High-Quality Immersive Video Streaming via Edge Caching and User Adaptation

Jinlei Jiang
Tsinghua University, China
jjlei@tsinghua.edu.cn

Joint work with: Dongbiao He, Cedric Westphal, Guangwen Yang, JJ Garcia-Luna-Aceves, Teng Ma, Shu-Tao Xia

About Me

Jinlei JIANG

- Associate Professor of Computer Science and Technology, Tsinghua University, P. R. China
- Humboldt Research Fellow (2007-2008)
- Research Interests
  - distributed computing and systems
  - big data storage and computing
  - cloud/edge computing
  - graph computing and database
  - software-defined networking

http://madsys.cs.tsinghua.edu.cn/~jinleijiang/

Outline

- Background & Challenges
- User Behavior Analysis
- Edge Caching & Prefetching
- Evaluation
- Concluding Remarks

Immersive video is popular now!

Immersive video, a.k.a 360-degree or spherical video, can provide users with **immersive and interactive experience** under their own control

Record: 360 camera

View: HMD or Glasses

Education *Games* *Business* …

Wide applications in various domains

$47.7B

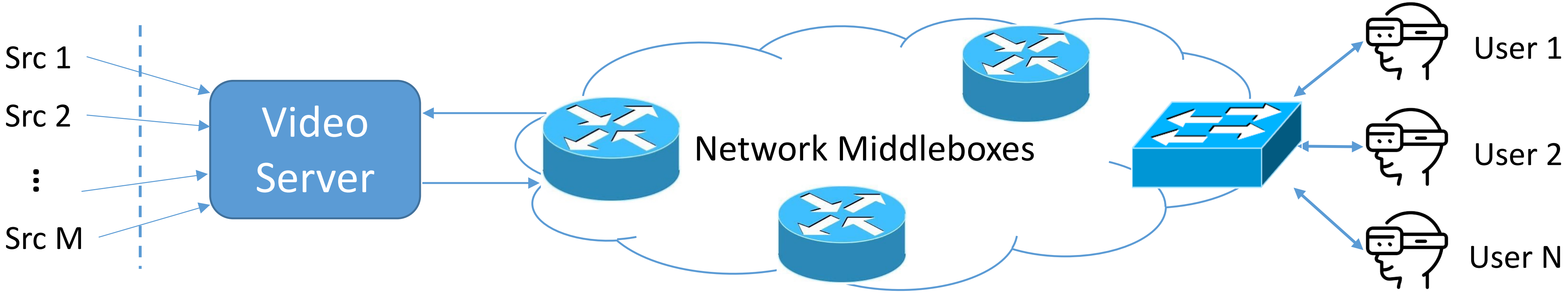
https://www.mordorintelligence.com/industry-reports/virtual-reality-market/

The global market of immersive video streaming would reach by 2024

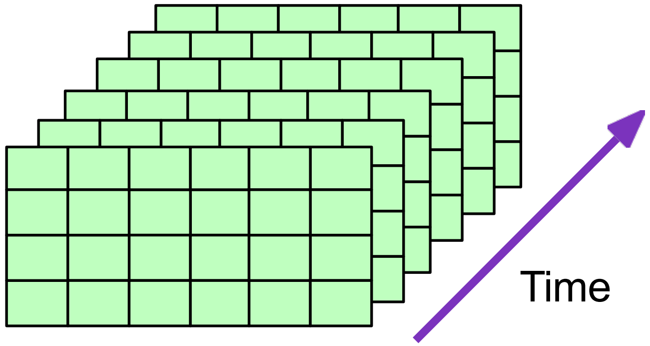
YouTube VR

VR CHAT

An overview of the video streaming system



Images from camera



Video frames at server

Stitched images shown for users

Challenges of streaming immersive videos

large storage need

- Store **multiple views** of each scene for a large variety of client devices
- Keep video **resolution high** for good experience

3GB/minutes in size

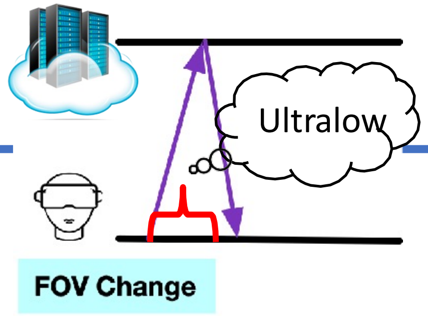
high BW consumption

- At least **4K stream** is needed to transmit a video in full view
- Serve **many users** at the same time

400Mbps
25Mbps (2D 4K video)

ultralow motion-to-photon delay

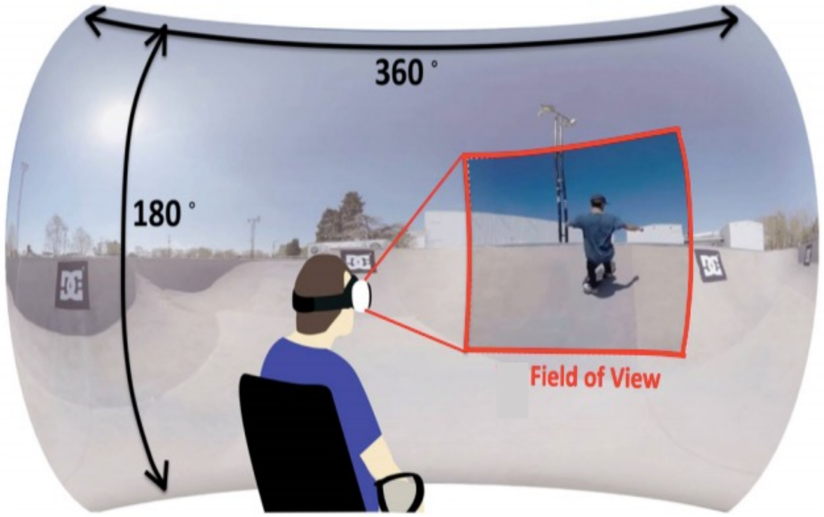
- The new view must be **rendered in very limited time** for good experience



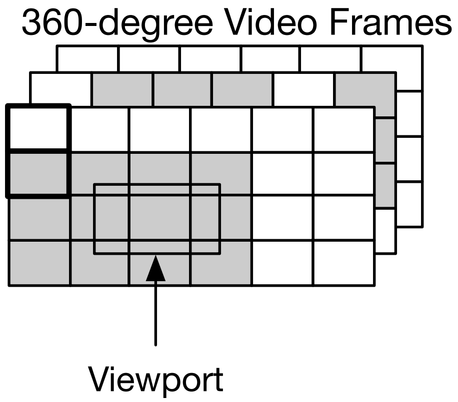
< 10 milliseconds

Refer to MICHAEL ZINK et al., PROCEEDINGS OF THE IEEE, Vol. 107, No. 4, for more!

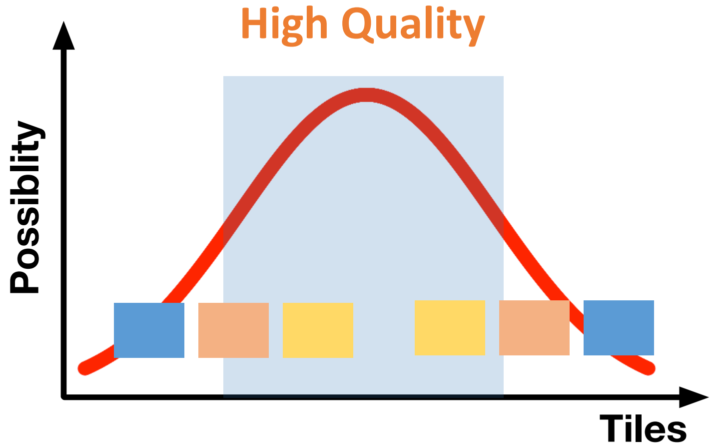
Practice: User/FoV adaptation



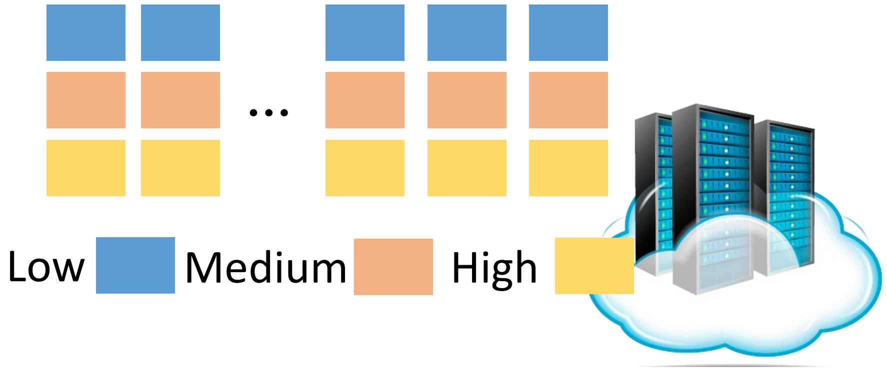
Tiled



FoV: Field of View ←→ Viewport



FOV



Practice: User/FoV adaptation (cont.)

