

基于Fov的全景视频缓存 ---- 论文学习汇报

CUC - 19 - 数媒技 - 杨雪婷

2021.01.22

汇报内容介绍

1. 整体介绍
2. 算法和模型分析
3. 评价指标&实验结果
4. 实验环境&实验数据

Part One : 整体介绍

(1) 目前困难:

- 减少宽带消耗
- 解决网络延迟

(2) 现有解决方法:

现有**减少宽带消耗**的方法 (FoV-adaptive 360°视频流) :

- 用户视角FOV内发送高分辨率的贴图
- 其他贴图发送低分辨率或不发送缓存请求

但是没有解决**网络延迟**

(3) 研究思路:

缓存**靠近**终端用户的**流行内容**(Caching popular content close to the end users)

-----> 同时解决网络延迟和宽带需求。

- 靠近：启用**基站**作为缓存模块来实现(BSs)位于RAN(无线接入网)边缘。
- 流行内容：预测下一次的Fov, **提高命中率**

(4) 研究结果

FoV-aware caching policy:

a probabilistic model based on previous users' watching histories

作用：当缓存容量超过时，缓存模块利用这个策略来确定清除块的优先级。

Part Two : 算法和模式

整体系统

1. 网络模式

2. 360°视频模式

网络：

- 如果缓存有请求的内容，直接调用；
- 如果没有缓存，先从CDN上调用下载，然后再给客户端。（增加延迟 & 带宽消耗）

请求：

目标：

满足当前网络吞吐量的情况下，为用户实际的FoV内提供尽可能高的质量。

- 1. 网络通畅：Fov内的高质量缓存，其他低质量**
- 2. 网络阻塞：全部视频块都低质量缓存**

缓存策略的算法

(1) 得到缓存策略需要的参数

1. Users' FoV Pattern

2. Impact of Video's Bitrate on FoV's Quality

