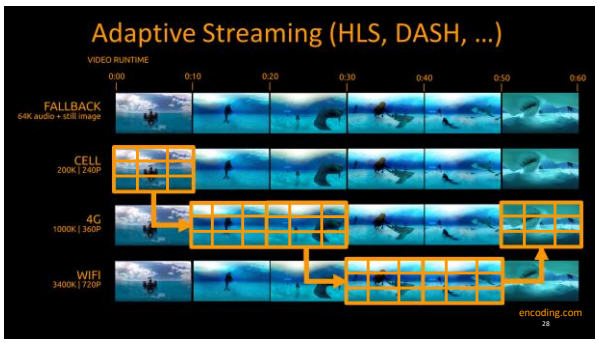

LightDB Architecture
User Queries
Query Processor
Physical Operators
GPU Operators
CPU Operators
FPGA Operators
Homomorphic Operators
Operate on physical video representation WITHOUT an (expensive) decode
Input Camera Rigs
VR Processing Hardware
Preprocess
Plan
Execute
Edge Cache
VR Viewers
Client Hardware
Catalog
Optimizer
Storage Manager
In-Memory Buffer
Persistent Storage
Processing at the Edge
Processing at the Edge
23


GPU-Based Map Operator @ 4K
30
0
FPS
0.1
Gaussian Blur
LightDB
FFmpeg
OpenCV
SciDB
24

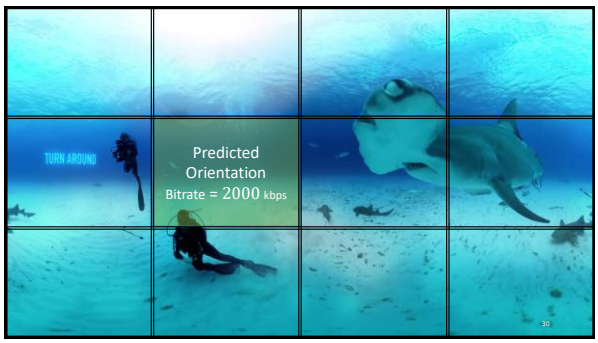

Homomorphic Tile Union Operator @ 4K
100,000
10,000
1,000
100
10
1
FPS
0.1
Preconditions:
• HEVC Codec with Tiles
• Non-Overlapping Union
• Temporal Alignment
LightDB
FFmpeg
OpenCV
SciDB
25

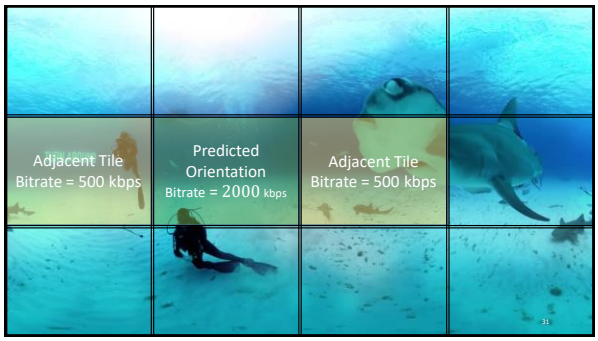

LightDB Application:
Predictive Panoramic Tiling
26

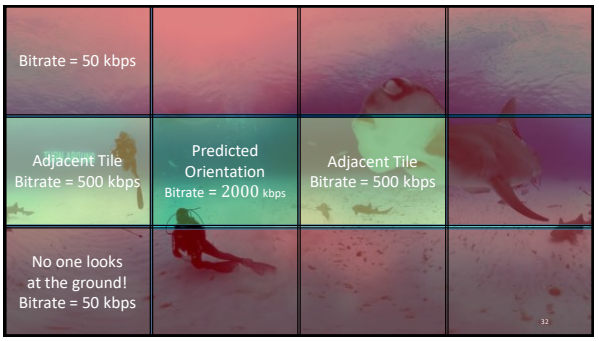

Adaptive Streaming (HLS, DASH, ...)
VIDEO RUNTIME
0:00
0:10
0:20
0:30
0:40
0:50
0:60
FALLBACK
64K audio + still image
CELL
200K | 240P
4G
1000K | 360P
WIFI
3400K | 720P
encoding.com
27


