An Approximation Algorithm to Maximize User Capacity for an Auto-scaling VoD System

Chang, Zhangyu

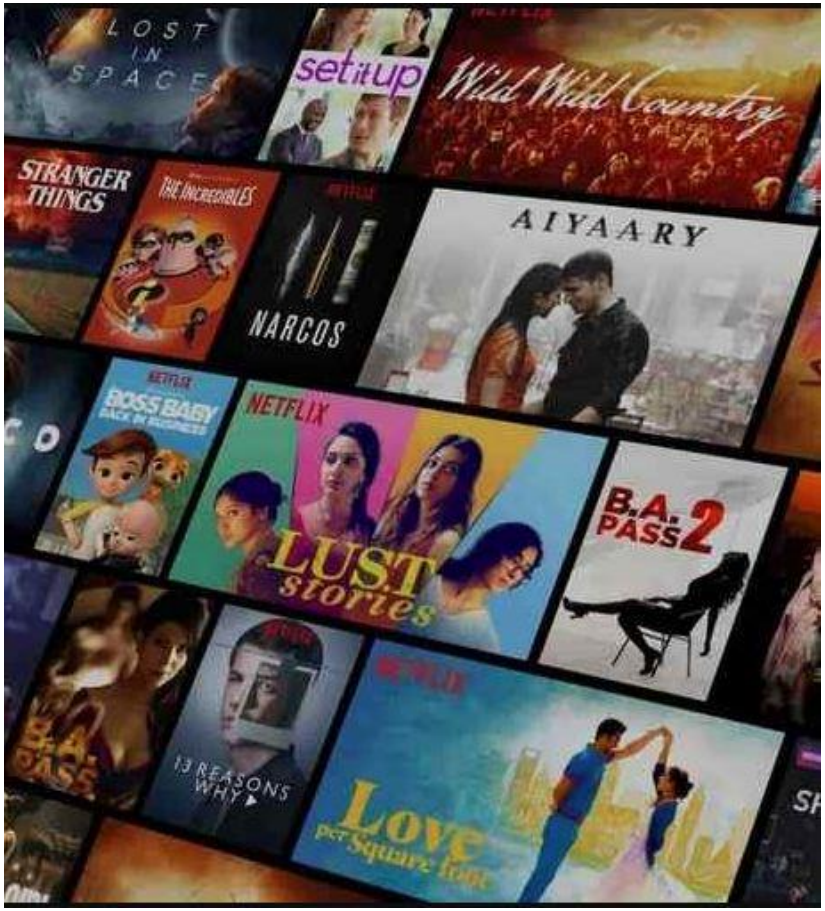
Supervised by Prof. Gary Chan

9 September 2019

Contents

1. **Introduction and Related Work**
2. Problem Formulation and its NP-hardness
3. AVARDO: An Approximation Algorithm
4. Illustrative Trace-driven Experimental Results
5. Conclusion

User Request Pattern for Blockbuster Videos: Stable Popularity, Volatile Traffic

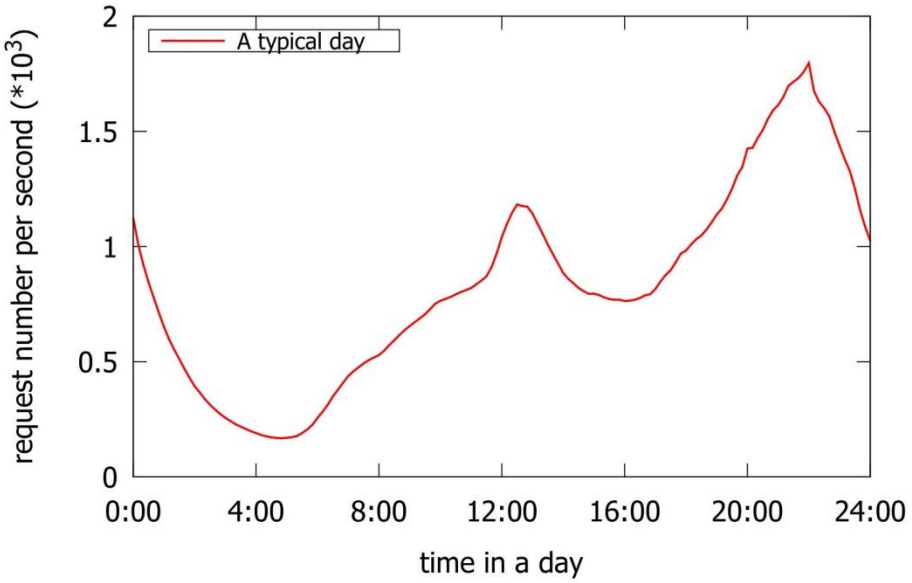


Blockbuster video service (e.g., Netflix) to a large group of audience

- **Video access popularity**: rather stable and predictable over days or weeks
- **User request traffic**: may vary by an order of magnitude in hours