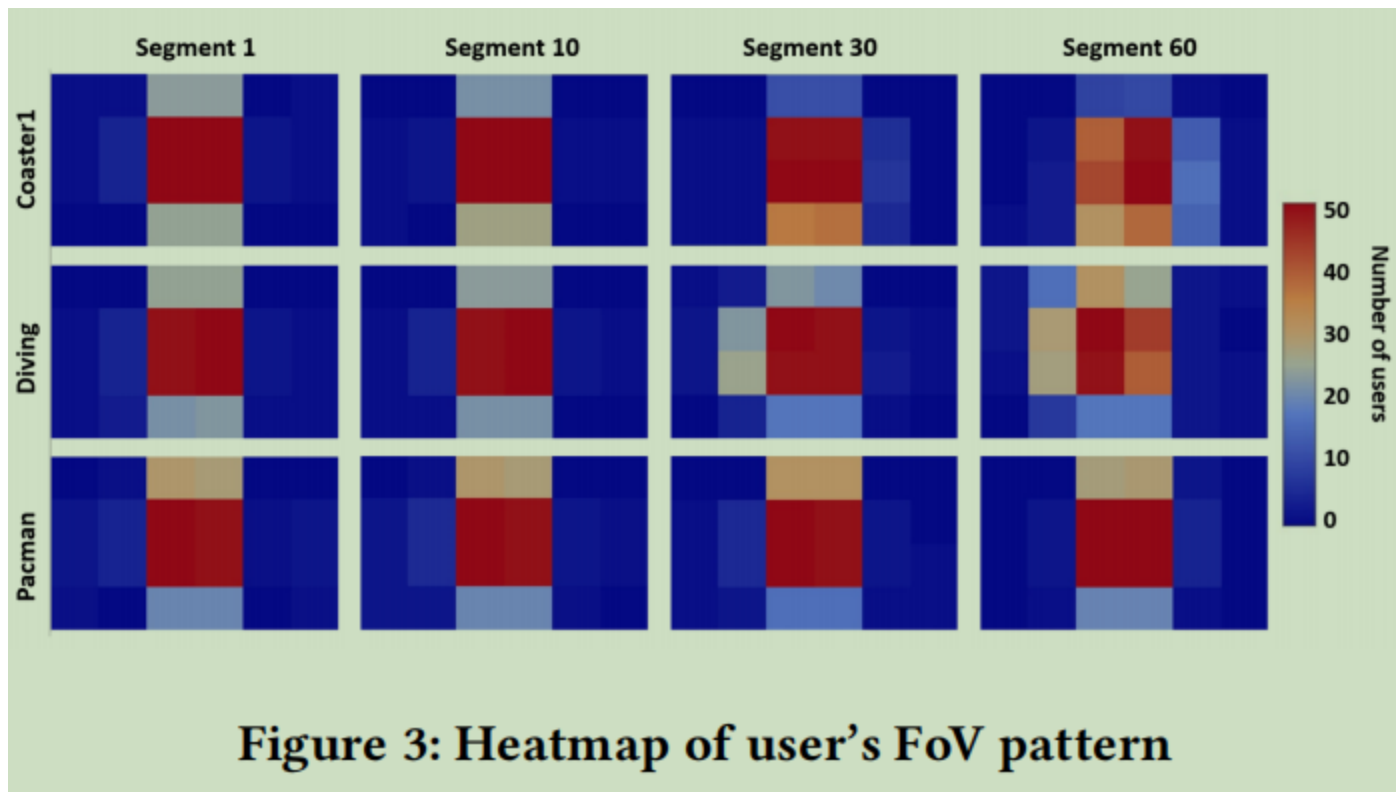
实验前期准备

对于Users' FoV Pattern -- 确定common-Fov

已知：用户的Fov和内容关系极大，赛车-->正前方，观光游览-->360°

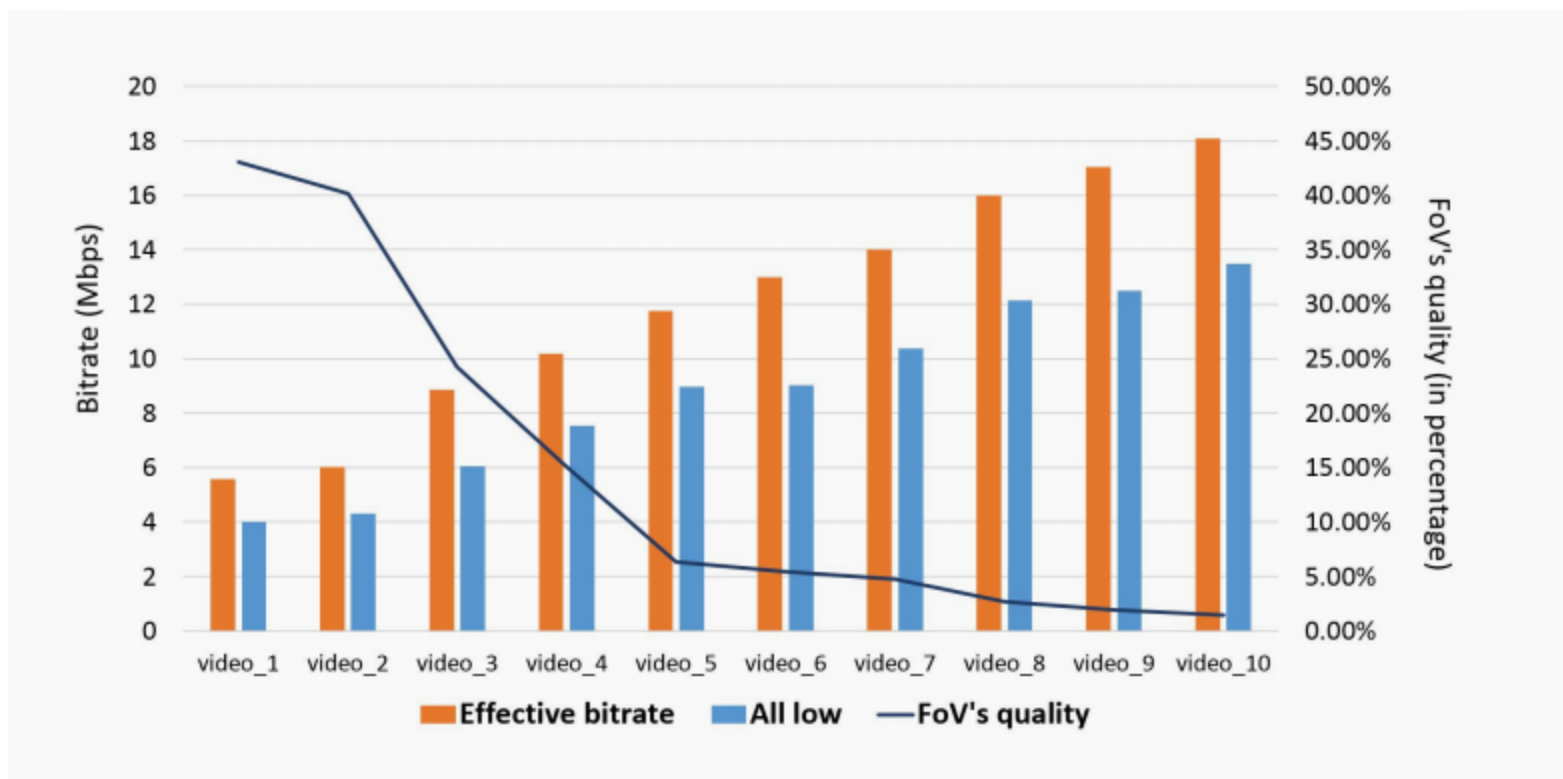
实验：收集真实轨迹的头部跟踪数据集，并显示用户在虚拟环境有某些类似的观看模式

数据集：10短视频，每段50人观看，计算FOV的热力图。



对于Bitrate to FoV's Quality

已知：视频由比特率编码<--视频内容的纹理和运动,也就是说场景越复杂，需要更高的比特率来维持相同水平。



(2) FoV-aware Caching Policy

Learning Parameters

- 参数 Q_{vsi} : 视频块 i 在common-Fov中出现的概率

训练数据: 对于第 i 块的请求 (信息为: 第 i 块是否在请求的Fov中)

- 参数 F_i : 用户请求视频块 i 为高质量的概率

训练数据: 对于Fov中第 i 块的请求 (信息为: 是否需要高质量)

描述参数的模型的选择:

- 因为都分别**只有一个变量影响**，选择**朴素贝叶斯算法**来构造每个参数的概率模型。
- 最大似然估计（MLE），使得 Q_{vsi} 和 F_i 最接近数据集中两者的公式。

建立:

$$\begin{aligned}\theta_{v,s,i} &= \arg \max_{\theta} P(\mathcal{D}_{\theta_{v,s,i}} | \theta) \\ &= \arg \max_{\theta} \ln(\theta_{v,s,i}^{\alpha_{inFoV_{v,s,i}}} * (1 - \theta_{v,s,i})^{\alpha_{outOfFoV_{v,s,i}}}) \\ \psi_v &= \arg \max_{\psi} P(\mathcal{D}_{\psi_v} | \psi) \\ &= \arg \max_{\psi} \ln(\psi_v^{\beta_{high_v}} * (1 - \psi_v)^{\beta_{low_v}})\end{aligned}$$

