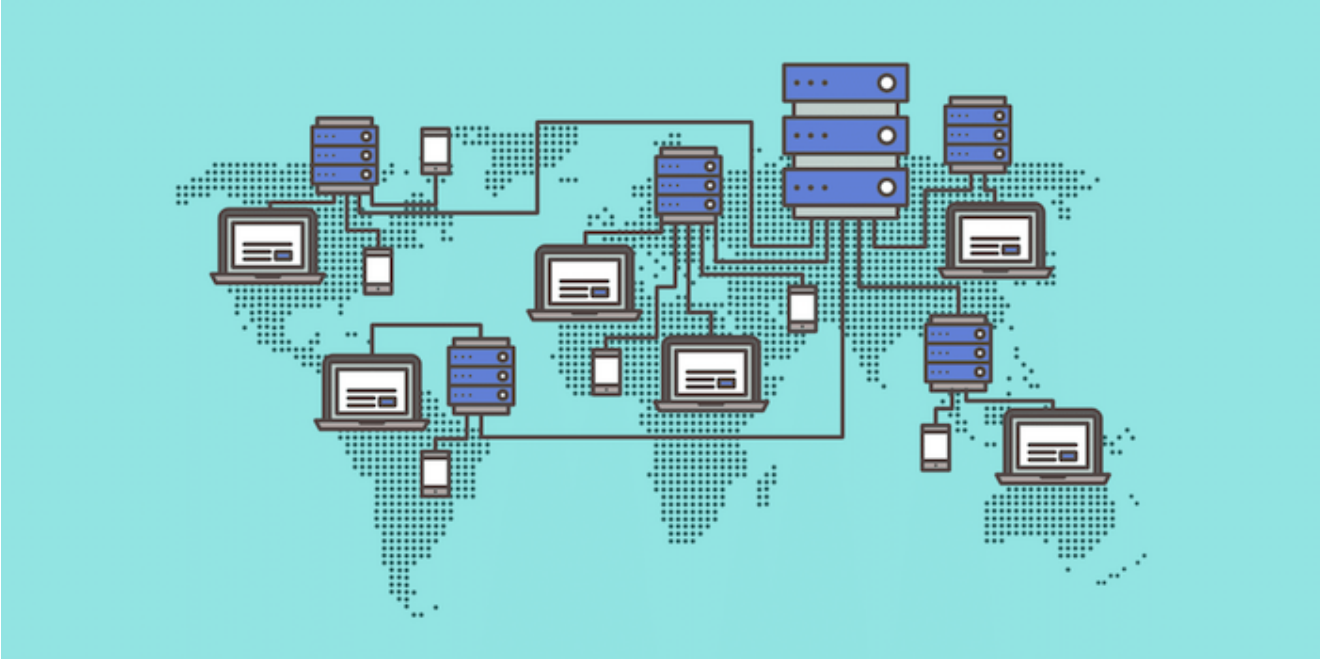
Category-aware Hierarchical Caching for Video-on-Demand Content on YouTube

Christian Koch, Johannes Pfannmüller, Amr Rizk, David Hausheer, Ralf Steinmetz

CDN (Content Distribution Network)

- Geographically distributed network of proxy servers and their data centers
- Content in caches close to the users
- Increasing the Quality of Experience
- Lower the amount of core Internet traffic by avoiding redundant transmissions from distant content sources



Typical topology in CDN cache hierarchies

- The first level caches are connected to the content source
- All caches only communicate with caches under or above their level
- Clients only attach to leaf caches