Viewport-Driven Rate-Distortion Optimized 360° Video Streaming

Jacob Chakareski, Ridvan Aksu, Xavier Corbillon, Gwendal Simon, and Viswanathan Swaminathan

Joint Rate and FoV adaptation in immersive video streaming

Dongbiao He
ersity
na
hua.edu.cn

Cedric Westphal
University of California, Santa Cruz
Santa Cruz, CA
cedric@soe.ucsc.edu

J.J. Garcia-Luna-Aceves
University of California, Santa Cruz
Santa Cruz, CA
jj@soe.ucsc.edu

FoV-Aware Edge Caching for Adaptive 360° Video Streaming

Anahita Mahzari, Afshin Taghavi Nasrabadi, Aliehsan Samiei and Ravi Praka
The University of Texas at Dallas

Two-Layer FoV Prediction Model for Viewport Dependent Streaming of 360-Degree Videos

Yiling Xu(✉), Shaowei Xie, Liangji Ma, and Jun Sun

View-Aware Tile-Based Adaptations in 360 Virtual Reality Video Stream

Mohammad Hosseini*

Flare: Practical Viewport-Adaptive 360-Degree Video Streaming for Mobile Devices

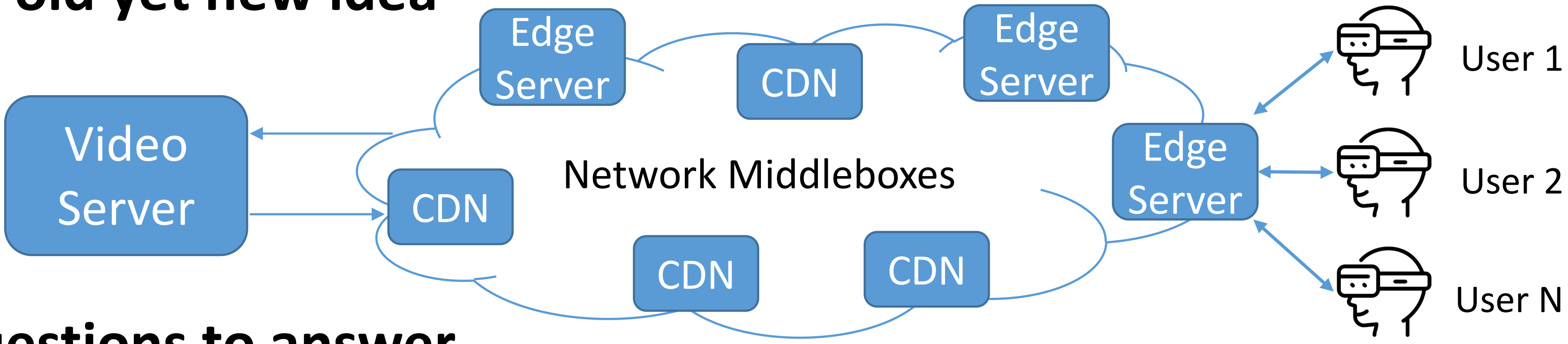
Feng Qian[1]* Bo Han[2] Qingyang Xiao[1] Vijay Gopal
[1]Indiana University [2]AT&T Labs – Research

CUB360: EXPLOITING CROSS-USERS BEHAVIORS FOR VIEWPORT PREDICTION IN 360 VIDEO ADAPTIVE STREAMING

Yixuan Ban[1], Lan Xie[1], Zhimin Xu[1], Xinggong Zhang[1,2,], Zongming Guo[1,2], Yue Wang[3]*

Practice: In-network caching

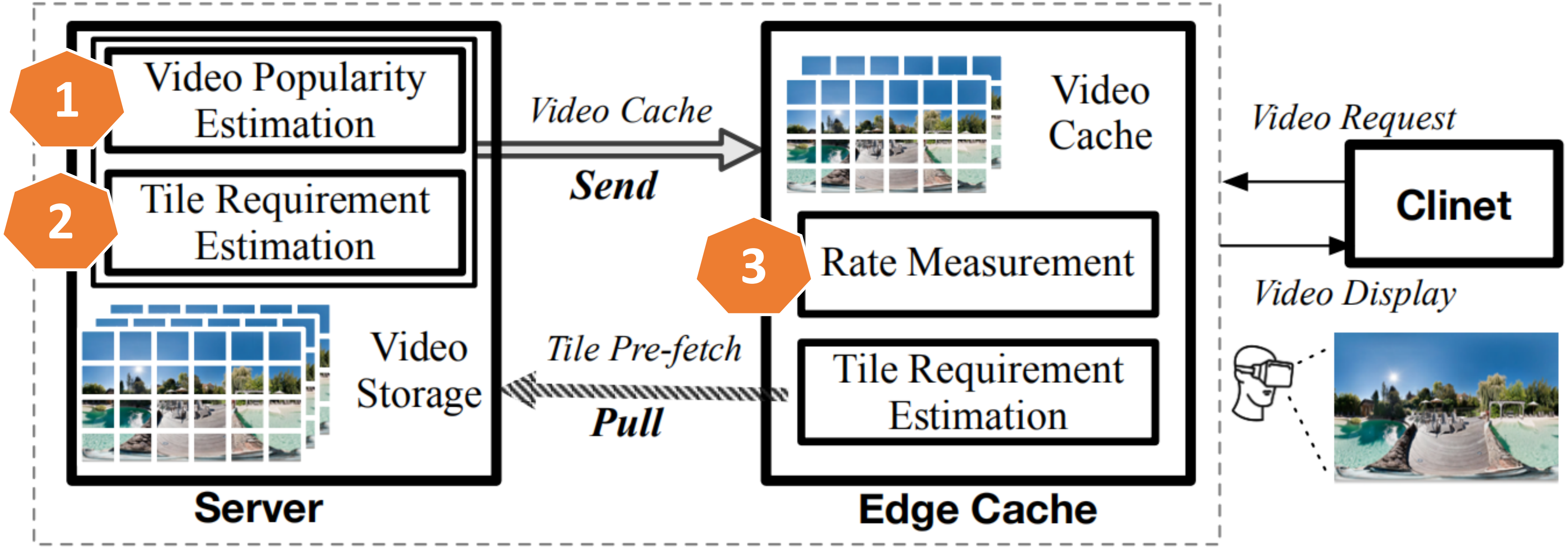
- **An old yet new idea**



- **Questions to answer**
  - **Where** to place the cache & what's the unit (video or tile) for caching
  - **How** to adapt the bitrate according to network condition
- **A lot of work**
  - FoV-aware edge caching (MM'18), tile-based caching (MobiHoc'19), JERTC (MMM'19), Allies (Cloud'20), …

Our solution: CUBIST

CUBIST = Edge Caching (video popularity) + Tile Prefetching (FoV predication)



Outline

- Background & Challenges
- User Behavior Analysis
- Edge Caching & Prefetching
- Evaluation
- Concluding Remarks

Datasets, analysis method and focus

Dataset: Xavier Corbillon et al., MMSys’17

Exploration (Paris)

Static Focus (Rhino)

Rides (Rollercoaster)

Moving Focus (Timelapse)

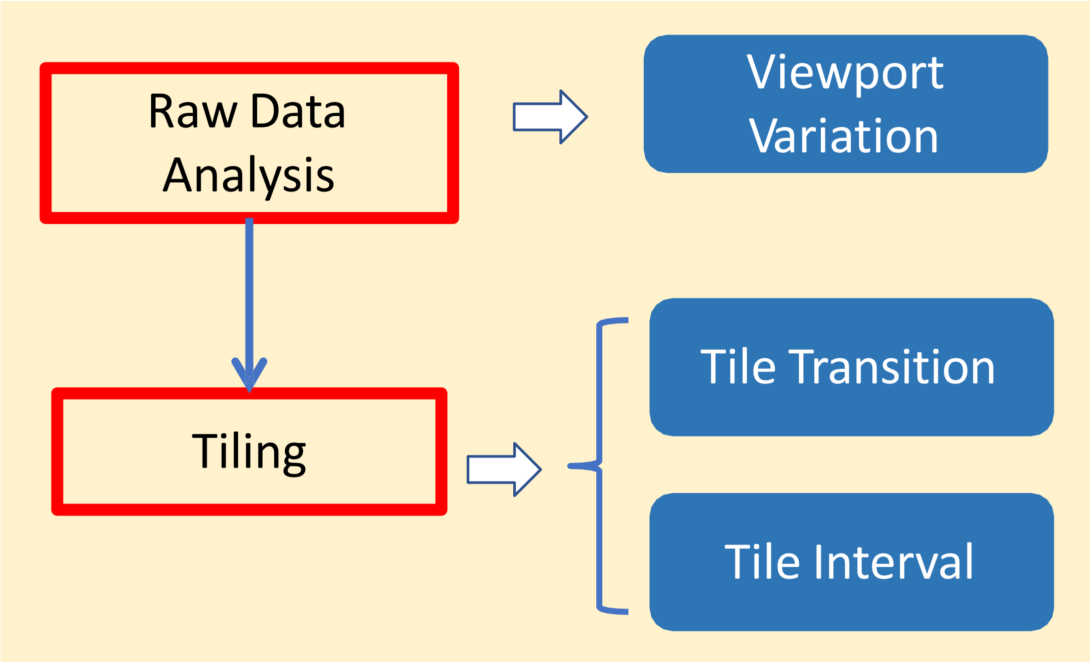
characteristics:

- It collects the user head movement data
- The dataset contains **59 users**
- Multiple kinds of videos: **6 videos**