为了使得上述方程最大化 (Max) , 我们对方程求导: 以第一个方程为例

$$\begin{aligned} \frac{d}{d\theta} [\ln(\theta_{v,s,i}^{\alpha_{inFoV_{v,s,i}}} * (1 - \theta_{v,s,i})^{\alpha_{outOfFoV_{v,s,i}}})] &= \\ \frac{d}{d\theta} [\alpha_{inFoV_{v,s,i}} \ln(\theta_{v,s,i}) + \alpha_{outOfFoV_{v,s,i}} \ln(1 - \theta_{v,s,i})] &= \\ \alpha_{inFoV_{v,s,i}} \frac{d}{d\theta} \ln(\theta_{v,s,i}) + \alpha_{outOfFoV_{v,s,i}} \frac{d}{d\theta} \ln(1 - \theta_{v,s,i}) &= \\ \frac{\alpha_{inFoV_{v,s,i}}}{\theta_{v,s,i}} - \frac{\alpha_{outOfFoV_{v,s,i}}}{1 - \theta_{v,s,i}} &= 0 \end{aligned}$$

最终得到了两个参数的模型:

$$\begin{aligned} \theta_{v,s,i} &= \frac{\alpha_{inFoV_{v,s,i}}}{\alpha_{inFoV_{v,s,i}} + \alpha_{outOfFoV_{v,s,i}}} \\ \psi_v &= \frac{\beta_{high_v}}{\beta_{high_v} + \beta_{low_v}} \end{aligned}$$