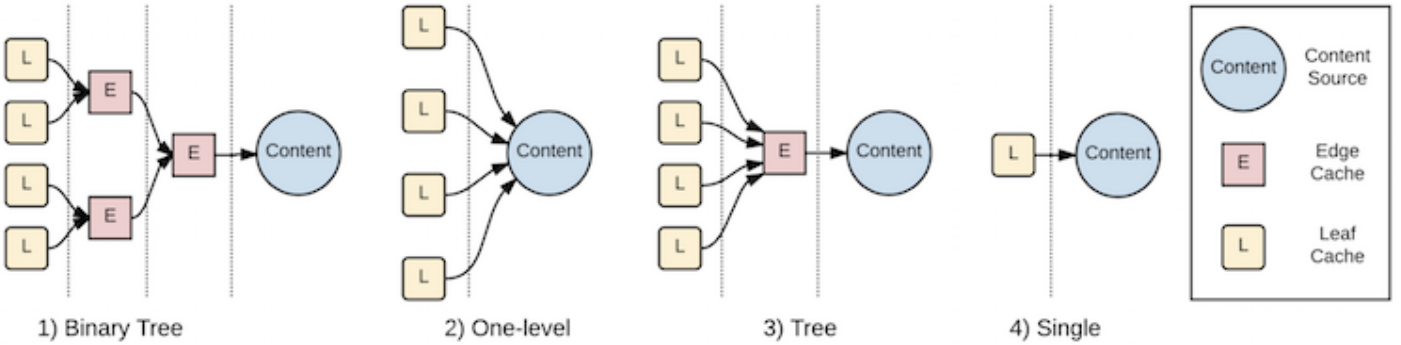


Figure 1: Overview of typical topology components in CDN cache hierarchies [26]

Motivation of the Problem

- Existing works on CDN cache performance focus mostly on distinct caching metrics given an abstract workload model
- CDNs serve contents of different popularity dynamics
- The nature of the geographical distribution and connection of caches is often oversimplified

Problem Description

- Design of a Content Category-aware Cache
  - Differentiating the caching strategies with respect to the different categories
  - Adapts the cache storage share assigned per category under changing workload compositions

Background

- Admission Policies
- Eviction Policies

Admission Policies

- Leave Copy Everywhere (**LCE**) stores the item on every cache it passes.
- Leave a Copy Down (**LCD**) stores the item only on the cache that is the direct successor of the cache on the answer path that generated a hit
- Move Copy Down (**MCD**) moves the requested item to the succeeding cache of the cache that generated a hit.
- Probability Admission (**Prob**) assigns a probability value for each cache on the answer path that defines how likely it is to store a bypassing item.
- **NHIT** Caching stores an item only if it has been seen at least N-times within a pre-defined time interval.

Eviction Policies

- Least Frequently Used (**LFU**)
- Least Recently Used (**LRU**)
- Segmented Least Recently Used (**SLRU**)
- Adaptive Replacement Cache (**ARC**)

Segmented Least Recently Used (SLRU)

- Splitting the cache storage in two parts
- The first part is denoted as the probationary part and stores items on the first request
- If a second request occurs the item is moved to the second cache part, denoted as the protected part
- If an item is evicted from the protected part, it is moved to the probationary part to provide it with a second chance
- Replaces the probationary part's last element in a FIFO-fashion

Adaptive Replacement Caching (ARC)

- Extends SLRU by introducing ghost lists, which keep track of recently evicted items
- Each cache division has its own ghost list
- The ID of the evicted item from one cache is stored in its ghost list
- Adjusting the size of the two caches based on ghost list

Contribution

- **Cache Policy Performance Analysis**
- **Design of a Content Category-aware Cache**
- **Trace-based Evaluation**

Cache Policy Performance Analysis:

- A thorugh analysis of combinations of cache admission and eviction policies
- Evaluate various caching strategies' perormance using a real world trace covering one week of YouTube requests observed in a large European mobile ISP network.
- Identify the best performing caching strategies overall as well as for distinct video categories

Design of a Content Category-aware Cache:

- Adaptive Category-aware Designed Caching approach, denoted as ACDC
- Divides the cache space in several divisions assigned to distinct video categories.
- This enables differentiating the caching strategies with respect to the workloads and popularity dynamics of the different categories.
- A dynamic cache division size adaptation strategy (DSAS)

Trace-based Evaluation:

- Benchmark ACDC against the best performing state-of-the-art caching strategies.
- Conduct trace-based simulations capturing more than 10^7 requests to YouTube to $\sim 1.6 \cdot 10^6$ different videos.
- Vary the cache storage capacity and investigate the influence of different cache topologies on cache hit rate, cache write operations, and transmission latency as an indicator of user-perceived QoE.

