Bucket-Filling:

An Asymptotically Optimal VoD Network with Source Coding

Published on *IEEE TRANSACTIONS ON MULTIMEDIA*, VOL. 17, NO. 5, MAY 2015

Chang, Zhangyu

Supervised by ***Prof. Gary Chan***

9 December 2019

Contents

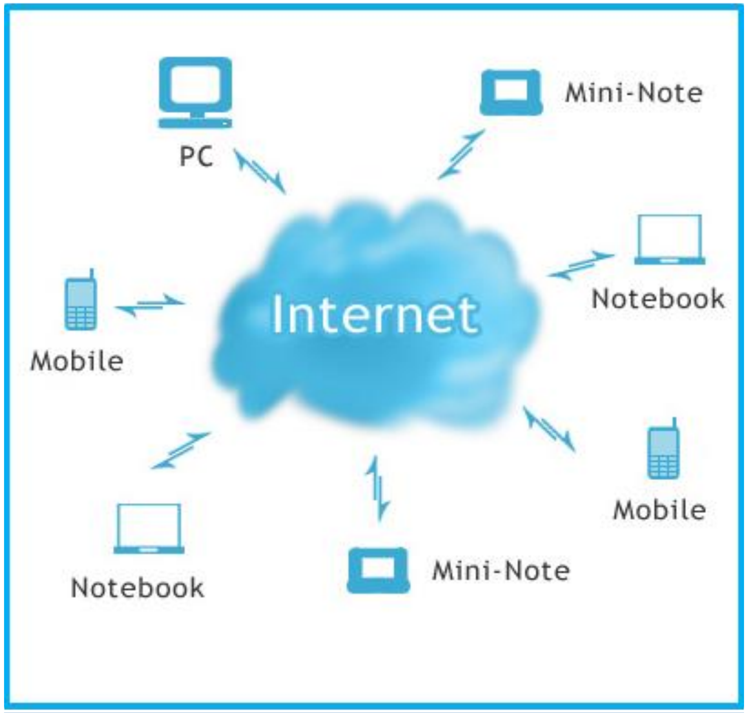
Video-on-demand as cloud service

Video-on-Demand

- **Anytime & anywhere**
- **Timely** content delivery
- **Resource** consuming
- **Most** (over 50%) of the Internet traffic

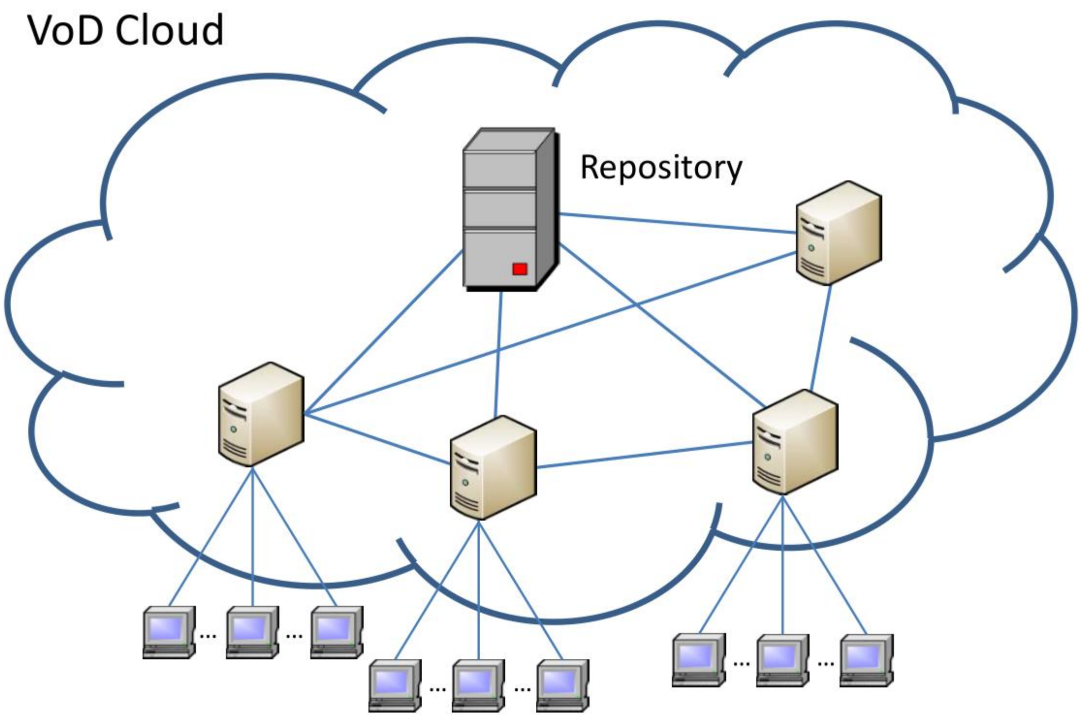
Distributed Cloud

- **Bandwidth** and **Storage** at geo-dispersed servers
- Servers cooperatively **store** and **retrieve** movies



A Typical Distributed VoD Cloud Service

Deployment of a VoD cloud



Repository
Complete movie storage
Proxy server
Distributed server to serve users cooperatively
User
Each user is associated with a local (home) server

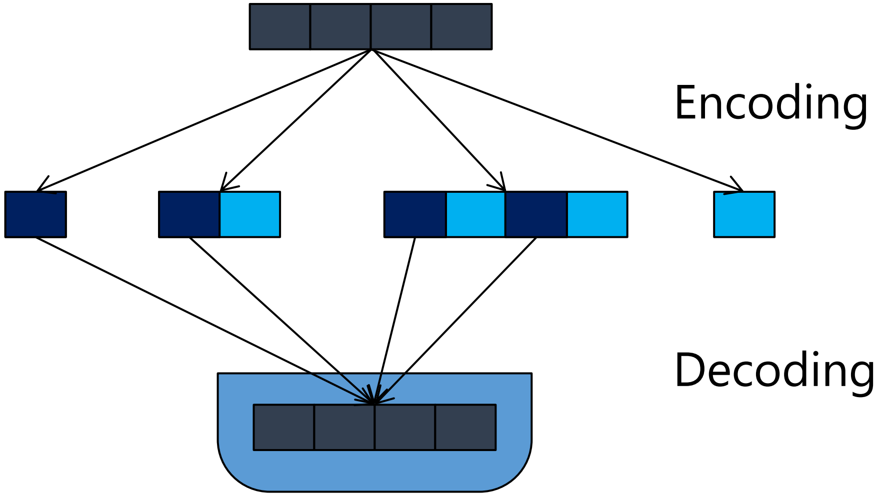
"Bucket-filling" with source coding

A movie can be divided into several packets for streaming. Each packet is further source-coded to generate n coded symbols.

By collecting **any** q coded symbols, one can recover the original movie source. (q is usually given in the system, $n \geq q$.)

This flexibility in choosing coded symbols leads to better optimality.