Method: projection and tiling

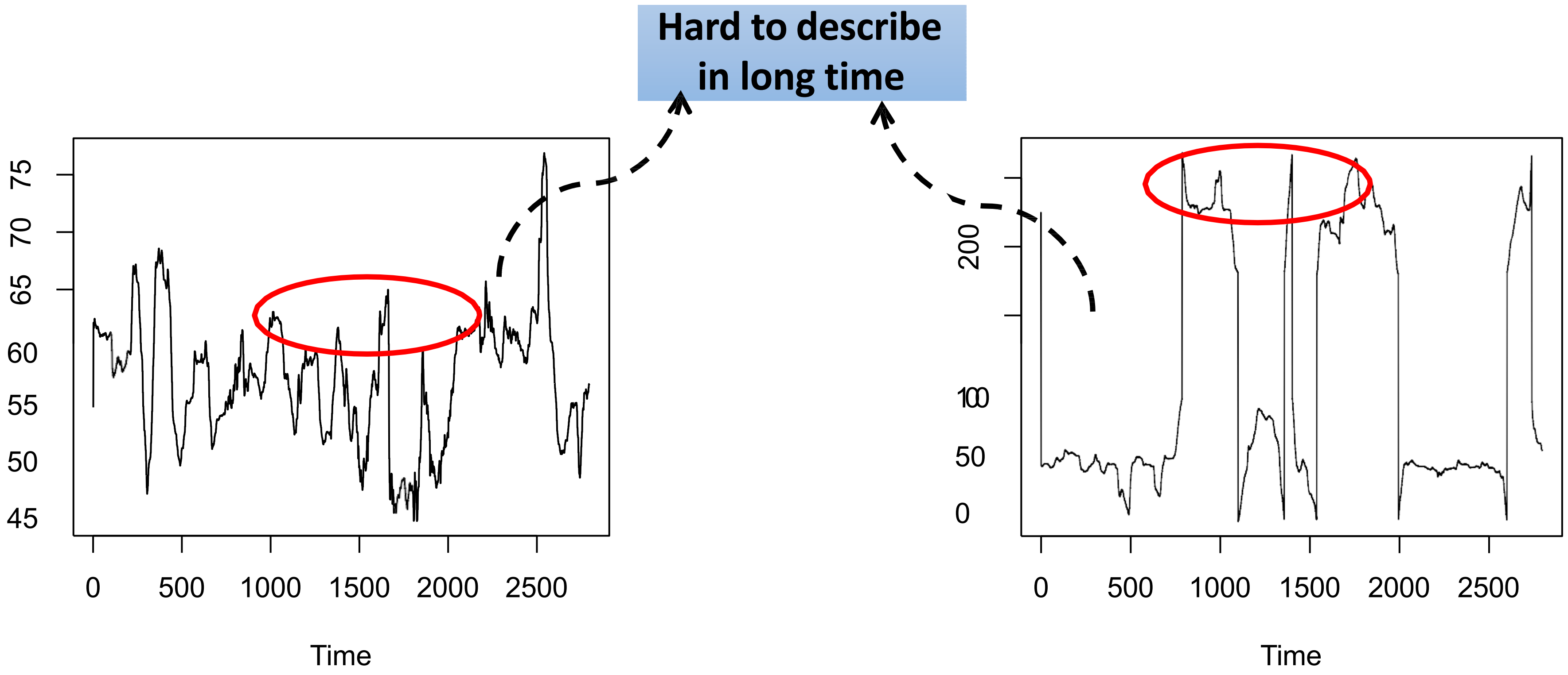
Focus:

- Viewer motion
- head movement



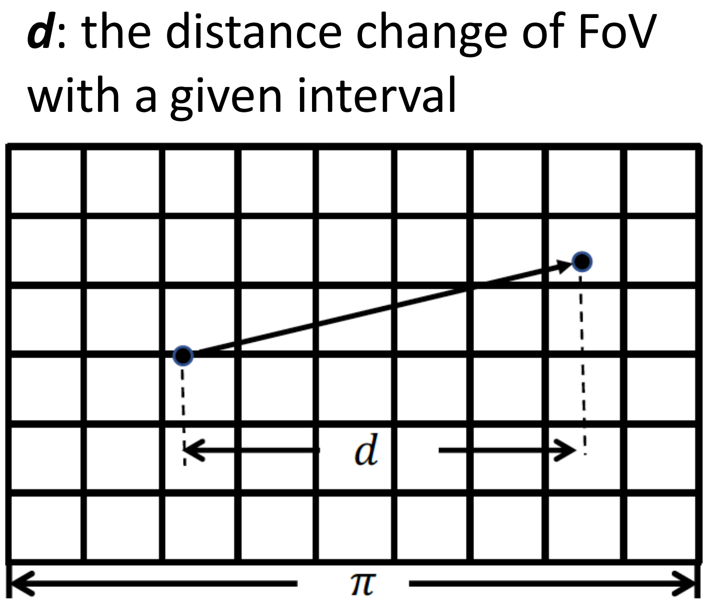
Result: raw data analysis

User's eye position is **hard to predict** especially in long time



Conclusion: it is **not useful** to directly estimate the eye position

Result: viewport is predictable



(1) User moves **shortly** during a given interval:

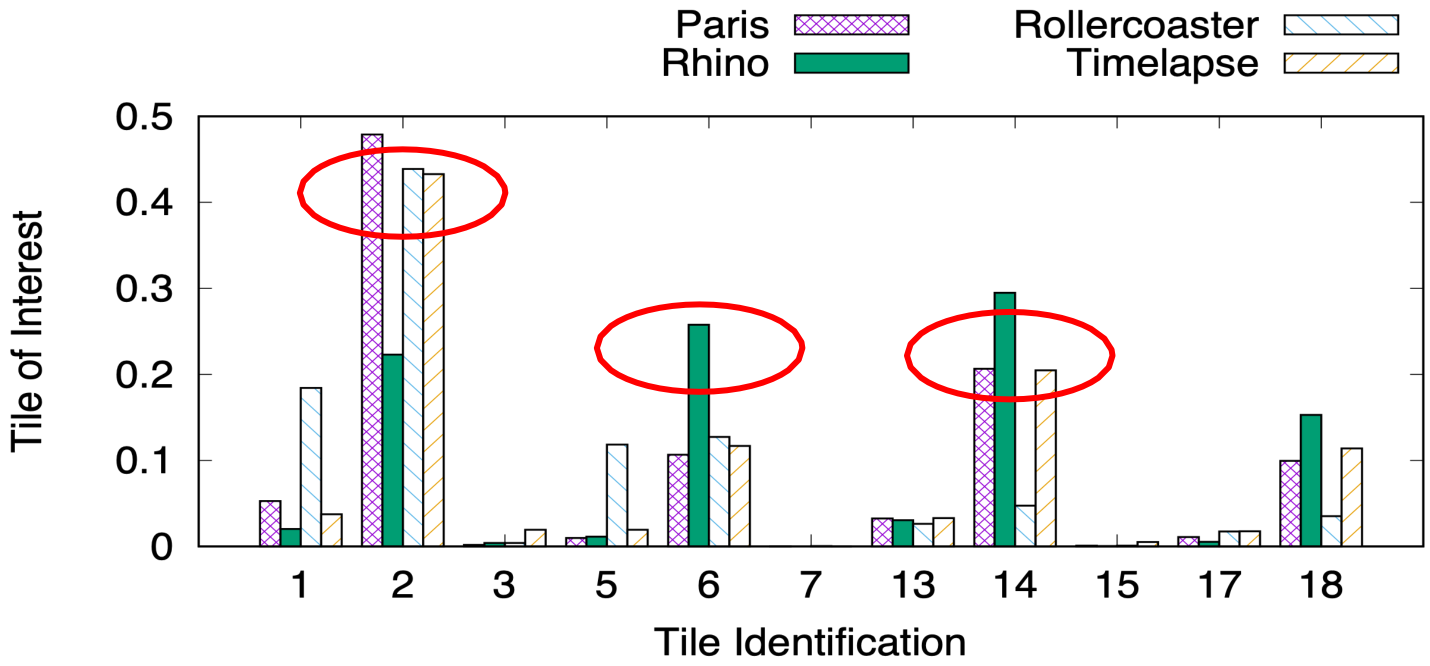
-e.g., 85% of users moves 0.956 unit within 1000ms

(2) Only **part of the view** (FoV) needed by the client

-e.g., uses less than 30.4% of the view in the sphere

	100ms	250ms	500ms	750ms	1000ms
95%	0.147	0.433	3.012	3.093	3.107
90%	0.096	0.255	0.567	1.11	2.983
85%	0.073	0.19	0.401	0.645	0.956

Result: tile request distribution



Tiling	Paris	Rhino	Rollercoaster	Timelapse
6*8	0.35	0.35	0.31	0.40
9*12	0.34	0.32	0.31	0.36
12*12	0.32	0.28	0.31	0.33

Key Findings:

(1) Only a **small portion** of tiles are requested by users;

(2) The tile frequency **varies greatly** inside a video

(3) Most kinds of videos show **the same behavior**, while some other videos are not