SYSTEM DESIGN

- Content-awareness
  - When a video enters the probationary cache, ACDC immediately retrieves the video's category ID based on the corresponding video ID by using YouTube's Data API v3
- Focus on an eviction strategy
  - Eviction strategy easier to extend
  - Admission strategies already consider the position of the content in the cache hierarchy which renders it difficult to combine with content categories

SYSTEM DESIGN

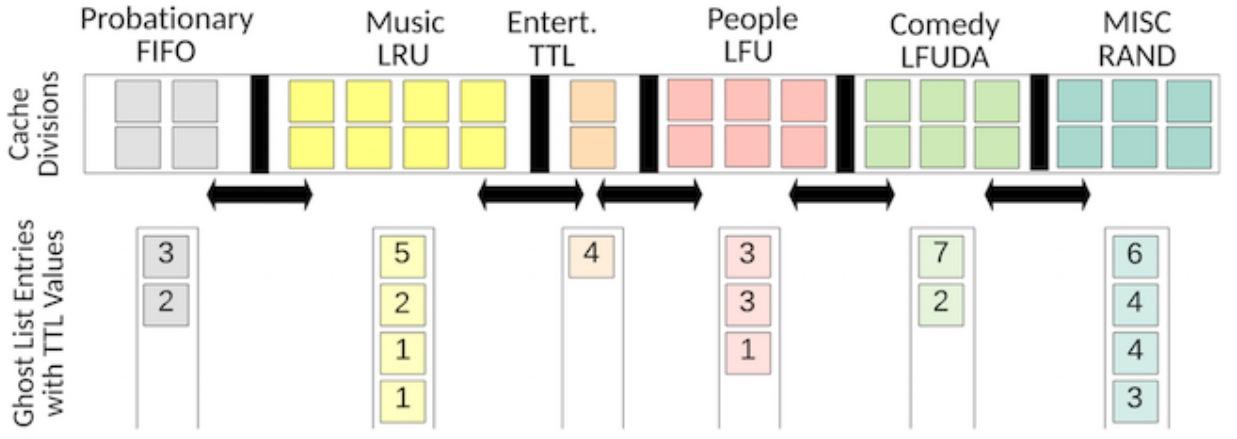


Figure 4: Example state cache space assignment. Numbers in the ghost list indicate the content's time-to-live (TTL).

Division Size Adaptation Strategy

(Division to be decreased)

- Smallest ghost list (**SGL**): A comparably small ghost list indicates a too large cache division as the ghost is rarely used
- Largest ghost list (**LGL**): A large ghost list indicates that video segments cached in this cache division are evicted more often, i.e., the corresponding cache division does not generate as many hits as cache divisions with small ghost lists
- Relatively smallest ghost list (**RSGL**): This is less fair for small cache divisions, but considers the overall request workload heterogeneity.
- Relatively largest ghost list (**RLGL**): Similar to LGL, this relative measure considers that a cache division larger than another one is allowed to have a larger ghost list

Evaluation

- **Trace**: A real-world trace that consists of over 10 million video requests to YouTube videos. The overall content catalog comprises more than 1.6 million videos. The trace was recorded in 2014 by a large European ISP
- **Evaluation Metrics**: Cache hit rate, Startup Delay, Write Operations.
- **Cache Hierarchies**:

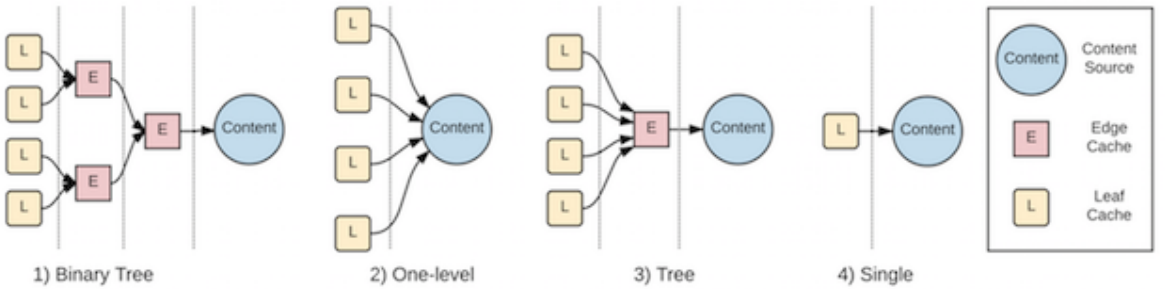


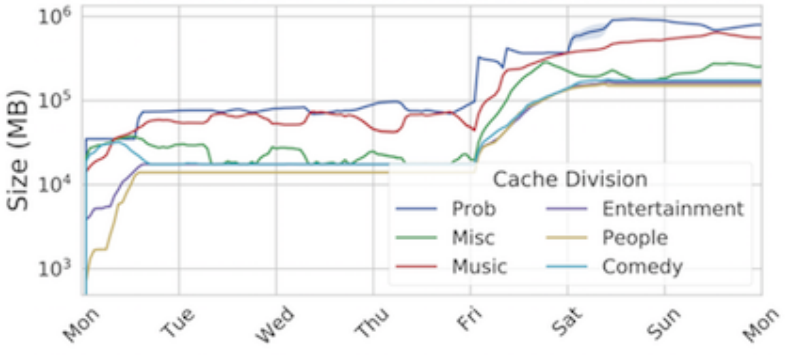
Figure 1: Overview of typical topology components in CDN cache hierarchies [26]

Result

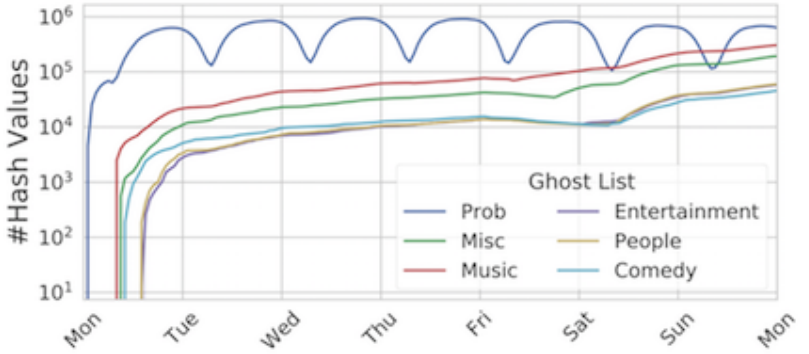
Table 3: The same thing as above but with ttfs instead of latency