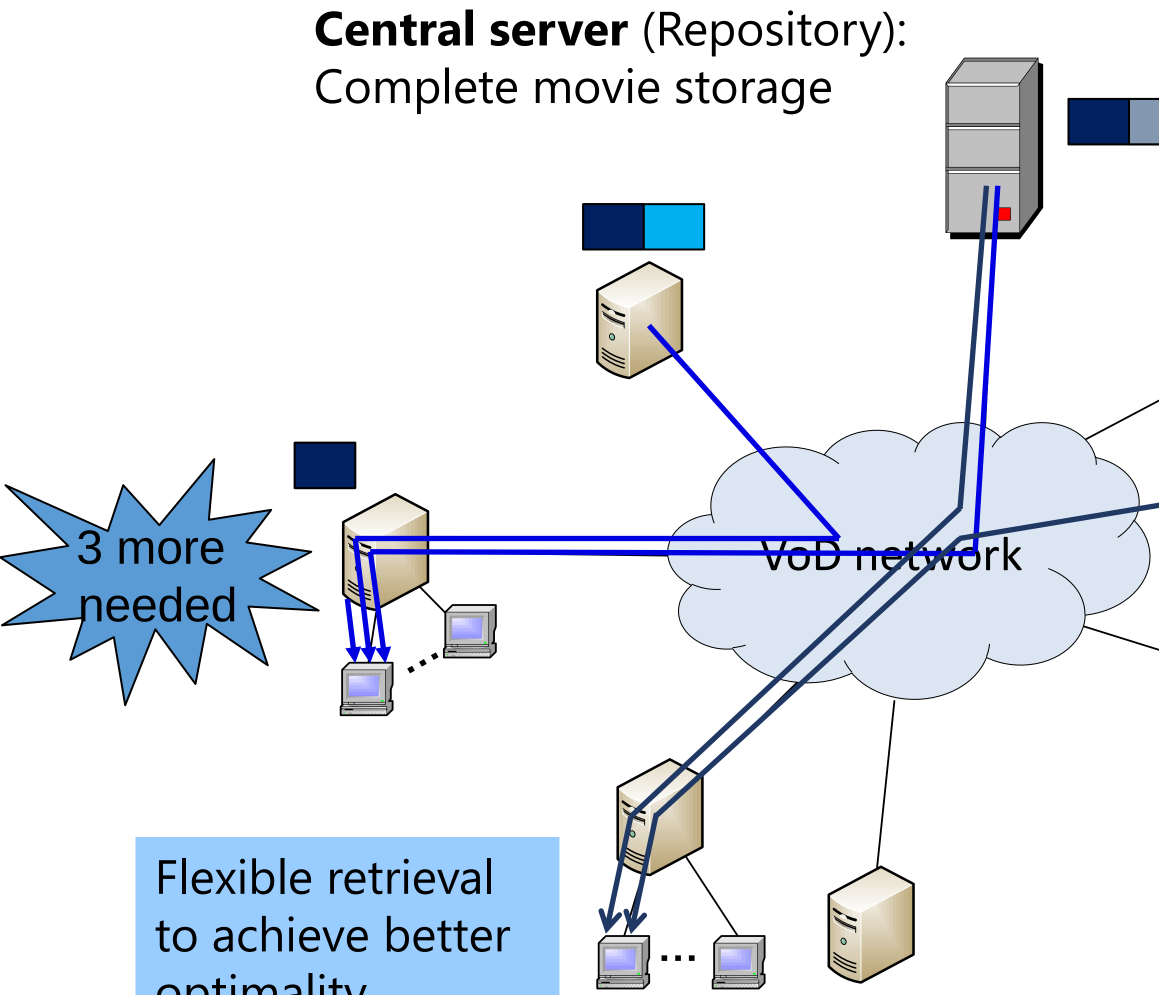
Example:
$n = 8$
$q = 4$



Complexity:
$O(q^2 n)$ for encoding
$O(q^2)$ for decoding

In practice:
Overhead **much lower** as compared with video decoding

Symbol distribution and retrieval



Example:
$n = 8$
$q = 4$

Flexible retrieval to achieve better optimality

Proxy server **(limited storage)**

Client

Major challenge: Storage, Retrieval & Complexity

Cloud parameters

Movie streaming rate, popularity, price, etc.

Storage

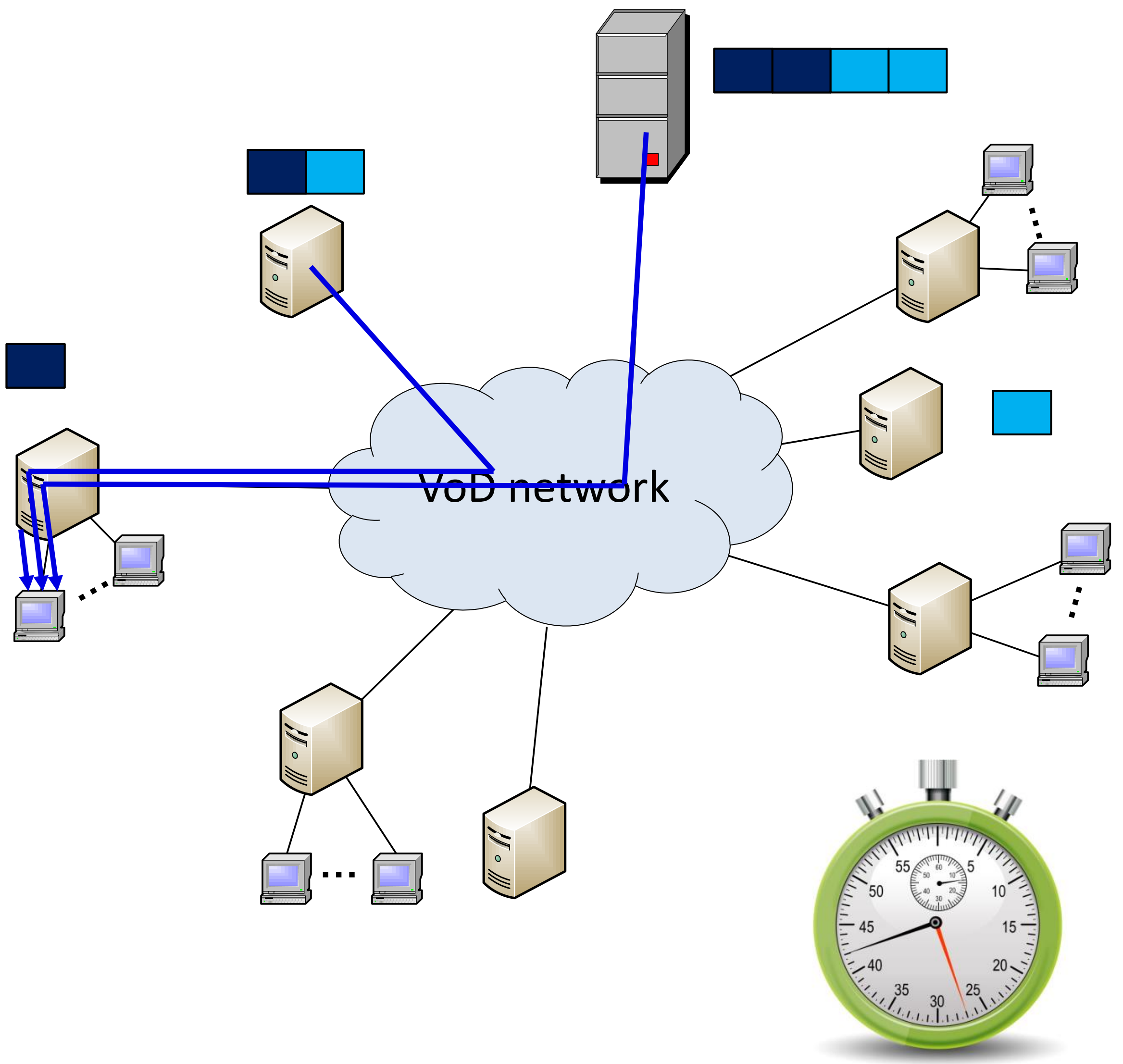
How many symbols to store at each server