- Statically allocates a fixed number of servers is not efficient.
- **Auto-scaling** can meet the demand in a **timely** and **cost-effective** manner.

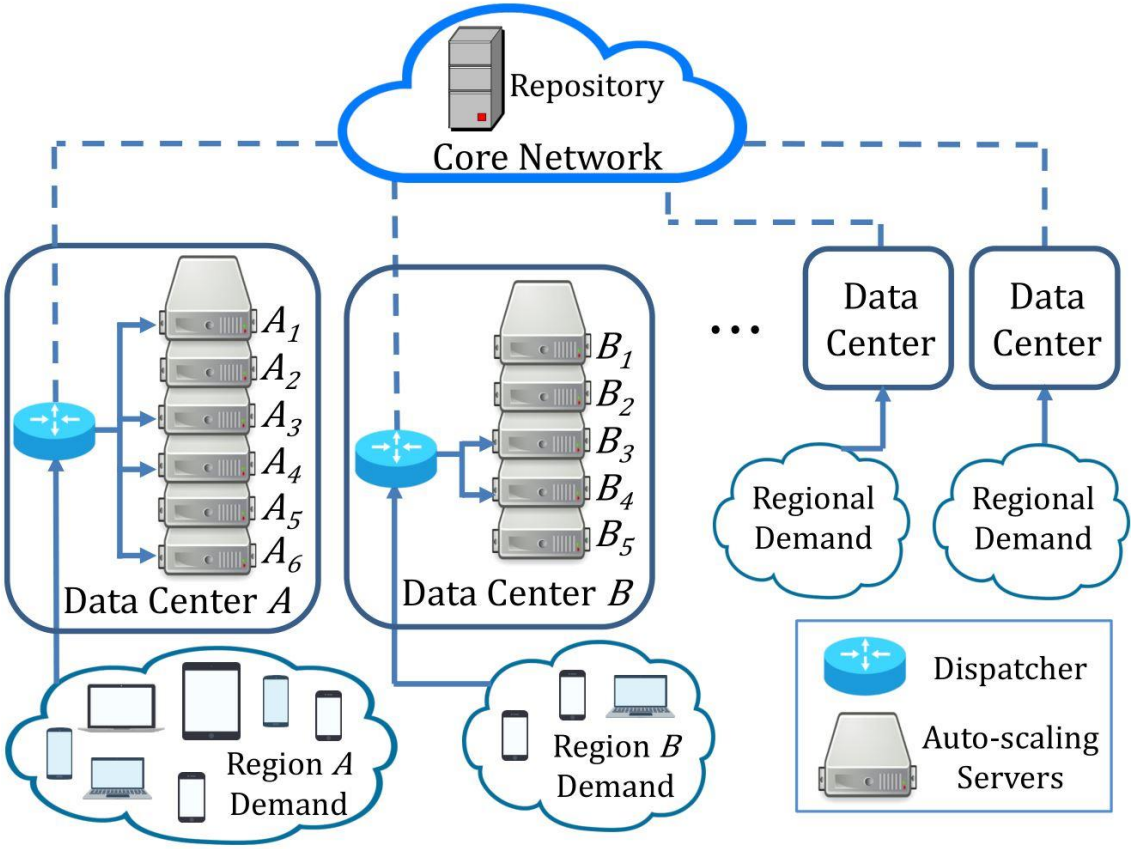
Request Rate Over a Typical Day



User request of a leading video service website in China over a day

- Blockbuster videos have rather stable and predictable over days or weeks (cf. UGC).
- Popularity remains quite stable (varies less than 10%) over a day.
- Request traffic may vary by an order of magnitude over merely hours.
- Auto-scaling is a solution to meet demand in a timely and cost-effective manner.