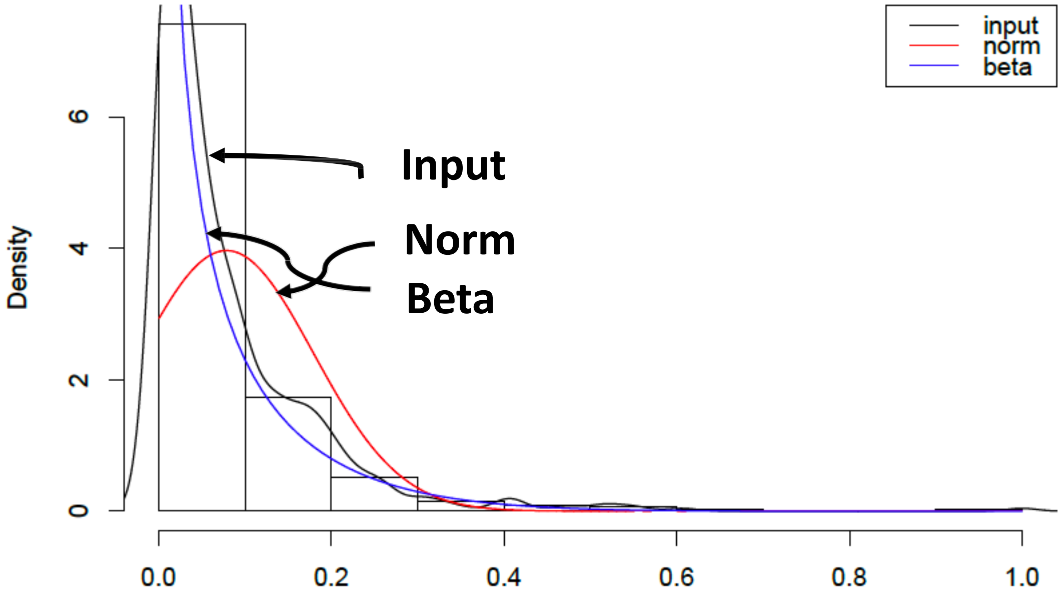
tile frequency: the number of times that a tile is watched in the center of user's FoV, measured with all users on the same video

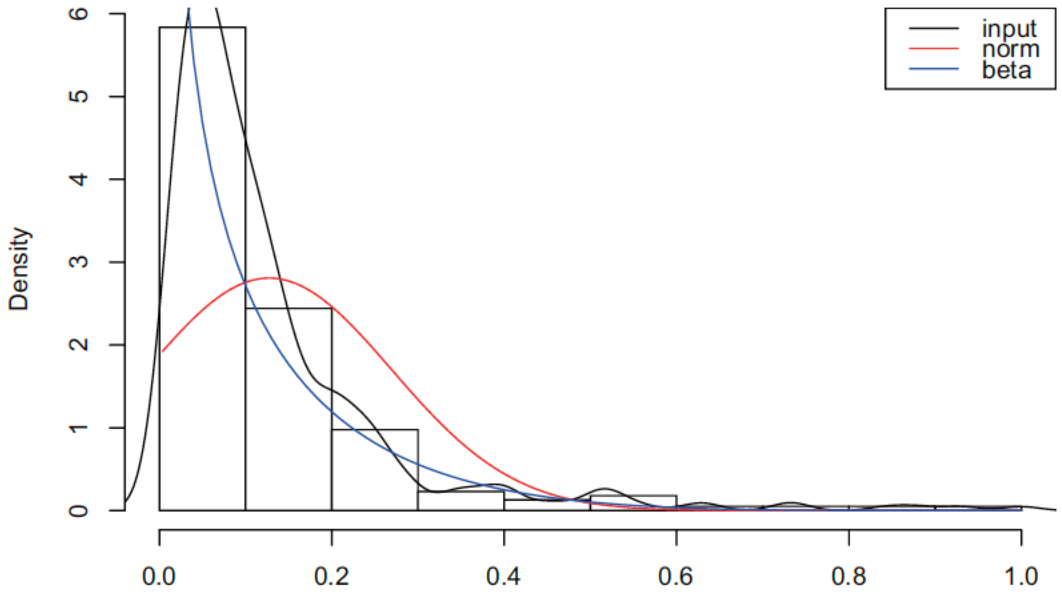
Result: tile interval distribution

Question to answer: **how long** will the user's gaze stay on the same tile?

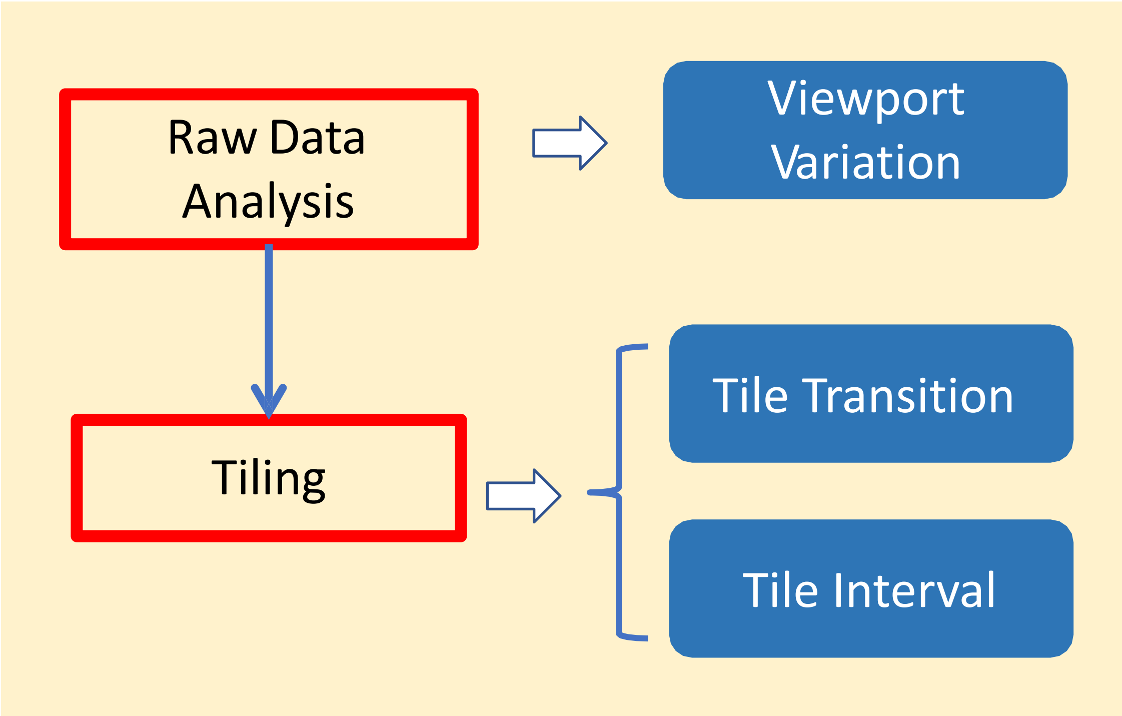
Purpose: to **predict** tile transition

Method: **normalize** the stay time and find the most suitable distribution





Summary about user behavior analysis



(1) User behavior changes **randomly** along the time;

(2) only **half of tiles** will be viewed (at the center of eyes focus);

(3) **Beta distribution** could be used to simulate the tile interval

(4) the behavior of different types of video follows **the same distribution**, but with some variation;

More can be done, please refer to Dongbiao He, Cédric Westphal et al., IFIP Networking 2019 for that

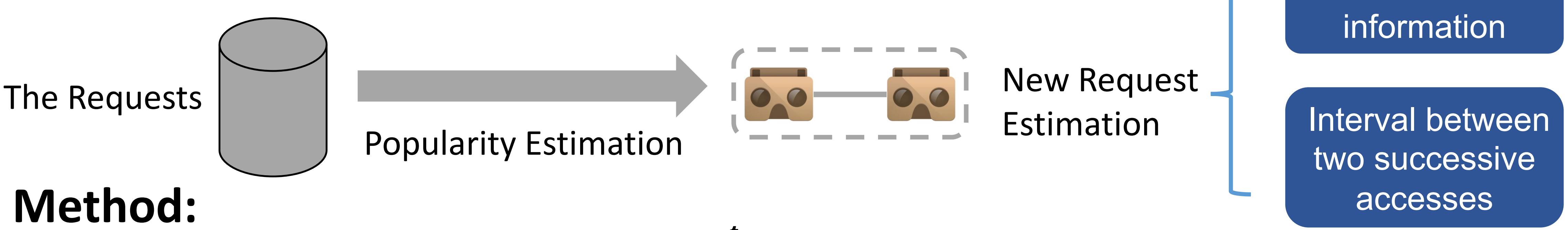
Outline

- Background & Challenges
- User Behavior Analysis
- Edge Caching & Prefetching
- Evaluation
- Concluding Remarks

Video as a unit for popularity estimation

Possible benefits:

- **Easy to implement**, e.g., reuse existing algorithms
- **Reduce jitter** in video quality



Method:

the self-exciting point process

$$\lambda \sum_{t'=1}^{t} n_i(t')\emptyset(t-t')$$

"Frequency"