Retrieval

Which servers to stream the rest symbols

Complexity

Running time for large movie pool



Objective

Total deployment /running cost

Server cost

- Storage
- Bandwidth utilization

Network cost

- streaming among servers to serve the misses

Minimize total deployment/running cost

Low running time complexity

Approach

Relaxed Linear Programming

- consider the number of symbols ($n_v^{(m)}$) stored in each server as continuous
- formulate a linear programming **(LP)** problem

Discretize the LP solutions for movie Storage & Retrieval

- Greater q leads to smaller discretization penalty
- Bucket-filling is **asymptotically optimal** in terms of q;
- i.e., system cost approaches the exact minimum as q increases

Clustering for Large Movie Pool

- Group movies by **K-means clustering** to reduce the algorithmic time

Contributions

1	***Bucket-filling*: distribution & retrieval with source coding**	**Comprehensive cost model** • Server cost (storage & streaming) • Network cost **Minimizing system deployment cost**
2	**Provably asymptotically optimality**	**Bucket-filling with LP is asymptotically optimal** • In terms of q • A greater q makes solution closer to the *exact global minimum* ($q = 30$ is good enough)
3	**Movie clustering & On-line re-optimization**	**Efficient movie clustering method** • Significantly reduce running time • With little sacrifice of deployment cost **On-line re-optimization with minimum system changes**

Related work

	Related Work	Bucket-filling
Heuristics: S. Borst *et al.* INFOCOM'10 A. Nimkar et al. IMSAA'09 S. Zaman *et al.* TPDS'11 *etc.*	• Not clear how far they are from the optimum	• Provably asymptotical optimality in *q*
Cost optimization: Y. R. Choe *et al.* ACMMM'07 D. Wu *et al.* CSVT'13 D. Niu *et al.* INFOCOM'12 *etc.*	• Consider cost only partially	• Comprehensively capture network access cost, storage constraint & bandwidth utilization
P2P VoD: Y. Zhou *et al.* INFOCOM'12 Y. Zhou *et al.* ToN'13 B. Tan *et al.* ToN'13 *etc.*	• Maximize the sharing of peers to offload the server load	• Minimize the deployment cost

Contents

Major Symbols Used

V	The set of servers (central and proxy servers)	$\boldsymbol{r_{uv}^{(m)}}$	Amount of movie m streamed from server u to v (seconds)
M	The set of movies	λ_v	Request rate at server v (requests/second)
$L^{(m)}$	Length of movie m before source coding (in seconds)	Γ_{uv}	Average transmission rate from server u to v (bits/s)
$p^{(m)}$	Access probability of movie m at server v	R_v	Total uploading rate of server v (bits/s)
$\boldsymbol{I_v^{(m)}}$	Amount of movie m server v stores (in seconds)	C_{uv}^{N}	Network cost due to directed traffic from server u to v
B_v	Storage capacity of server v (in seconds)	C_v^{S}	Cost of server v
$\alpha^{(m)}L^{(m)}$	Average holding (viewing) time of movie m (in seconds)	s	Movie streaming rate (bits/s)

JOSR:
*J*oint *O*ptimization on Movie *S*torage & *R*etrieval

Server Cost Network Cost

Minimize $\sum_{v \in V} C_v^S (B_v, R_v) + \sum_{u,v \in V} C_{uv}^N (\Gamma_{uv})$ → Total system deployment cost

Storage

Bandwidth utilization

Access bandwidth (consumed)

Subject to

Storage $0 \le I_v^{(m)} \le L^{(m)}, \ \forall v \in V, m \in M$ → Amount of symbols stored at v

Retrieval $0 \le r_{uv}^{(m)} \le L^{(m)}, \ \forall u, v \in V, m \in M$ → Amount of symbols retrieved from u to v

$\sum_{m \in M} I_v^{(m)} \le B_v, \ \forall v \in V,$ → Storage constraint at v

$r_{uv}^{(m)} \le I_v^{(m)}, \ \forall u, v \in V, m \in M$ → Supply only the amount stored

$\sum_{u \in V} r_{uv}^{(m)} \ge L^{(m)}, \ \forall v \in V, m \in M$ → A movie shall be retrieved

$$\Gamma_{uv} = \sum_{m \in M} p^{(m)} \lambda_v \alpha^{(m)} r_{uv}^{(m)} s, \qquad \forall u, v \in V$$

$$R_v = \sum_{u \in V, u \neq v} \Gamma_{uv}, \qquad \forall v \in V$$

Contents

Parameter discretization to achieve asymptotic optimum

Step 1: Linear Program

- Assume the number of symbols in each server as **continuous** variable

- Solve LP to get super-optimal symbol:

  storage $I_v^{(m)}$

  retrieval $r_{uv}^{(m)}$

Step 2: Discretization

- Symbol Storage: ($\boldsymbol{I_v^{(m)}} \rightarrow \boldsymbol{n_v^{(m)}}$)
  - **Step 1:** $\boldsymbol{n_v^{(m)}} \propto \boldsymbol{I_v^{(m)}}$
  - **Step 2:** round up/down $n_v^{(m)}$ by popularity

- Symbol Retrieval: ($\boldsymbol{r_{uv}^{(m)}} \rightarrow \boldsymbol{n_{uv}^{(m)}}$)
  - **Step 1:** $\boldsymbol{n_{uv}^{(m)}} \propto \boldsymbol{r_{uv}^{(m)}}$
  - **Step 2:** round up $n_{uv}^{(m)}$ to satisfy requests
  - **Step 3**: unsatisfied request to repository

Algorithmic complexity

LP	LP solver has constant expected iterations and $O(N^3)$ for each iteration (N: number of variables)

$\|V\|$	Number of servers
$\|M\|$	Number of movies

What if $|M|$ is large?