A Typical Auto-scaling VoD Cloud



A video cloud consisting of auto-scaling VoD data centers.

Auto-scaling Server

- Server has a certain storage and streaming capacity
- Server can be activated or deactivated in a short time
- Homogeneous servers
- Activating server according to incoming traffic

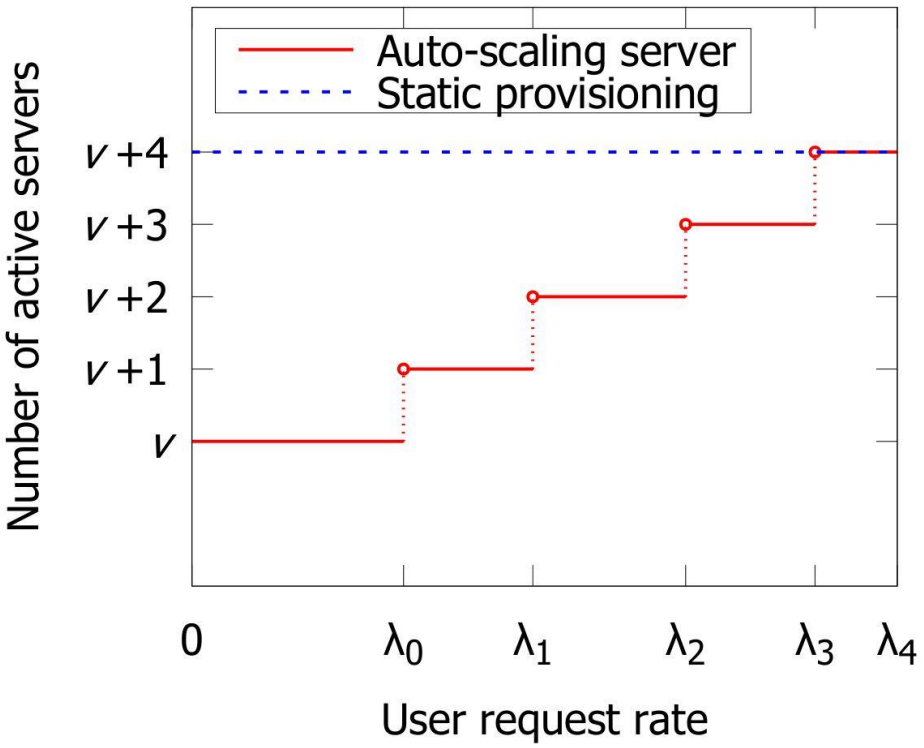
Traffic Dispatcher

- Distribute request to an active server with the video
- Otherwise to core network

Video Block

- Blocks have the same size
- Partition large video into blocks
- Video block is only for management purpose (cf. DASH segments)

Maximizing the User Request Rate Threshold



Request rate	Increase	Decrease
Auto-scaling level	Increase	Decrease
Active servers	More	Less

- The total block request rate λ (requests/second) is mapped to an **auto-scaling level** i (i=0, 1, 2, ...).
- Auto-scaling level i has a request rate threshold λ_i with a predefined set of active server V_i. V_i contains all the video blocks (at least one replica for each block).
- When $\lambda_i < \lambda \leq \lambda_{i+1}$, servers in V_{i+1} are activated.
- Let $|V_0| = \nu$, we have $|V_i| = \nu + i$.
- To minimize the deployment cost, we seek to maximize the **user capacity** supported by the active servers, which is proportional to λ_i.

Optimizing Following Inter-dependent Dimensions

Block Allocation (BA)

- A server has limited storage and cannot store all the video blocks.
- Which blocks should be allocated (or replicated) in each server?
- Servers in V_i shall store at least one replica of each video block.

Server Selection (SS)

- Which servers should be activated (i.e., in V_i) for auto-scaling level i?
- Servers in V_i shall have enough replicas for each video block.

Request Dispatching (RD)

- Some video blocks may be stored on multiple active servers.
- Which server to cater a video block request?
- The dispatcher has to balance the load of each active server.