(3) 建立更新参数模型

学习更新 Q_{vsi} 和 F_i 得到 R （第 i 视频块在将来被以 q 的质量请求的概率）

- 对于低质量版本的视频块有两种可能被请求：
 - i. 不在Fov内
 - ii. 在Fov内但是用户请求为低质量
- 对于高质量的视频块请求只有一种可能：在Fov内且高质量请求。

$$\gamma_{v,s,i,q} = \begin{cases} \theta_{v,s,i} * \psi_v, & \text{if } q = high \\ (1 - \theta_{v,s,i}) + \theta_{v,s,i} * (1 - \psi_v), & \text{if } q = low \end{cases}$$

因此我们可以通过以下的update_parameters函数得到R:

```
Function update_parameters( $t_{v,s,i}, q$ ):  
  if is_in_FoV( $t_{v,s,i}, q$ ) then  
    |  $\alpha_{\text{inFoV}_{v,s,i}} += 1$   
    | if  $q == \text{high}$  then  
    | |  $\beta_{\text{high}_v} += 1$   
    | else  
    | |  $\beta_{\text{low}_v} += 1$   
    | end  
  else  
    |  $\alpha_{\text{outOfFoV}_{v,s,i}} += 1$   
  end  
   $\theta_{v,s,i} = \alpha_{\text{inFoV}_{v,s,i}} / (\alpha_{\text{inFoV}_{v,s,i}} + \alpha_{\text{outOfFoV}_{v,s,i}})$   
   $\psi_v = \beta_{\text{high}_v} / (\beta_{\text{high}_v} + \beta_{\text{low}_v})$   
  if  $q == \text{high}$  then  
    |  $\gamma_{v,s,i,q} = \theta_{v,s,i} * \psi_v$   
  else  
    |  $\gamma_{v,s,i,q} = (1 - \theta_{v,s,i}) + \theta_{v,s,i} * (1 - \psi_v)$   
  end  
  return  $\gamma_{v,s,i,q}$ 
```

(4) 建立 *Caching Policy*

目标：描述缓存模块在接收到质量为 q 的视频块 i 的请求时所采取的动作。

```
Function upon_request( $t_v, s, i, q$ ):  
   $\gamma_{v,s,i,q} \leftarrow \text{update\_parameters}(t_v, s, i, q)$   
  if  $\text{cache.contains}(t_v, s, i, q)$  then  
     $\text{tile} \leftarrow \text{cache.get}(t_v, s, i, q)$   
    send(tile)  
  else  
    // downloading tile from remote content server  
     $\text{tile} \leftarrow \text{download}(t_v, s, i, q)$   
    // adding tile to cache  
     $\text{cache.add}(\text{tile}, \text{key}=\gamma_{v,s,i,q})$   
    // sending tile to client  
    send(tile)  
    while  $\text{cache.size()} > \text{cache\_capacity}$  do  
      |  $\text{cache.remove\_min}()$   
    end  
  end
```

Part Three 评价指标&实验结果

我们将我们的研究成果和现有的3种做对比：

- *end-to-end*: 客户端和服务端间无缓存（炮灰）
- *LFU*: 当缓存已满时，访问频率最低的贴图将被从缓存中删除
- *LRU*: 我们删除从最后一次访问以来持续时间最长的贴图