Linear Program

Asymptotically optimal

Discretize Symbol Storage

Discretize Symbol Retrieval

$O(|V|^6|M|^3)$

$O(|V||M|)$

$O(|V|^2|M|)$

$O(|V|^6)$ is a huge factor

Overall time complexity: $O(|V|^6|M|^3)$

Contents

Motivation of movie clustering

Load index:

$$d^{(m)} = p^{(m)}\alpha^{(m)}$$

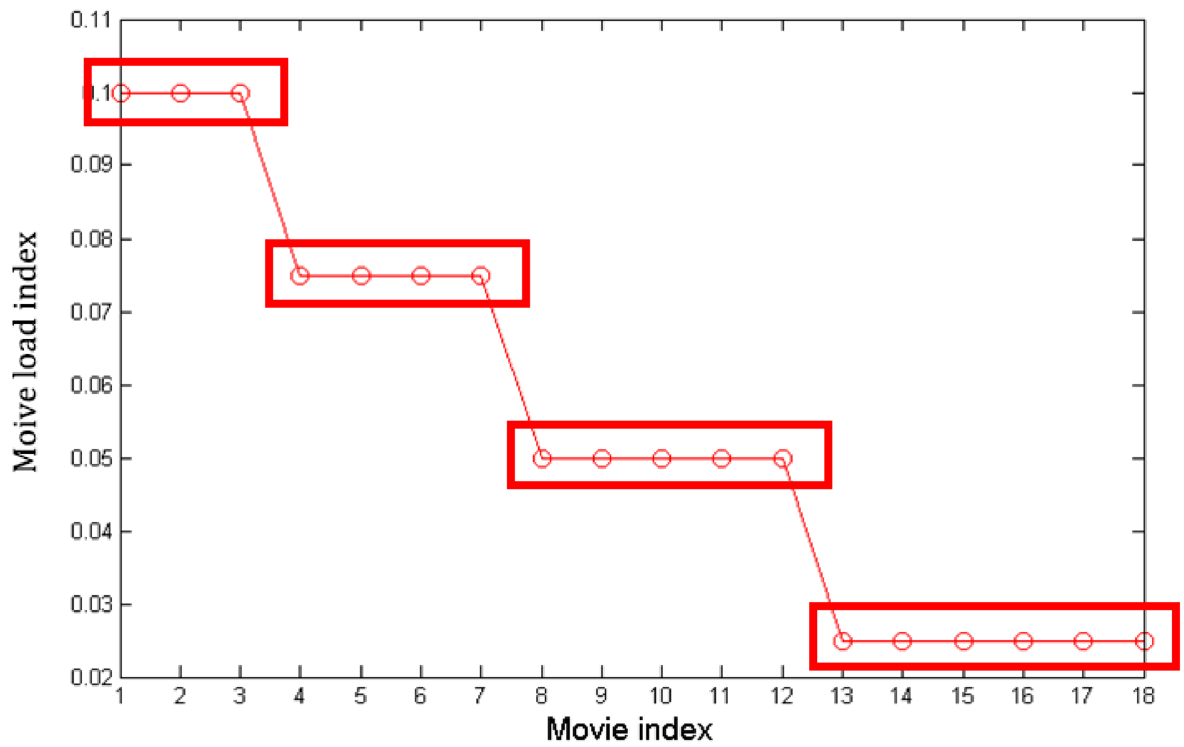
Access probability and **holding time** indicate the streaming load of movie

If group the movies with the **same** load index

Linear programming result will **NOT** change

Movie clustering

Minimize the load index difference within each group



K-means clustering for movie grouping

Minimize: $\arg_{g_i} \sum_{i=1}^{|G|} \sum_{m \in g_i} \left| \boldsymbol{d}^{(m)} - \boldsymbol{\mu}^{(g_i)} \right|^2$

- $\mu^{(g_i)}$ is the mean load index of group g_i
- Resulting group size may not be the same

K-means

Algorithmic complexity

|*G*|	Number of groups
|*M*|	Number of movies

K-means Clustering in 1D can be solved in polynomial time: $\boldsymbol{O}(|\boldsymbol{M}|^2|\boldsymbol{G}|)$

Movie group as a "*super movie*"

- Group length: Sum of the group movie length
- Group load index: weighted average of movie index within group

Parameter discretization from *group* to *movie*

	Guiding principle	Method
$\boldsymbol{n_v^{(g_i)}} \rightarrow \boldsymbol{n_v^{(m)}}$	movies in group g_i have similar $n_v^{(m)}$	**Rarest first**: increases the smallest $n^{(m)}$ by 1 for $m \in g_i$ until space for g_i used up.
$\boldsymbol{n_{uv}^{(g_i)}} \rightarrow \boldsymbol{n_{uv}^{(m)}}$	$n_{uv}^{(m)} = n_{uv}^{(g_i)}$ if possible	• If $n_{uv}^{(m)} > n_{uv}^{(g_i)}$ for some u, we reduce $n_{uv}^{(m)}$ to make $n_{uv}^{(m)} = n_{uv}^{(g_i)}$ • remaining requests to repository

Time complexity reduction

$\vert V\vert$	Number of servers
$\vert M\vert$	Number of movies
$\vert G\vert$	Number of clusters

LP

LP solver has constant expected iterations and $O(N^3)$ for each iteration (N is the number of variables)

$O(|V|^6|G|^3)$

$O(|V|^6)$ is a huge factor

Without Clustering:

$O(|V|^6|M|^3)$

Discretize Symbol Storage

$O(|V||M|)$

Discretize Symbol Retrieval

$O(|V|^2|M|)$

K-means Clustering

$O(|M|^2|G|)$

Reducing complexity by $O(|M|)$

On-line re-optimization

System Changes
Movie Changes
Server Changes
Network Changes

Re-Optimize

	Symbols in server v			
t				
$t+1$				
$\vec{\Delta}_v(t)$	$+1$	-1	-3	$+3$

Symbol changes

$+n > 0$	repository transfer n symbols of movie m to server v
$-n < 0$	server v discard n symbols of movie m

Contents

Environment setup

Movie popularity

- Zipf distribution: $f(i) \propto 1/i^s$
- $f(i)$: popularity of ith movie
- s: Zipf parameter