Challenges: Timescale and Interdependence

Block Allocation (BA)

- Timescale: in day or week
- Videos are pre-allocated (preloaded) in all the servers for SS and RD
- On-the-fly BA is not necessary due to the relatively stable popularity

Server Selection (SS)

- Timescale: in hour
- SS decision should be based on a given BA

Request Dispatching (RD)

- Timescale: in second
- RD decision should be based on a given BA and SS

- We shall jointly optimize these 3 interdependent dimensions.

Contributions

Problem formulation and its NP-hardness

- Study the novel problem: maximize λ_i for each auto-scaling level
- A multi-objective mixed-integer linear programming problem
- We prove the problem is NP-hard
- Traditional static provisioning is a special case of our problem

Stack-based algorithm with proven approximation ratio

- AVARDO: Auto-scaling Video Allocation and Request Dispatching Optimization
- Efficient and closely optimal algorithm with proven approximation ratio
- Stack-based approach with minimum overhead: servers are activated (deactivated) due to the increment (decrement) of auto-scaling level

Extensive trace-driven experimental study based on real-world data

- Trace-driven experiments with real-world VoD data
- Achieve significantly lower optimality gap in active server number (by multiple times) compared with other state-of-the-art schemes

Related Work

Cloud-based VoD architecture resource provisioning

- Yet to consider some important features inside the data center due to model abstraction [1], [2], [3]
- AVARDO complements to these studies by investigating from a more detailed point of view

Content replication in traditional and cloud-based VoD data centers

- Assumes no dynamics within the data center: the server configurations and bandwidth reservation are rarely changed [4]
- Not considered the change of storage and video replication of the auto-scaling servers [5]
- AVARDO optimize for every possible auto-scaling levels

Cloud resources auto-scaling mechanism

- Predict the user demand and improves the performance in the online phase [6], [7]
- Each request or task considered in the problems is served by only one server [8]-[10]
- AVARDO considers BA and RD as some videos are too popular to be served by one server

Contents

1. Introduction and Related Work
2. **Problem Formulation and its NP-hardness**
3. AVARDO: An Approximation Algorithm
4. Illustrative Trace-driven Experimental Results
5. Conclusion

Symbol Used in Formulation

u	The streaming capacity of a server (bits/s)	p^m	Access probability of video block m
c	The storage capacity of a server (bits)	L^m	Average holding time of video block m (in seconds)
f	The file size of block (bits)	b^m	Video streaming rate of video block m (bits/s)
V	The set of all standby servers in data center	$R^m(\lambda)$	Traffic of block m (bits/s) at request rate λ
V_i	The set of active servers at auto-scaling level i	I_v^m	Binary variable indicating server v stores block m
M	The set of all blocks	$r_v^m(i)$	Probability of streaming a request of block m from server v at auto-scaling level i
M_v	The set of video blocks stored in server v		
λ	Total block request rate (requests per second)	μ	Server utilization limit to ensure quality-of-service

Problem Formulation of AVARD: Auto-scaling Video Allocation and Request Dispatching

Objective $\max(\lambda_0, \lambda_1, \ldots \lambda_n)$ → User request rate threshold

Subject to

$$R^m(\lambda) = \lambda p^m L^m b^m, \forall m \in M$$ → Traffic of video block m (bits/s) at request rate λ

Storage

$$\sum_{m \in M(v)} I_v^m f \leq c, \forall v \in V$$ → Server cannot store video blocks beyond its storage

Streaming

$$r_v^m(i) \leq I_v^m, \forall v \in V_i, m \in M$$ → Server can serve the traffic of a block only if it has this block

$$\sum_{v \in V_i} r_v^m(i) \geq 1, \forall m \in M$$ → All the user request for each video block shall be served

QoS

$$\sum_{m \in M} r_v^m(i) R^m(\lambda_i) \leq \mu u, \forall v \in V_i$$ → The utilization of the streaming capacity of every server should not exceed a certain limit μ