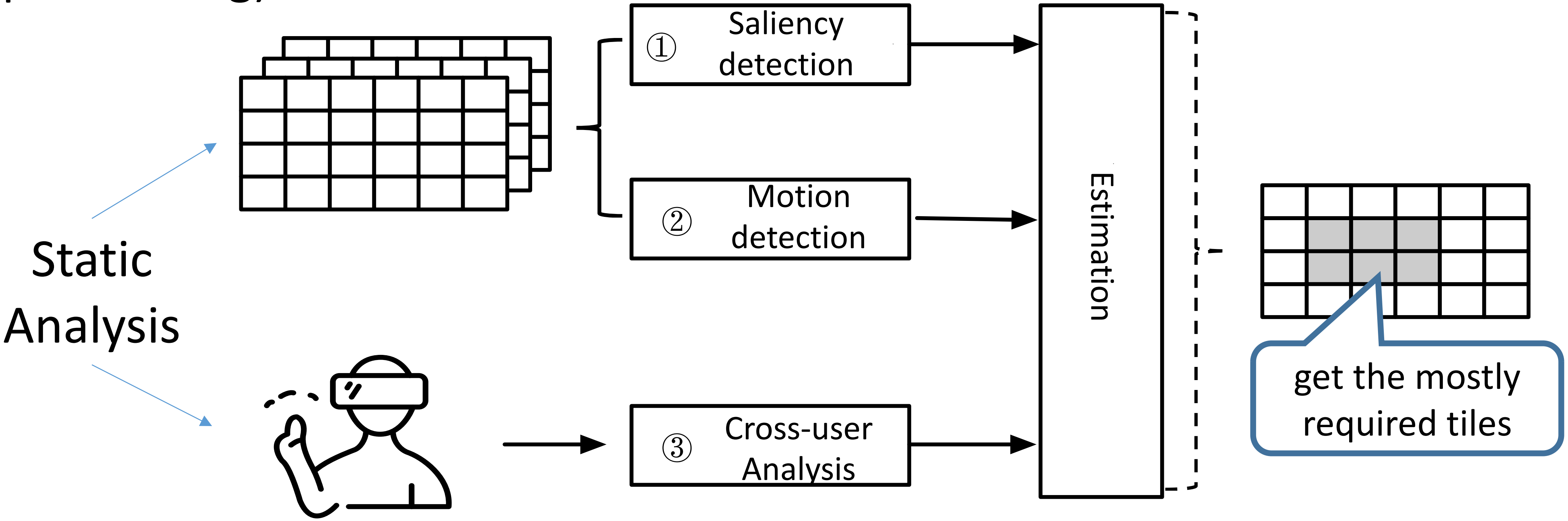
"Recency" *(Kernel Function)*

to reserve the benefit of both LRU and LFU

Tile requirement estimation

Tile as a unit for caching

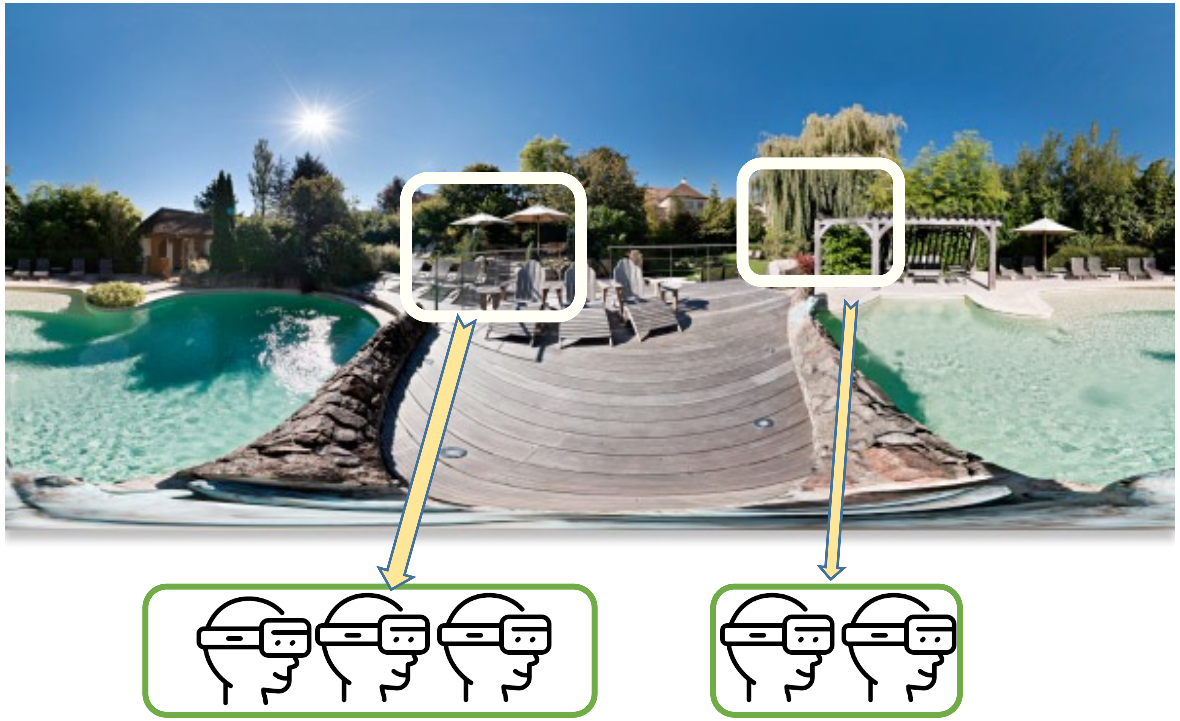
Method: Static Analysis (for caching in advance) + Dynamic Analysis (for prefetching)