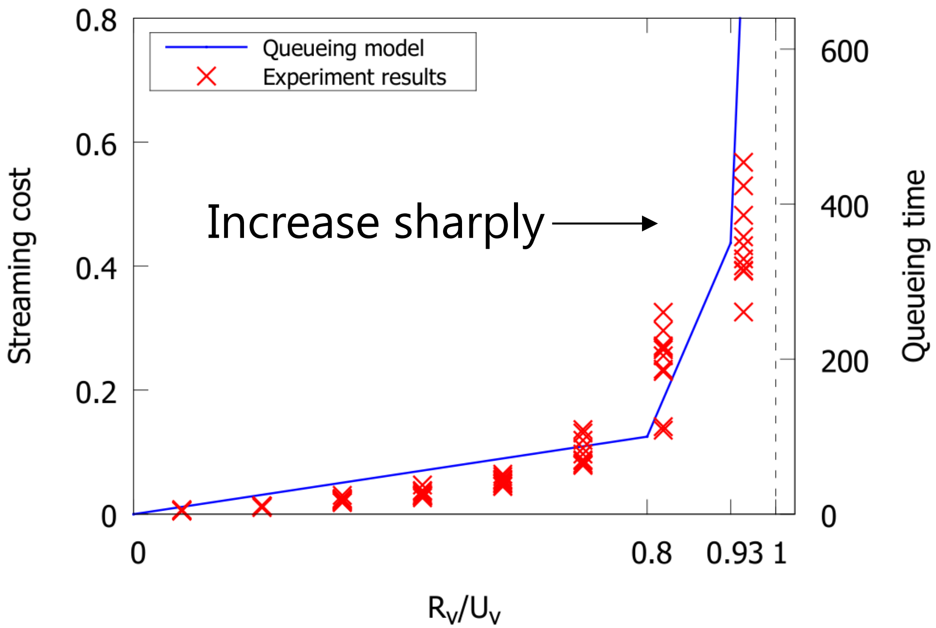
Server Cost

- **Storage cost**: proportional to storage capacity (B_v)
- **Streaming cost**: delay-based model (piece-wise linear)

Network cost

- proportional to end-to-end transmission bandwidth (Γ_{uv})

Delay-based Streaming cost model



Bandwidth utilization percentage

Performance metrics & comparison schemes

Performance Metrics

Total cost & components

- Server Storage cost
- Server streaming cost
- Network cost

Running time

- Time to obtain results by running algorithm

Comparison Schemes

Random

- Popularity-blind
- Randomly store

MPF

- Most Popular

Local Greedy

- IEEE *Infocom* 2010
- Full replication: *most popular*
- Single copy: *medium popular*
- No copy: *unpopular*