		ACDC as Eviction Policy				Best Caching Strategy w/o ACDC					Improvement by ACDC (%)		
Hier.	Size	Adm.	CHR	#Writes	TTFS	Adm.	Evict.	CHR	#Writes	TTFS	CHR	TTFS	#Writes
1	1 GB	NHIT1	0.10	5.29	37.77	NHIT2	ARC	0.22	5.03	38.10	-52.83	-1.12	-5.24
1	10 GB	NHIT2	0.91	5.02	36.66	LCD	ARC	0.77	3.30	36.40	18.39	0.72	-52.03
1	100 GB	LCD	2.95	3.69	34.70	LCD	ARC	2.93	3.98	34.44	0.38	0.71	7.13
1	1 TB	NHIT2	8.04	4.96	31.30	NHIT2	ARC	7.9	4.95	31.00	1.92	0.97	-0.05
1	10 TB	NHIT1	19.49	5.11	26.11	NHIT1	SLRU	19.22	5.21	25.11	1.36	3.78	1.79
1	100 TB	LCE	32.67	5.26	20.80	LCE	LFUDA	32.62	5.39	19.17	0.17	7.86	2.41
1	1 PB	LCE	41.19	4.70	17.39	LCE	LFU	40.51	5.04	15.43	1.65	11.28	6.81
2	1 GB	NHIT1	0.25	5.10	46.03	NHIT2	ARC	0.43	4.83	46.70	-42.92	-1.44	-5.61
2	10 GB	NHIT1	1.63	5.09	44.27	NHIT2	ARC	1.46	4.82	44.17	11.48	0.22	-5.64
2	100 GB	NHIT2	6.40	4.75	40.95	NHIT2	ARC	5.64	4.80	41.08	13.36	-0.33	0.87
2	1 TB	NHIT2	15.25	4.77	35.93	NHIT2	SLRU	14.80	4.87	35.70	3.07	0.64	2.02
2	10 TB	NHIT1	39.62	4.94	26.35	LCE	SLRU	36.44	5.50	24.68	8.74	6.36	10.25
2	100 TB	LCE	60.00	5.01	18.05	LCE	LFUDA	56.10	5.16	15.86	6.94	12.15	2.89
2	1 PB	LCE	63.33	4.52	15.44	LCE	LRU	62.25	4.84	15.86	1.74	2.97	6.55
3	1 GB	LCD	0.17	2.22	41.56	NHIT2	ARC	0.33	4.97	42.19	-48.66	-1.50	55.29
3	10 GB	NHIT2	1.24	4.96	39.98	LCD	ARC	1.23	3.59	39.54	0.88	1.09	-38.33
3	100 GB	LCD	4.81	3.86	37.34	LCD	ARC	4.42	4.13	36.70	8.80	1.71	6.49
3	1 TB	NHIT2	13.06	4.90	32.82	NHIT2	ARC	11.52	4.90	31.94	13.38	2.68	0.10
3	10 TB	NHIT1	30.15	5.07	25.22	LCE	SLRU	28.33	5.60	24.11	6.41	4.41	9.59
3	100 TB	LCE	47.87	5.20	18.38	LCE	LFUDA	45.65	5.31	18.11	4.86	1.49	2.02
3	1 PB	LCE	54.83	4.90	14.51	LCE	LFUDA	54.46	4.97	14.31	0.67	1.42	1.32
4	1 GB	PROB0.75	0.96	5.42	44.77	NHIT2	ARC	0.98	5.51	44.03	-0.53	1.64	1.54
4	10 GB	NHIT2	3.76	5.45	42.24	NHIT2	ARC	3.64	5.49	44.03	3.30	2.29	0.85
4	100 GB	NHIT1	10.85	5.26	37.84	NHIT2	ARC	10.38	5.45	36.45	4.50	3.68	3.50
4	1 TB	NHIT2	24.98	5.41	31.24	NHIT2	ARC	22.37	5.41	30.57	11.68	2.13	0.03
4	10 TB	NHIT1	47.43	5.52	21.63	LCE	SLRU	46.60	6.05	21.30	1.76	1.51	8.87
4	100 TB	LCE	67.41	5.56	13.28	LCE	LFUDA	67.09	5.62	13.02	0.48	1.98	1.05
4	1 PB	LCE	71.56	5.56	11.48	LCE	LFUDA	71.56	5.56	11.48	0.00	0.00	0.08

Result



(a) Cache division size evolution



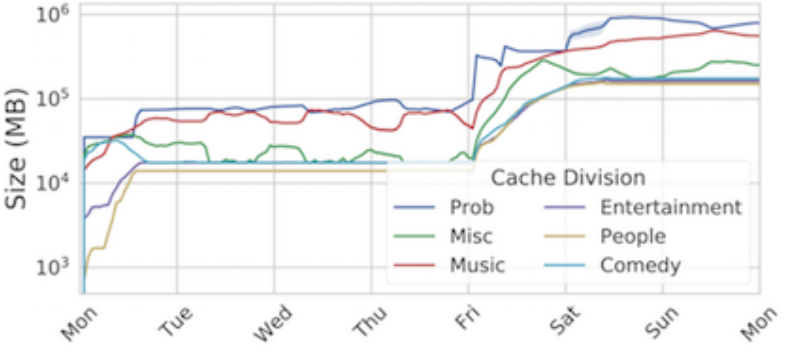
(b) Ghost list size evolution

Pros and Cons

- Pros:
  - ACDC outperforms state-of-the-art strategies
  - Real world large scale trace
- Cons:
  - Category-retrieving Overhead
  - Too much heuristic

Questions to discuss

- Does their DSAS even work like they said?
  - "We observed that DSAS often led to a convergence of smaller cache divisions to a very small size."
  - "we introduce a minimum cache division size **M**"
  - Largest ghost list (**LGL**): the corresponding cache division does not generate as many hits as cache divisions with small ghost lists



(a) Cache division size evolution

Thanks!!!