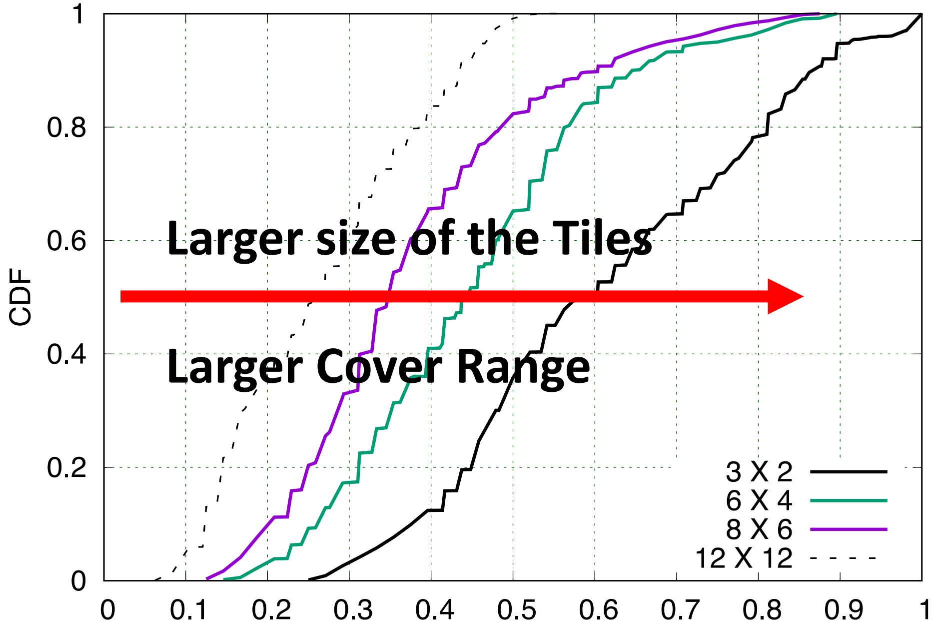


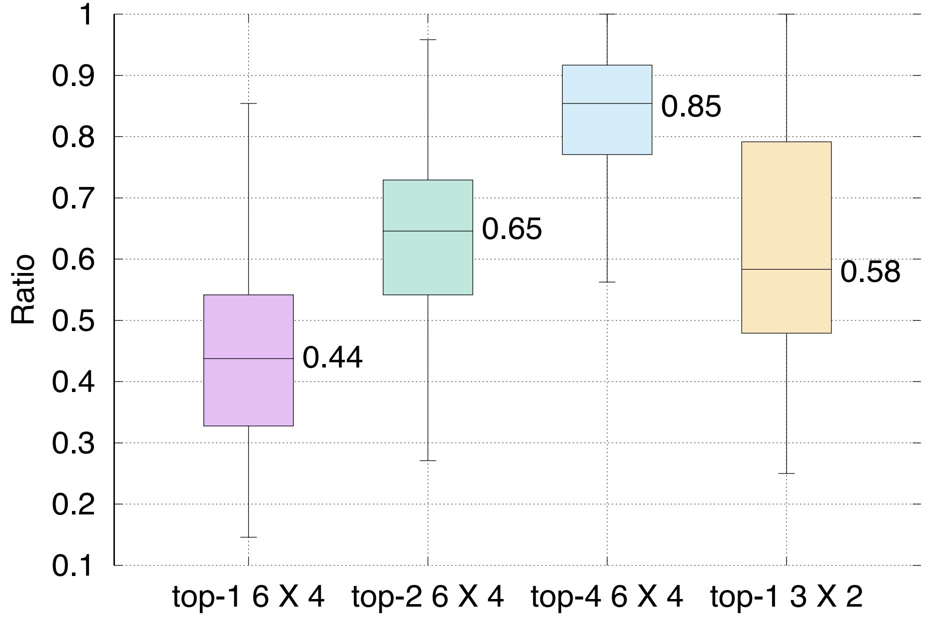
Region Of Interest/ROI

Philosophy: most users focus on some specific regions of the picture => ROI

Param.: #ROIs & Distance between ROIs





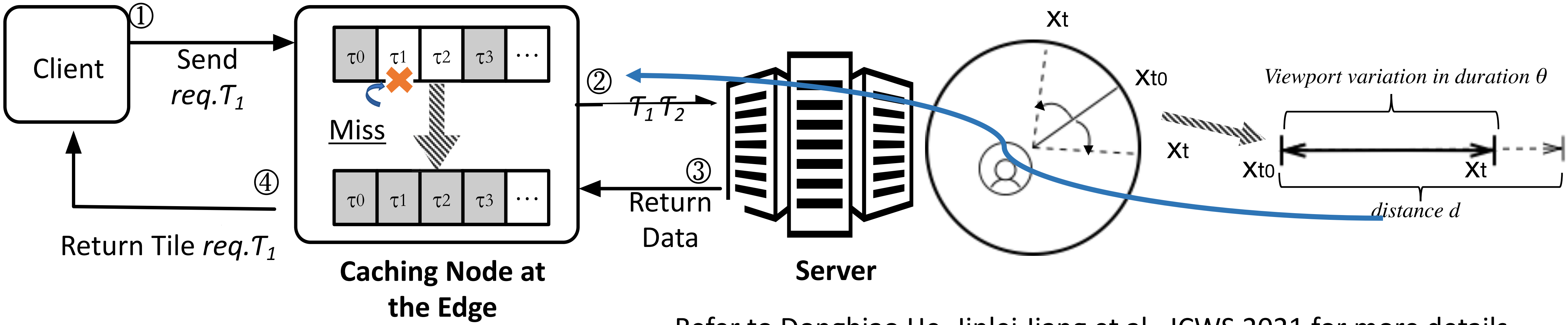


Dynamic analysis

To **prefetch missing tiles** based on

- locality of user movement
- RTT

	100ms	250ms	500ms	750ms	1000ms
95%	0.147	0.433	3.012	3.093	3.107
90%	0.096	0.255	0.567	1.11	2.983
85%	0.073	0.19	0.401	0.645	0.956



Refer to Dongbiao He, Jinlei Jiang et al., ICWS 2021 for more details

Bitrate determination for video caching

Challenge: shared bandwidth

Reactive Caching:

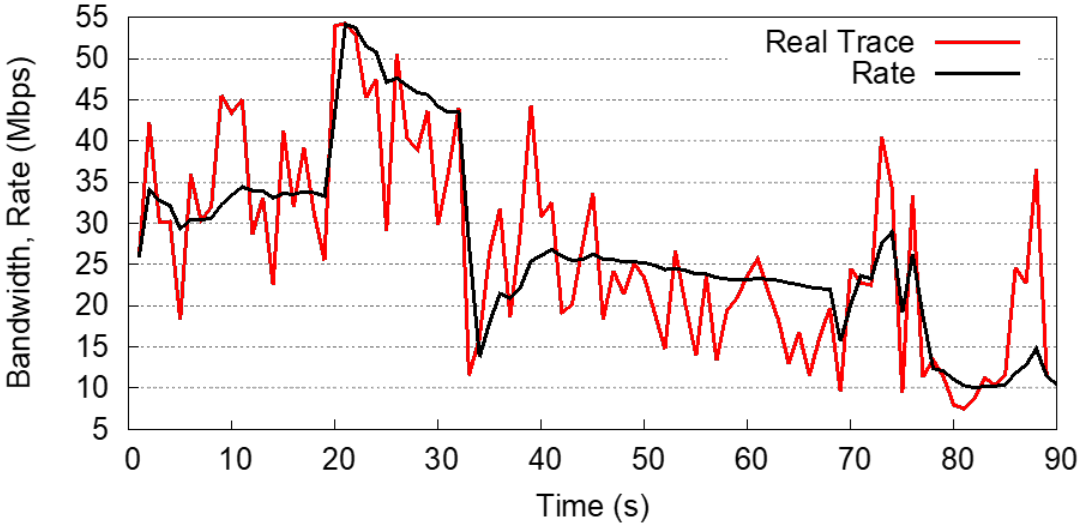
- Choose the video resolution based on "the average available bandwidth over a period"

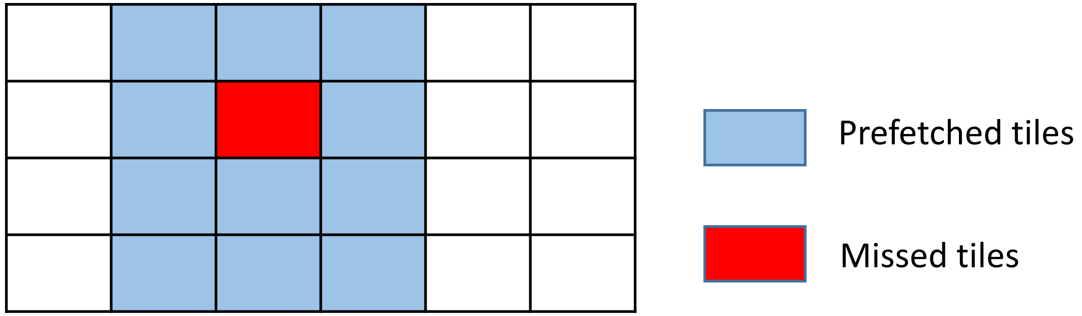
$$\delta(t) = \frac{|Bw_t - Bw_{t-1}|}{\min\{Bw_t, Bw_{t-1}\}}$$

Proactive Prefetching:

- Triggered after a cache miss happens
- Predict and prefetch tiles (identified by id) to be accessed soon but not in the cache yet
- Adapt to the real-time end-to-end delay

$$bitrates(\tau) = \frac{Bw \times \tau.t}{|d| * |\pi|}$$



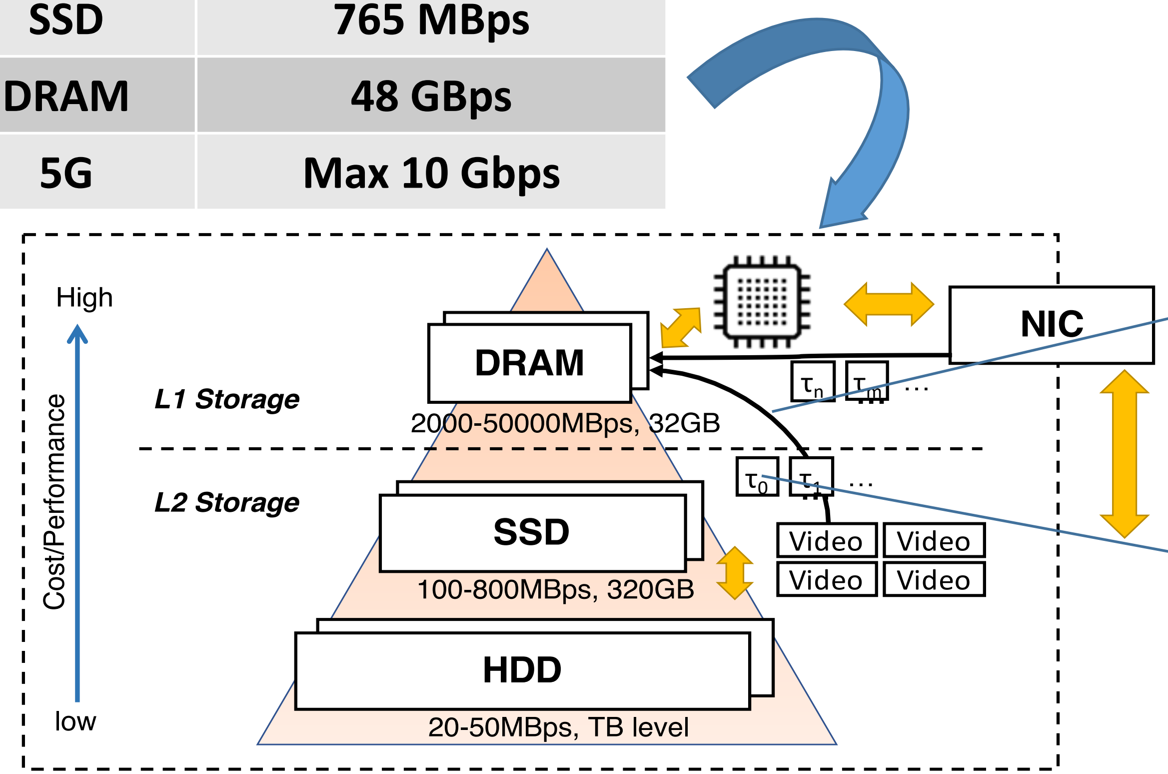


CUBIST implementation

➢ **The storage would be the bottleneck**

Device	Throughput
HDD	36.8 MBps
SSD	765 MBps
DRAM	48 GBps
5G	Max 10 Gbps

✓ **Hierarchical cache**



➢ **Place items with the caching reward**

- r_i : the popularity of the video
- $T(u, s_i)$ and $T(u, cache)$: the cost to get the segment
- $\tau.f$: the ratio τ is accessed;

$$G_i(\tau) = r_i \sum_{\tau \in i} (size(\tau) \times \tau.f[\mathcal{T}(u, s_i) - \mathcal{T}(u, cache)])$$