验证策略优良的指标：

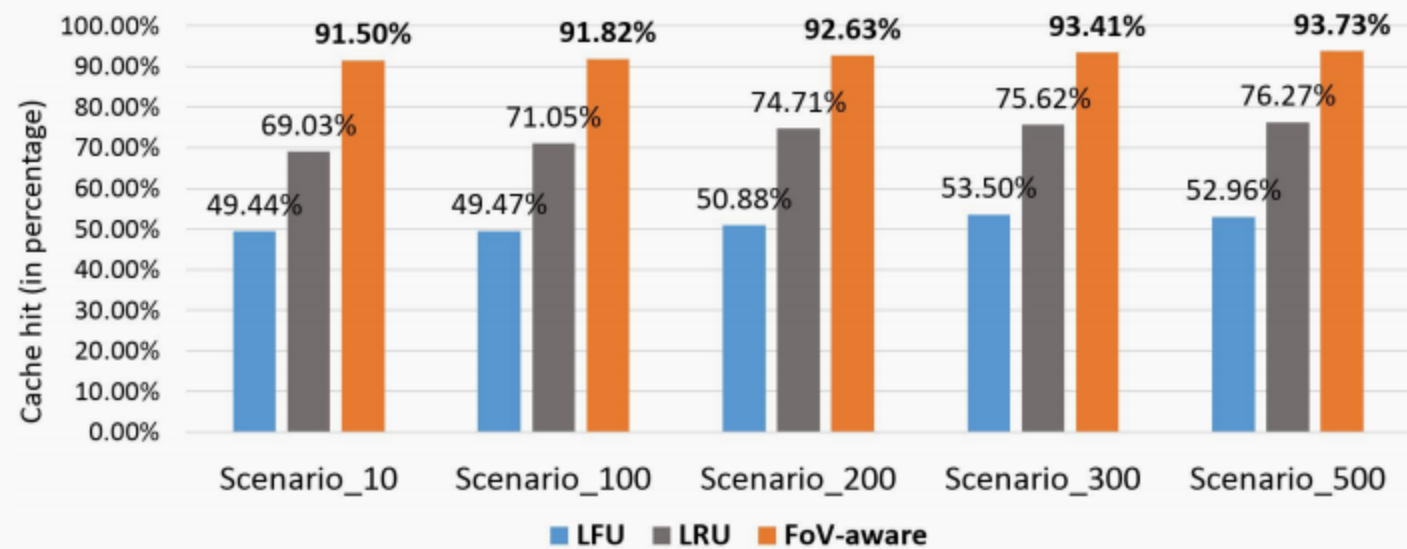
- *Tile hit* (缓存命中率):请求的贴图在缓存中找到的比例。
- *Bandwidth saving*[带宽节省(以百分比表示)]:当客户端请求命中缓存时, 所节省的带宽超过所有请求的总带宽消耗。
- *Reb*(缓冲频率): 没有任何视频可以播放了
- *DoR* [延迟时间(以秒为单位)]:该指标显示客户端经历延迟的总持续时间。
- *Qua*[高质量的FoV(以百分比表示)]:客户端接收瓷砖内FoV高质量超过360视频的总段数的片段数。