Uniform Clustering

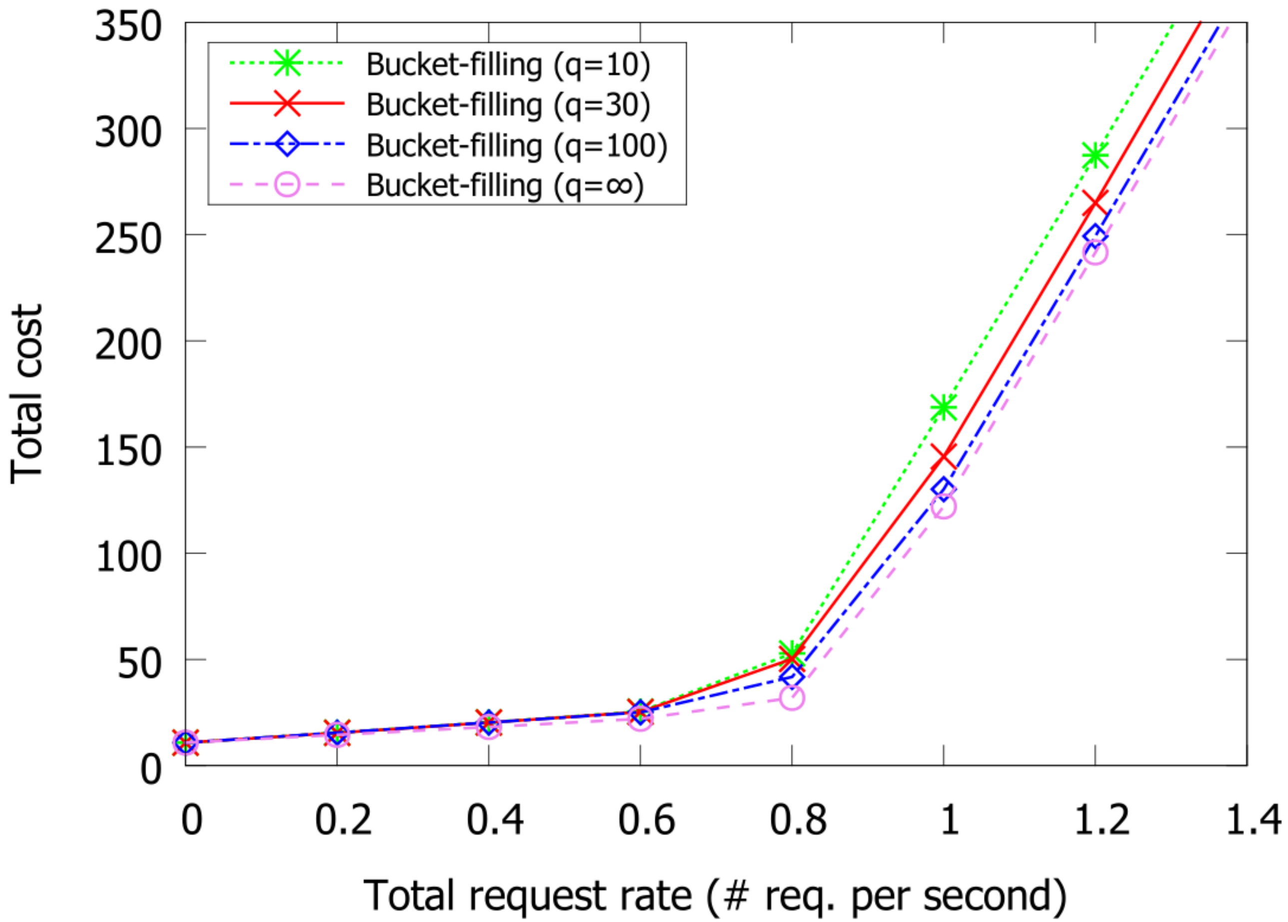
- Groups have the same size

Super-optimal

- Considering q as continuous

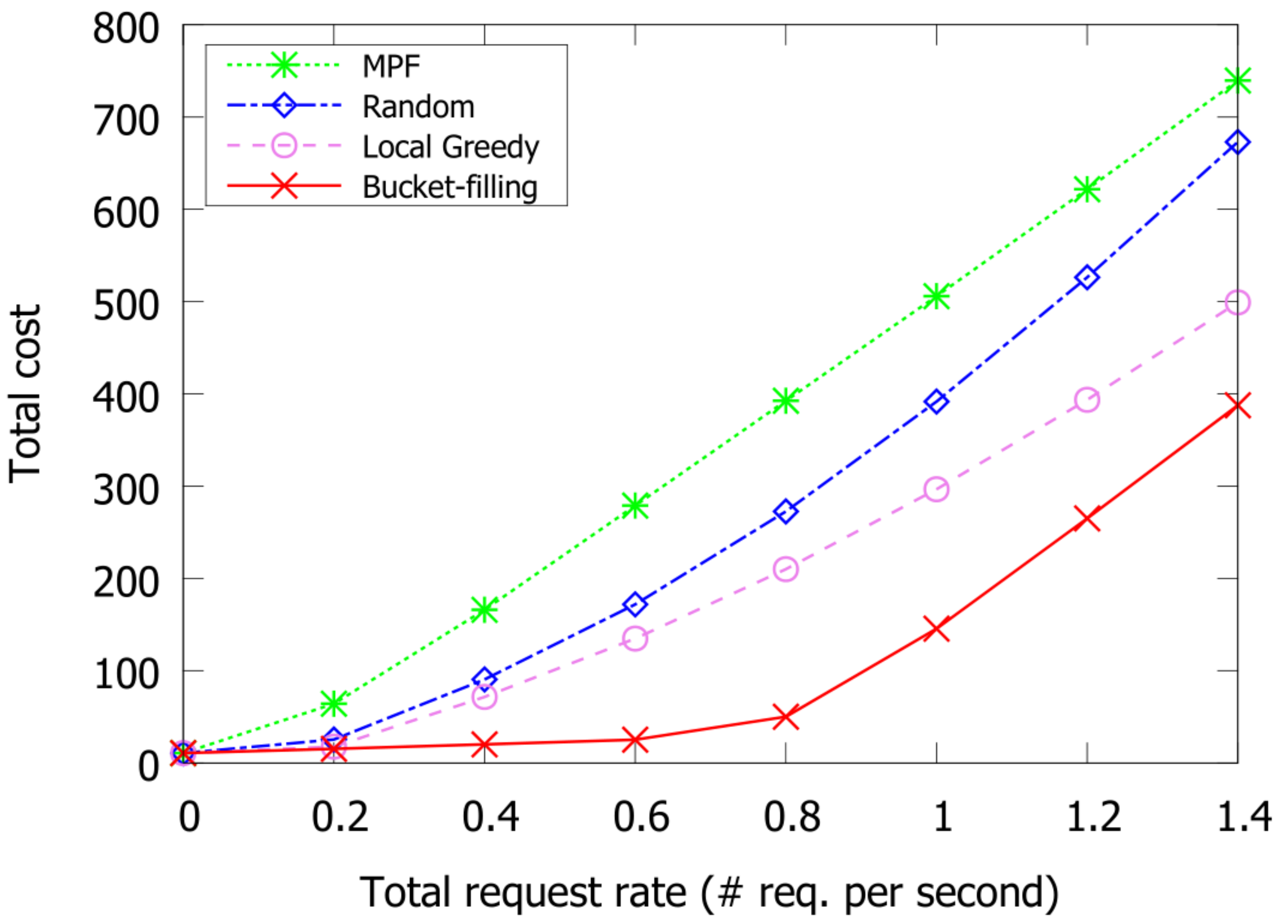
Asymptotically optimal

- Larger q, closer to *super-optimum*
- For finite q, the performance is close to *super-optimum*