Adaptive Streaming (HLS, DASH, ...)
VIDEO RUNTIME
0:00
0:10
0:20
0:30
0:40
0:50
0:60
FALLBACK
64K audio + still image
CELL
200K | 240P
4G
1000K | 360P
WIFI
3400K | 720P
encoding.com
28

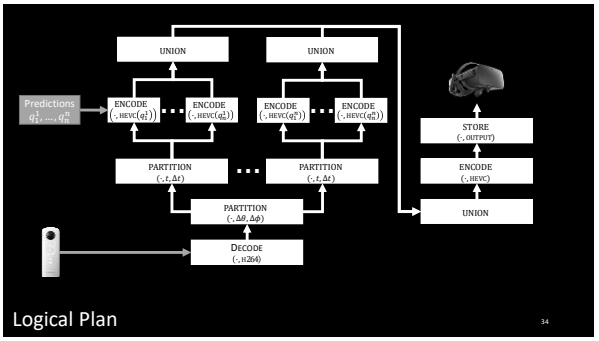

TURN AROUND
29

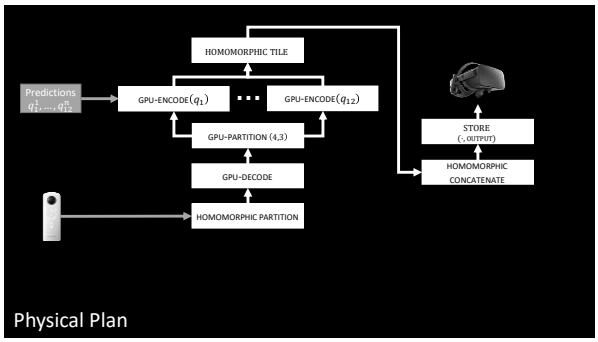

TURN AROUND
Predicted
Orientation
Bitrate = 2000 kbps
30


Adjacent Tile
Bitrate = 500 kbps
Predicted
Orientation
Bitrate = 2000 kbps
Adjacent Tile
Bitrate = 500 kbps
31

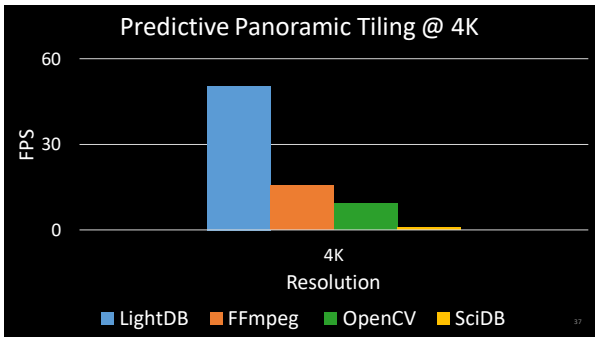

Bitrate = 50 kbps
Adjacent Tile
Bitrate = 500 kbps
Predicted
Orientation
Bitrate = 2000 kbps
Adjacent Tile
Bitrate = 500 kbps
No one looks
at the ground!
Bitrate = 50 kbps
32


Current VR Video Applications
LightDB Queries
Decode(rtp://…)
>> Partition(Time, 1)
>> Partition(Theta, 90°)
>> Partition(Phi, 45°)
>> Transcode(f)
>> Store(output);
Confundo!
33


UNION
UNION
Predictions
ENCODE
ENCODE
ENCODE
ENCODE
PARTITION
PARTITION
PARTITION
DECODE
STORE
ENCODE
UNION
Logical Plan
34


HOMOMORPHIC TILE
Predictions
GPU-ENCODE
GPU-ENCODE
GPU-PARTITION (4,3)
GPU-DECODE
HOMOMORPHIC PARTITION
STORE
HOMOMORPHIC
CONCATENATE
Physical Plan


LightDB
500
20




LightDB

Brandon Haynes
bhaynes@cs.washington.edu

## Key Features:

- Data management system for VR/AR/MR video applications
- Unified data model for panoramic (360°) and light field video
- Declarative queries with automatic optimization
- Full stack: data ingest, processing, and real-time streaming

## Key Results:

- Decreased development complexity ($\sim {}^{1}/_{10}$ LOC)
- Increased performance (up to 4× for real-world workloads)
- Reduced client bandwidth & power requirements

38