(1) Caching Policy Performance

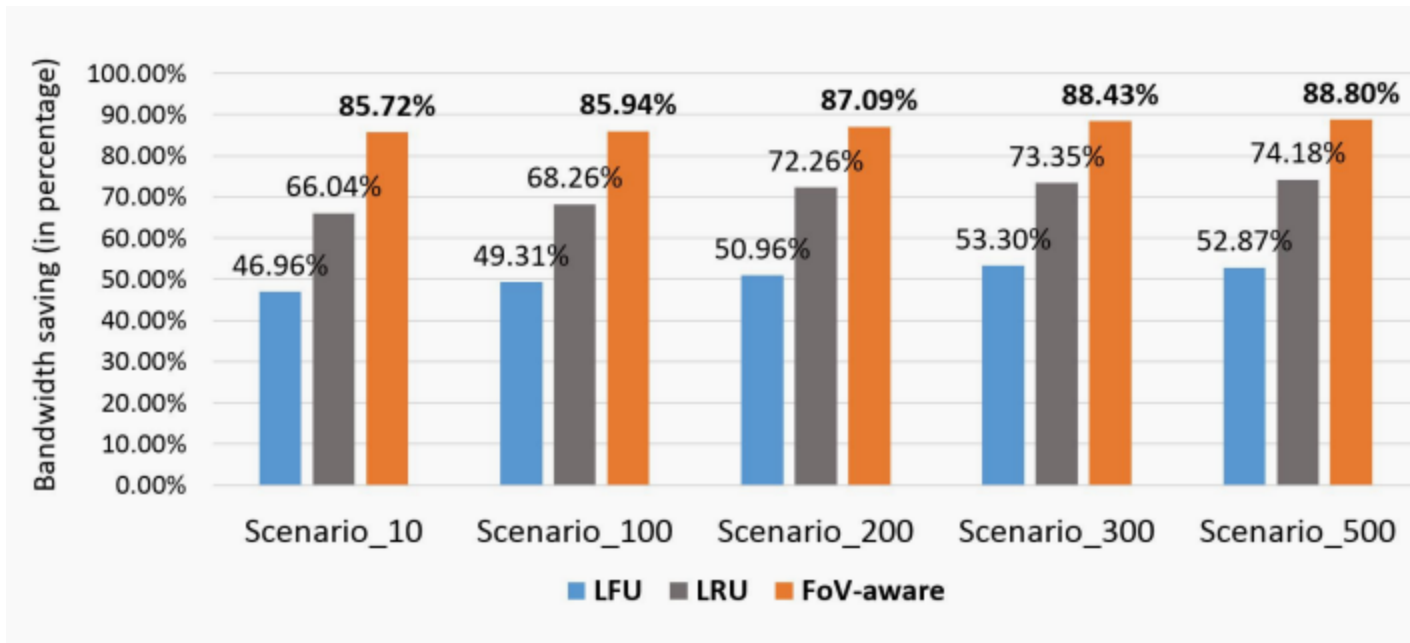
性能指标: *Tile hit + Bandwidth saving*

- 较高的缓存命中率减少了对内容服务器的请求数量，从而节省了核心网络中的带宽。

Tile hit:



Bandwidth saving:



结论：

最好的缓存策略：删除将来被请求的可能性最小的块。

FoV-aware缓存策略利用常见的fov现象(即，观看同一视频的大多数用户的fov有显著的重叠)来驱动对未来请求贴图的更好预测。

(2) Streaming Performance

性能指标: Reb, Dor, Qua

Table 2: Number, duration of rebuffering events and high quality for FoV in End to End

Scenarios	10			100			200			300			500		
	Reb	DoR	Qua	Reb	DoR	Qua	Reb	DoR	Qua	Reb	DoR	Qua	Reb	DoR	Qua
End to End	20.84%	5.75 Sec	20.65%	22.85%	6.17 Sec	11.08%	25.84%	7.02 Sec	3.12%	26.69 %	8.16 Sec	0.31%	40.19%	10.92 Sec	0.0%

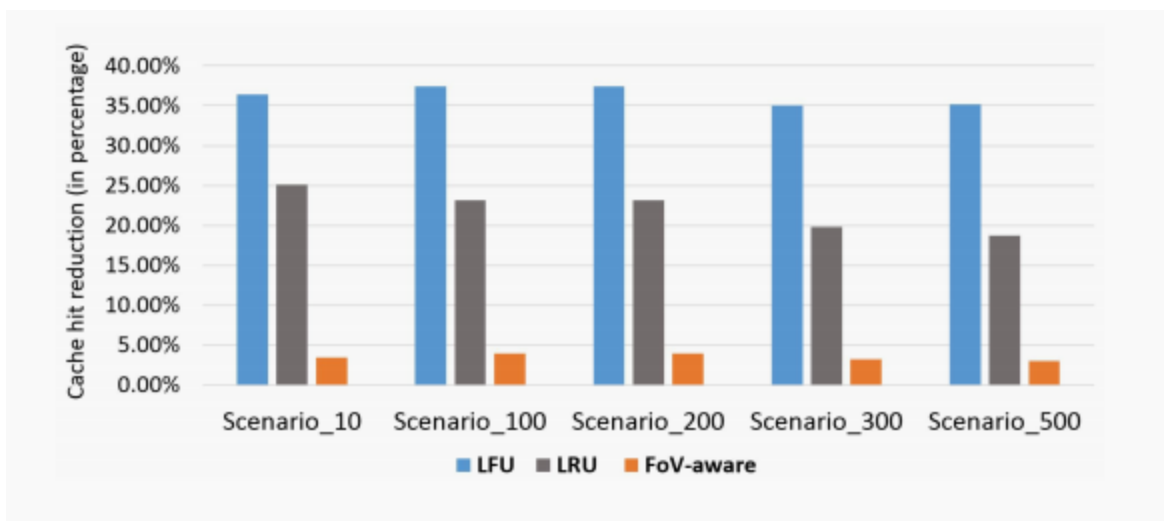
Table 3: Number, duration of rebuffering events and high quality for FoV in FoV-aware, LFU and LRU - All scenarios - Cache capacity: case_1