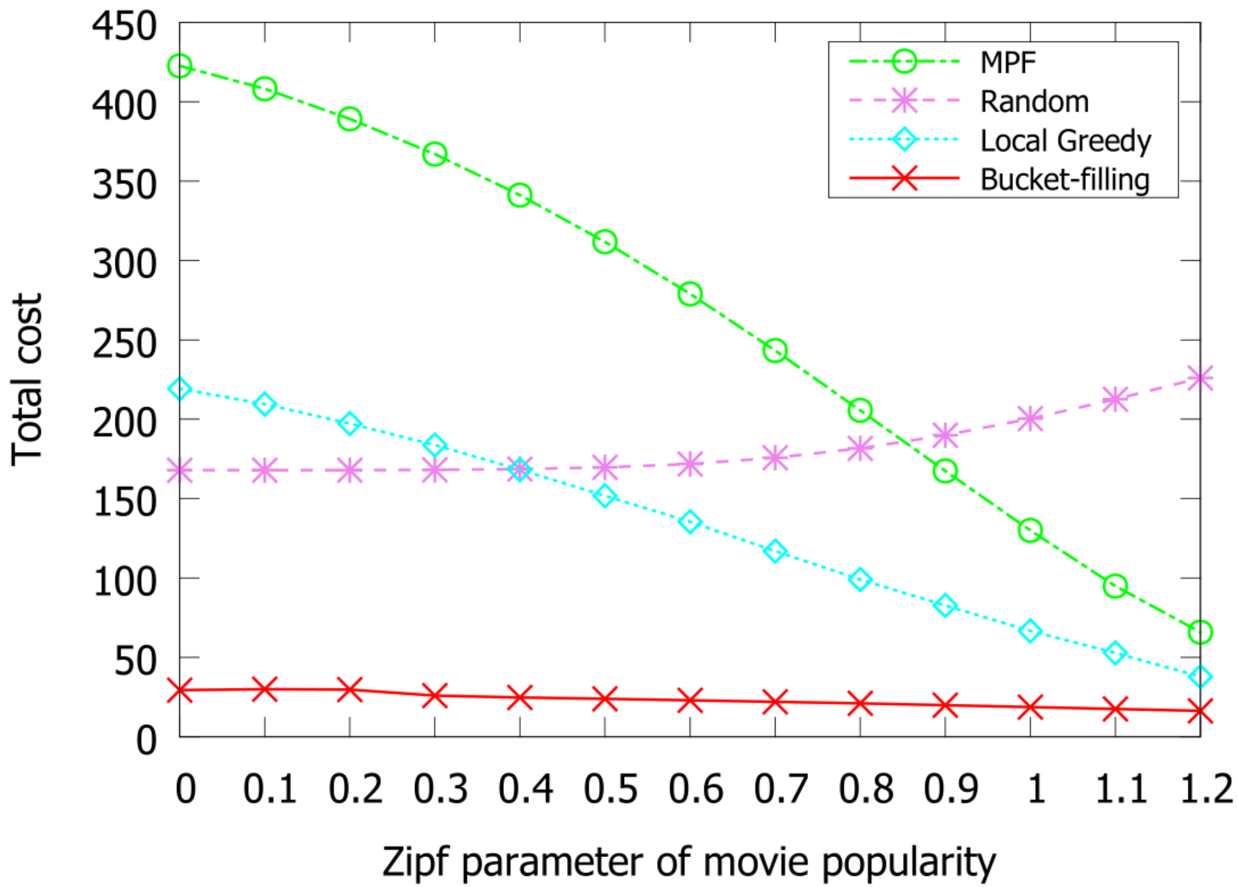
Substantially low cost

- Outperform by a wide margin



Insensitive to popularity skewness

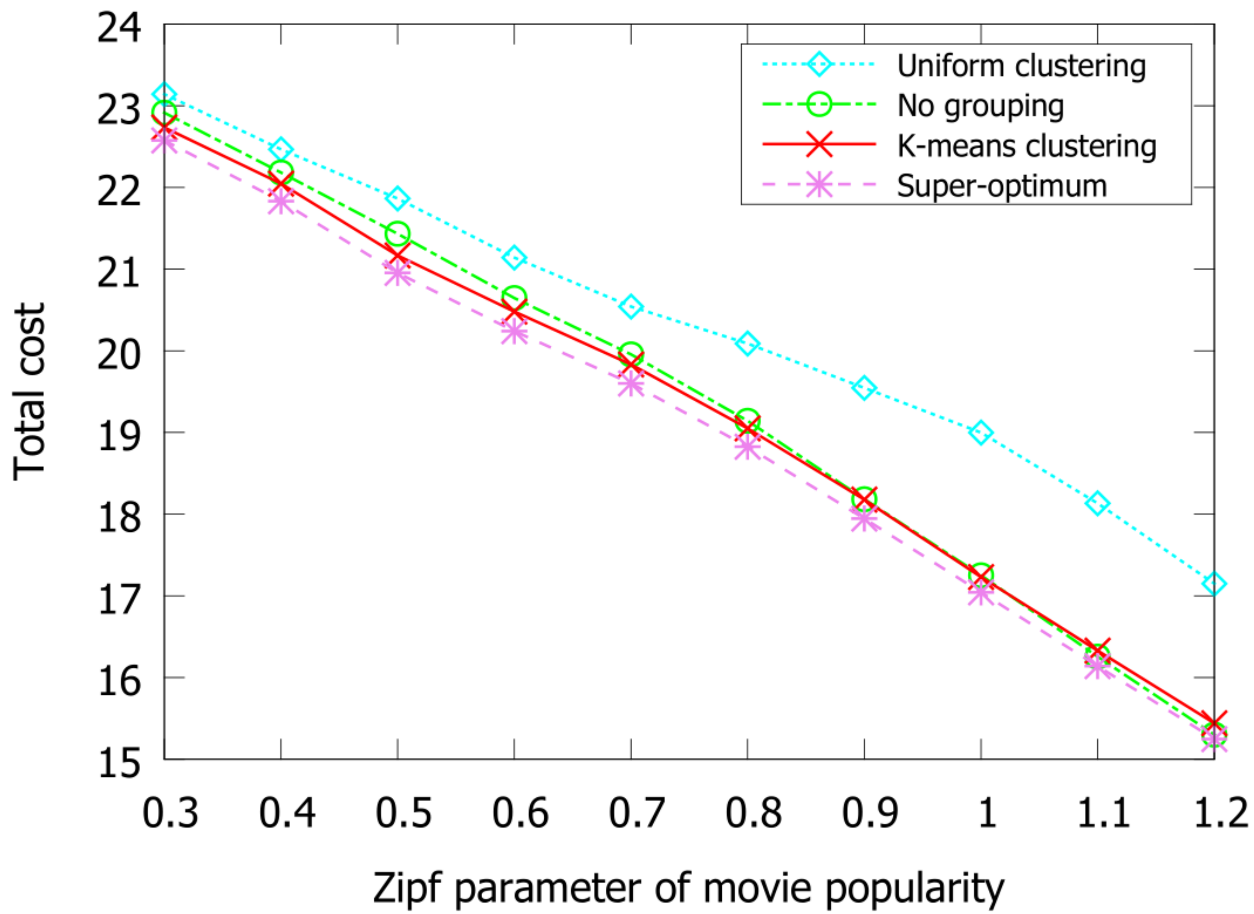
- Better utilize the proxy servers (Versus MPF)
- Cooperatively store (Versus Local-Greedy)
- Low miss rate (Versus Random)



Closely optimal grouping

- Still near optimum when grouped
- **K-means** Clustering outperforms more for larger skewness
- **K-means** even outperform the ungrouping method for small Zipf parameters

Smaller Zipf parameter leads to smaller grouping error; ungrouping method has larger round-off error.



Perform well for large movie pool

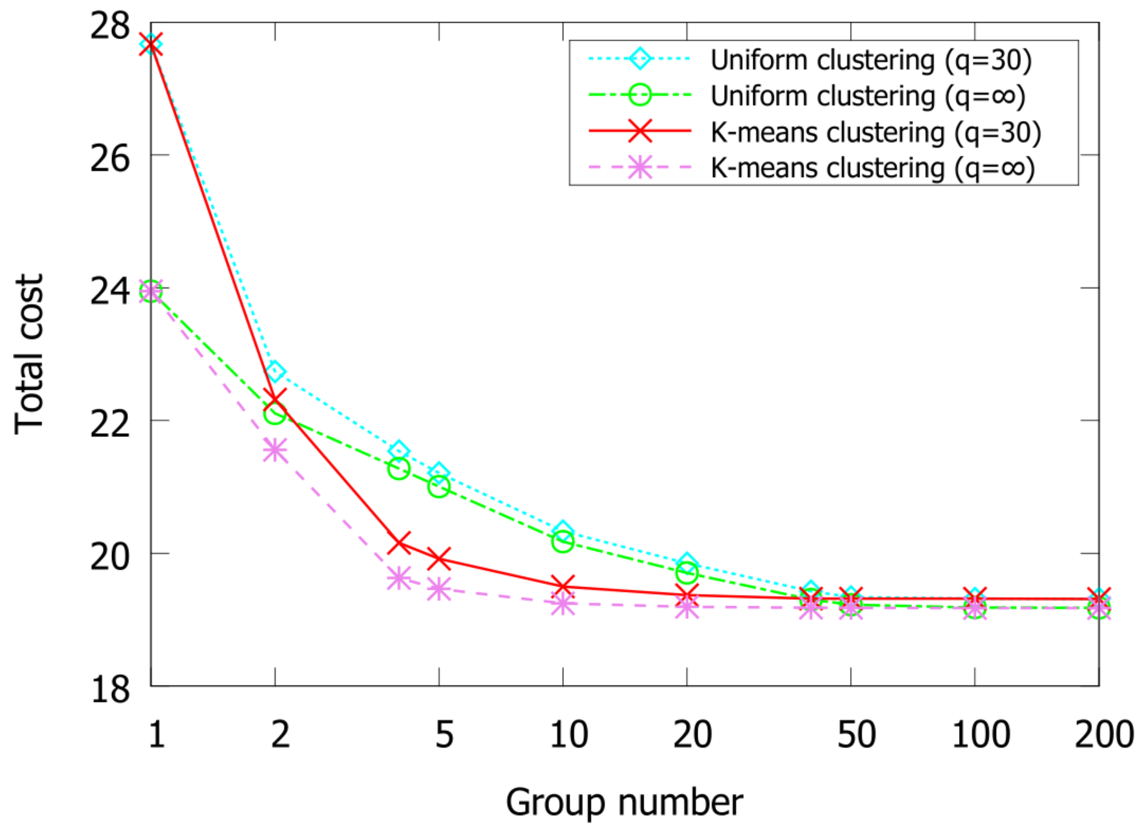
- Group number does not need to be large

$$|M| = 10{,}000$$

Clustering Type	Complexity as $\|M\|$ increases
K-means	$O(\|M\|^2)$
Uniform	$O(\|M\| \log \|M\|)$