- L1 has enough space: assign *τ* a lifetime and put it into L1;

- Move the last ranked tiles to L2

Outline

- Background & Challenges
- User Behavior Analysis
- Edge Caching & Prefetching
- Evaluation
- Concluding Remarks

CUBIST evaluation: settings

- **Datasets**
  - 25 videos
  - 109 users

- *X. Corbillon, F. De Simone, and G. Simon. 360-degree video head movement dataset.*
- *C. Wu, Z. Tan, Z. Wang, and S. Yang. A dataset for exploring user behaviors in VR spherical video streaming.*

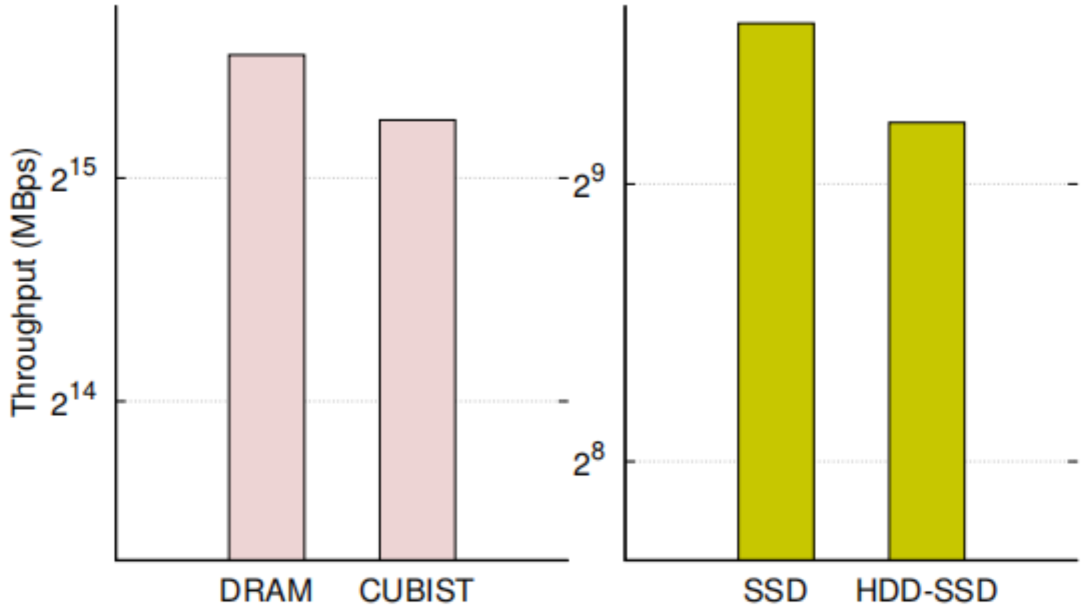
- **Requests & Bandwidth**
  - User requests: GlobeTraff
  - Bandwidth variation: 4G Trace

- *J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck. HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks.*

- **Benchmarks**
  - Video Cache, CUBIST-NP
  - Tile Cache

- *A. Mahzari, A. T. Nasrabadi, A. Samiei, and R. Prakash. Fov-aware edge caching for adaptive 360° video streaming. MM 2018*

Evaluation: benefit of hierarchical cache

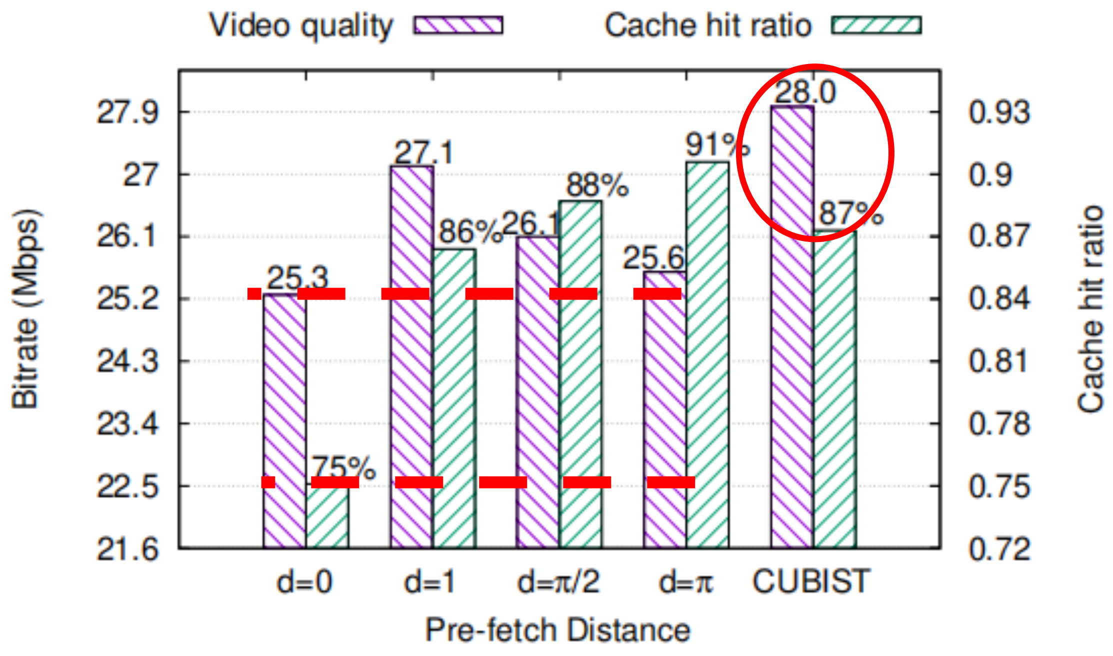


1) the cache space is **20%** of the total video size
2) the ratio of L1 to L2 cache is **3:2**
3) he ratio of L1 to L2 hit varies between **9:1** and **7:3** randomly

CUBIST improves the throughput from **765MBps** to **39GBps**

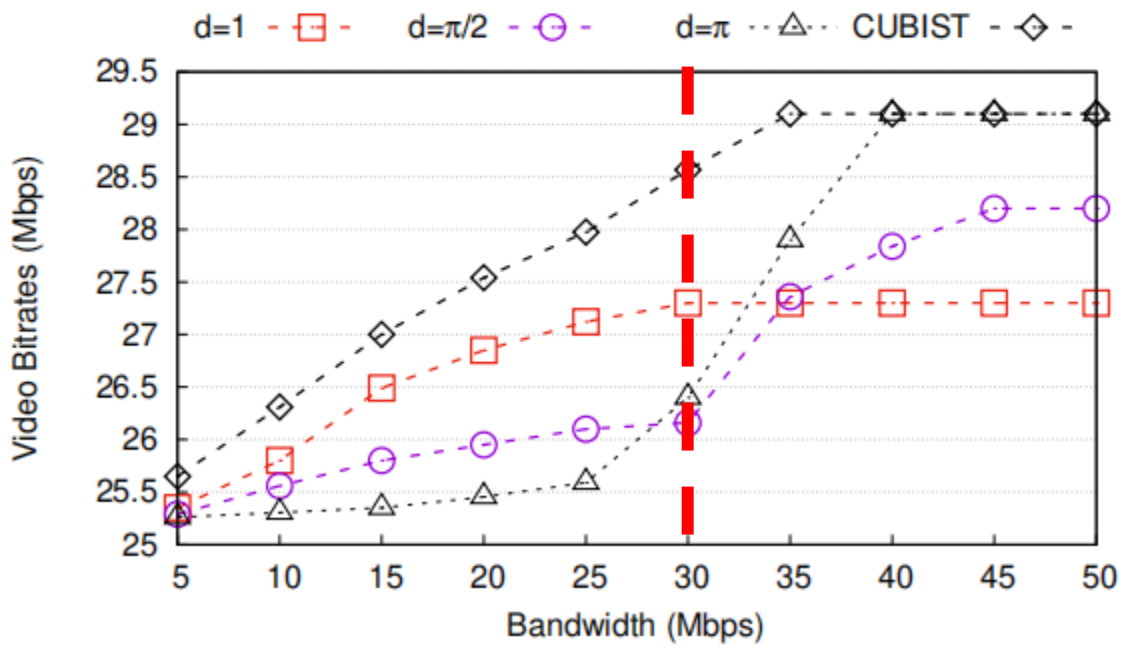
Highlight: hierarchical cache design with fixed cache cost means larger cache space and higher cache hit ratio, which would bring in more benefit.