Scenarios	10			100			200			300			500		
	Reb	DoR	Qua	Reb	DoR	Qua	Reb	DoR	Qua	Reb	DoR	Qua	Reb	DoR	Qua
FoV-aware	20.20%	5.71 Sec	22.08%	20.36%	5.73 Sec	21.45%	20.78%	5.78 Sec	19.17%	20.92%	5.80 Sec	16.32%	22.50%	6.10 Sec	15.18%
LFU	20.71%	5.75 Sec	19.24%	22.34%	6.03 Sec	14.83%	23.86%	6.47 Sec	10.88%	25.53%	6.94 Sec	8.84%	31.75%	8.46 Sec	7.50%
LRU	20.69%	5.72 Sec	21.49%	21.16%	5.79 Sec	18.77%	21.61%	5.87 Sec	14.52%	21.76%	6.07 Sec	11.61%	24.81%	6.69 Sec	10.23%

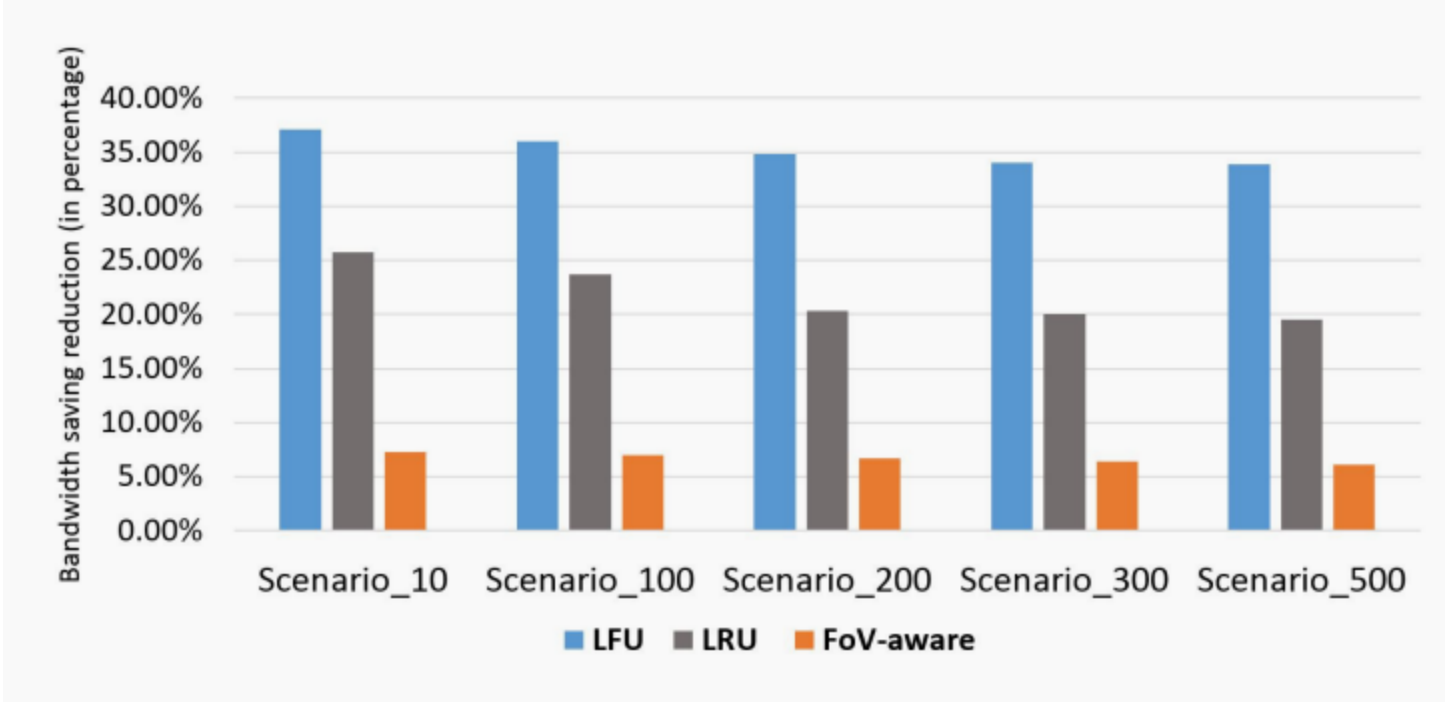
(3) Impact of Cache Capacity

研究缓存容量对所有三种缓存策略的影响，我们让缓存容量都变小。

对于命中率，Fov - aware的策略比另外两种降低的都小



同时对于Bandwidth saving来说，也是Fov策略性能下降的更小。



Part Four 实验环境和数据集