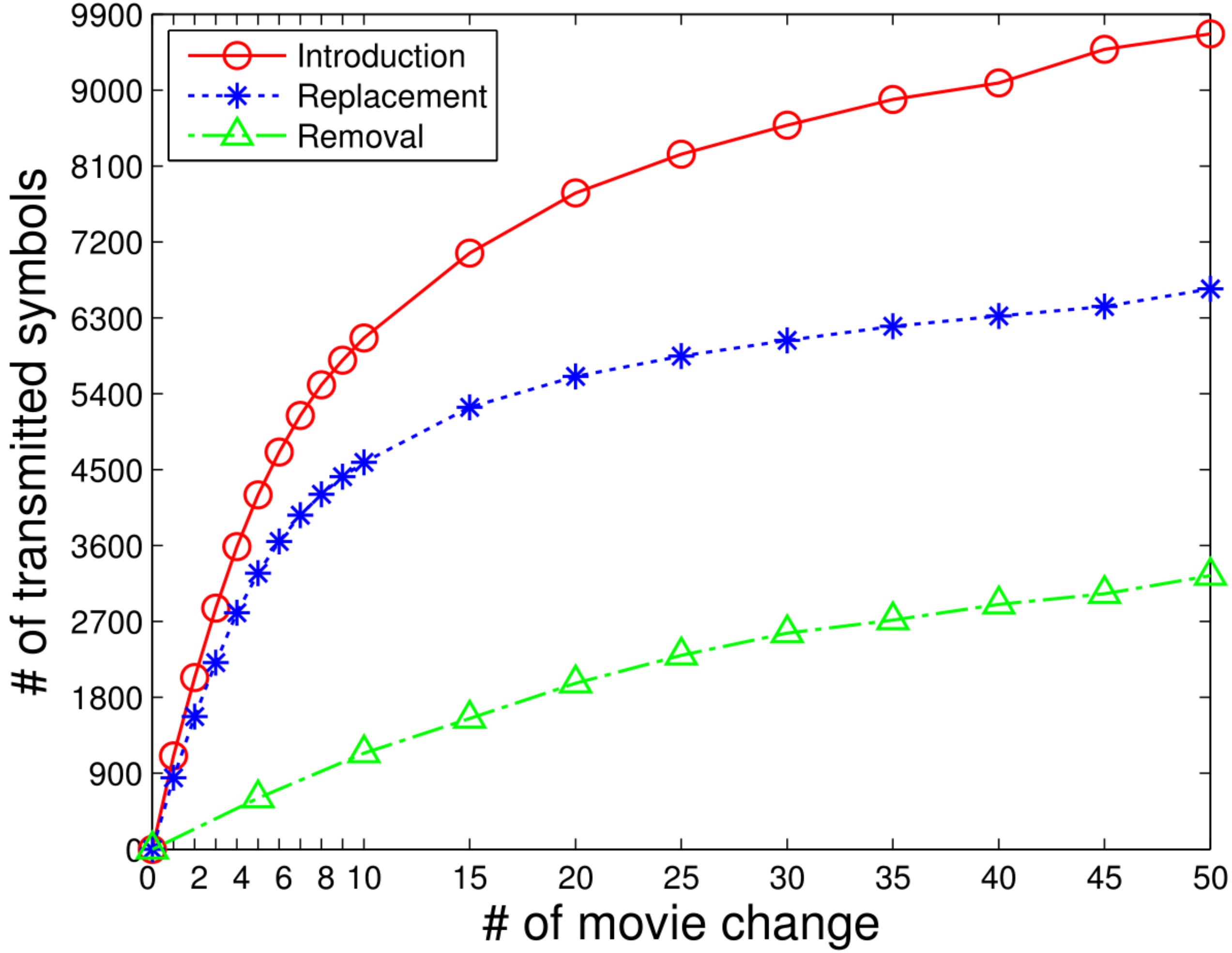
LP with **K-means** clustering running time for $|G|$ = 10 on laptop:

Less than **10** seconds



Efficient On-line re-optimization

- The transmission of symbols increases sub-linearly



Contents

Conclusion

Comprehensive **Cost Model**	• Minimize total deployment cost • Server cost : storage & streaming • Network cost • Content replication & Server selection
Bucket-filling **Asymptotically optimal**	• LP formulation → super optimum solution • Symbol storage & retrieval • Asymptotically optimal discretization
Movie Grouping **K-means Clustering**	• Efficient computation • Little performance Loss • Polynomial time complexity reduction • Efficient online re-optimization
Extensive **Simulation Study**	• Close to optimum performance • Outperform by multiple times

Selected References

- S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proc. IEEE INFOCOM* , Mar. 2010, pp. 1–9.
- A. Nimkar, C. Mandal, and C. Reade, "Video placement and disk load balancing algorithm for VoD proxy server," in *Proc. IEEE Int. Conf. Internet Multimedia Services Archit. Appl.*, Dec. 2009, pp. 1–6.
- S. Zaman and D. Grosu, "A distributed algorithm for the replica placement problem," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 9, pp. 1455–1468, Sep. 2011.
- Y. R. Choe, D. L. Schuff, J. M. Dyaberi, and V. S. Pai, "Improving VoD server efficiency with bittorrent," in *Proc. MULTIMEDIA '07*: 15th Int. Conf. Multimedia, New York, NY, USA, 2007, pp. 117–126.
- D. Wu, J. He, Y. Zeng, X. Hei, and Y. Wen, "Towards optimal deployment of cloud-assisted video distribution services," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 10, pp. 1717–1728, Oct. 2013.
- D. Niu, H. Xu, B. Li, and S. Zhao, "Quality-assured cloud bandwidth auto-scaling for video-on-demand applications," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 460–468.
- Y. Zhou, T. Z. J. Fu, and D. M. Chiu, "A unifying model and analysis of P2P VoD replication and scheduling," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 1530–1538.
- Y. Zhou, T. Z. J. Fu, and D. M. Chiu, "On replication algorithm in P2P VoD," *IEEE/ACM Trans. Netw.*, vol. 21, no. 1, pp. 233–243, Feb. 2013.
- B. Tan and L. Massoulié, "Optimal content placement for peer-to-peer video-on-demand systems," *IEEE/ACM Trans. Netw.*, vol. 21, no. 2, pp. 566–579, Apr. 2013.

Thank You

Any Questions?

Appendix: An example of source coding

Suppose we want to code 3 numbers: $a = 5, b = 6, c = 2013$ in to n symbols.

We compute

$$s_1 = a + b + c$$
$$s_2 = a + 2b + 2^2c$$
$$s_3 = a + 3b + 3^2c$$
$$\dots \dots$$
$$s_n = a + nb + n^2c$$

Then, by taking **any** 3 of $s_i, i\epsilon\{1 \dots n\}$, we formulate a linear system and **solve** it to get **original** a, b, c.

Appendix: Z_p field algebra

To avoid overflow problem, we use Z_p field algebra for computation

In Z_p field algebra

$$a \ +_p b = (a + b) \bmod p$$
$$a \ *_p b = (a * b) \bmod p$$

If p is a prime number, for every number (except 0), we can find a multiplicative inverse

For example, in Z_5 2 ,and 3 are multiplicative inverses to each other

$$a \ *_5 2 *_5 3 = a$$