Evaluation: benefit of prefetching



(a) video bitrate & cache hit ratio

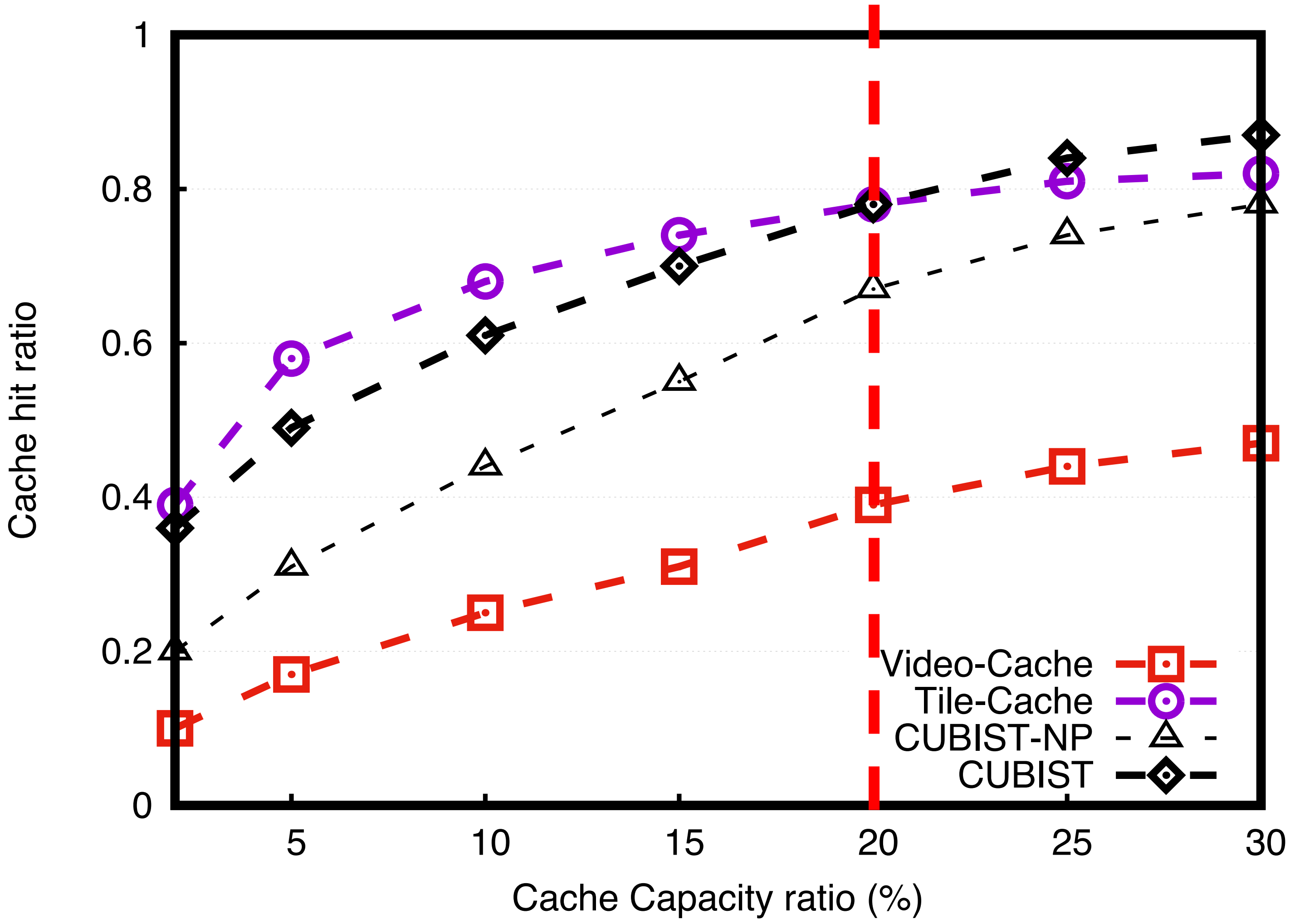
video quality and cache hit ratio are balanced



(b) Prefetch performance vs. bandwidth

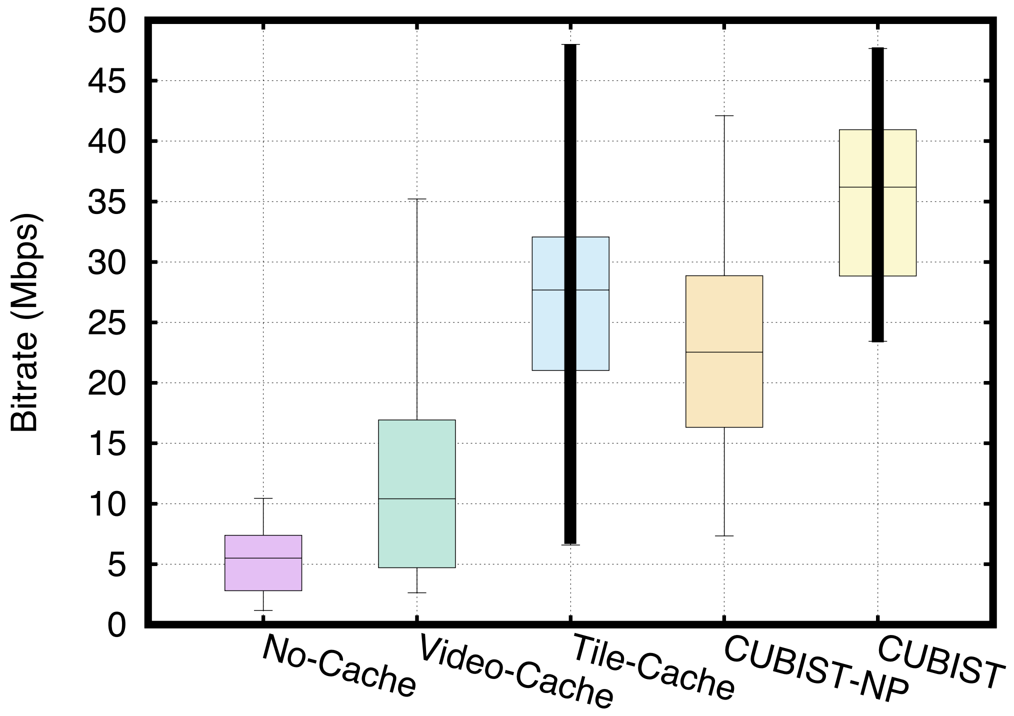
utilize network resources better

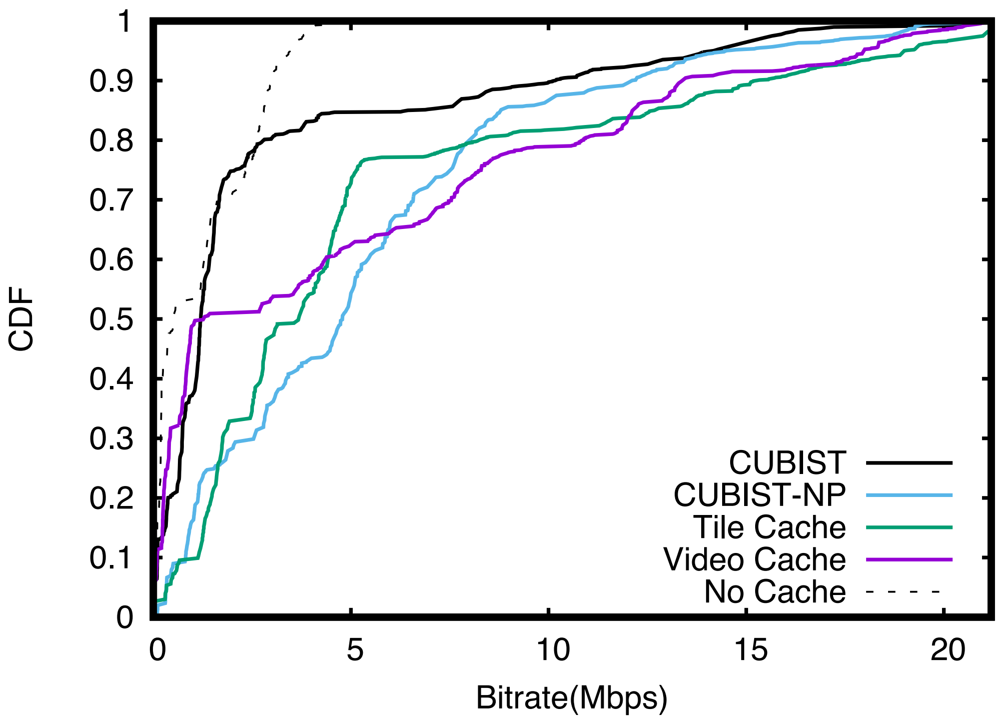
Evaluation: benefit of caching



- ✓ Tile Prefetching gets **10%** more gains for caching
- ✓ CUBIST costs **20%** less caching space than Tile Cache

Evaluation: QoE of videos





- ✓ Compared with Tile Cache, CUBIST only needs **half** of the video transitions
- ✓ CUBIST outperforms Tile Cache, whose median bitrate is **26.9Mbps**, by **12.9%** in video bitrate

Outline

- Background & Challenges
- User Behavior Analysis
- Edge Caching & Prefetching
- Evaluation
- Concluding Remarks

Conclusions

Immersive video streaming is challenging

- Ultrahigh bandwidth requirement
- Ultralarge Storage Requirements
- Ultralow Motion-to-Photon Delay

CUBIST employs edge caching to solve the problem

- Video-based popularity estimation → simplified implementation
- Proactive tile prefetching → more cache hit
- Hierarchical cache organization → reduced cache node cost
- Bitrate determination: Clients <-> Edge Nodes <-> Servers → better QoE

Limitations and future work

Limitations

- Not applicable to live immersive video streaming
- No consideration of joint caching at multiple edge servers

Future work

- More effective algorithms for tile caching and prefetching, possibly via machine learning
- Coordinated caching at multiple edge servers
- More efficient video coding scheme for transmission
- In-network quality enhancement or even tile generation

Thanks for your attention!