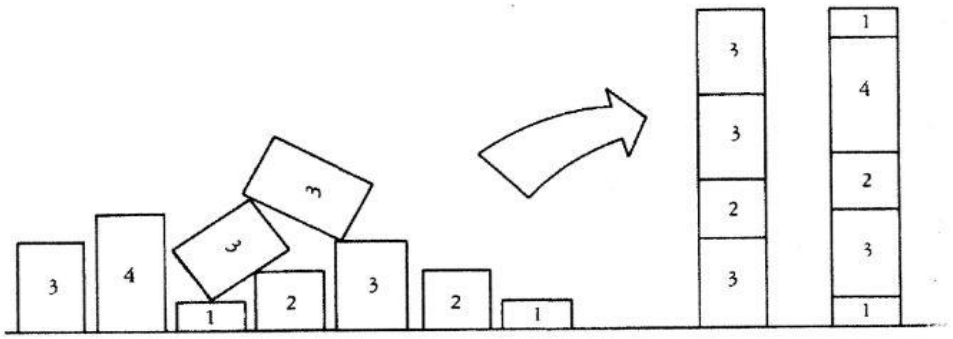
NP-Hardness of AVARD Problem

The **Partition Problem**: (NP-complete)

Whether a given multiset $S = \{s_1, s_2, \ldots, s_n\}$ of n positive integers can be divided into two subsets S_1 and S_2 such that the sums of the numbers in S_1 and S_2 are the same.

The **AVARD** problem is **NP-hard**

- The **Partition Problem** is reducible to our **AVARD** optimization problem.
- Considering that:
  - The Auto-scaling VoD system has only auto-scaling level 0.
  - We have 2 servers with storage n and streaming capacity $s/2 + n$.
  - We have 2n videos with $f = 1$.
  - Half of videos have $R^m = s_m + 1$; the other half videos have $R^m = 1$.



Contents

1. Introduction and Related Work
2. Problem Formulation and its NP-hardness
3. **AVARDO: An Approximation Algorithm**
4. Illustrative Trace-driven Experimental Results
5. Conclusion

Additional Symbol Used in Algorithm

v_i	The server to activate when auto-scaling level goes from $i-1$ to i (i.e., $V_i = V_{i-1} \cup \{v_i\}$)	G	The set of video clusters
		$G(v)$	The set of video clusters on server v
P^m	Streaming ratio of video block m	G_k	The set of video clusters that have k replicas
N^m	Number of replicas for block m stored in V_0	$P(g)$	Total streaming ratio of replicas in cluster g
N_T	Number of replicas can be stored in V_0	$C(g)$	Storage capacity used for cluster g
N_A	Number of surplus replicas in V_0 (i.e., $N_T - \|M\|$)	q_g^m	Probability of streaming a request of block m from cluster g at auto-scaling level 0
σ^m	Average replica streaming ratio of block m		
σ	Average replica streaming ratio threshold	λ_{op}	Theoretical upper limit of λ threshold

AVARDO: Approximation Algorithm for an Auto-scaling Video-on-Demand System

- **A**uto-scaling **V**ideo **A**llocation and **R**equest **D**ispatching **O**ptimization
- Jointly optimize *Video Allocation*, *Server Selection*, and *Request Dispatching*

AVARDO has a stack-based server selection scheme

- Consider the set of active servers as a stack
- Push (activate) or pop (deactivate) a server in an orderly sequence

Preprocessing: Block Replication and Clustering

- Simplify the algorithm by putting the video blocks into clusters.
- Each cluster has the same *file size* and generates same *user traffic.* Thus, it can be treated as a mega video file.

1. The *block replication* step decides how many replicas are required for a video block (i.e., N^m).
2. The *replica clustering* step decides which replicas are in a cluster (i.e., g).

Block Allocation and Request Dispatching

1. Consider cluster allocation for servers in V_0.
2. Consider server v_i incrementally.
3. Given request dispatching at auto-scaling level i.

Preprocessing Stage: Block Replication

$$P^m = \frac{p^m L^m b^m}{\sum_{m \in M} p^m L^m b^m}, \forall m \in M$$

Streaming Ratio P^m is proportional to the traffic of video block m

The Block replication is a popularity-based scheme (in terms of P^m)

1. The least popular block has at least one replica in V_0 (i.e., $N^m \geq 1$).
2. For the most popular block m, each server has at most one replica (i.e., $N^m \leq \nu$).
3. For the other blocks, N^m is proportional to P^m .

$$N^m = \begin{cases} \nu, & \text{if } P^m > \nu\sigma, \\ \lceil P^m/\sigma \rceil, & \text{if } \sigma < P^m \leq \nu\sigma, \\ 1, & \text{if } P^m \leq \sigma. \end{cases}$$

→ fully replicated blocks

→ partially replicated blocks

Average replica streaming ratio threshold σ

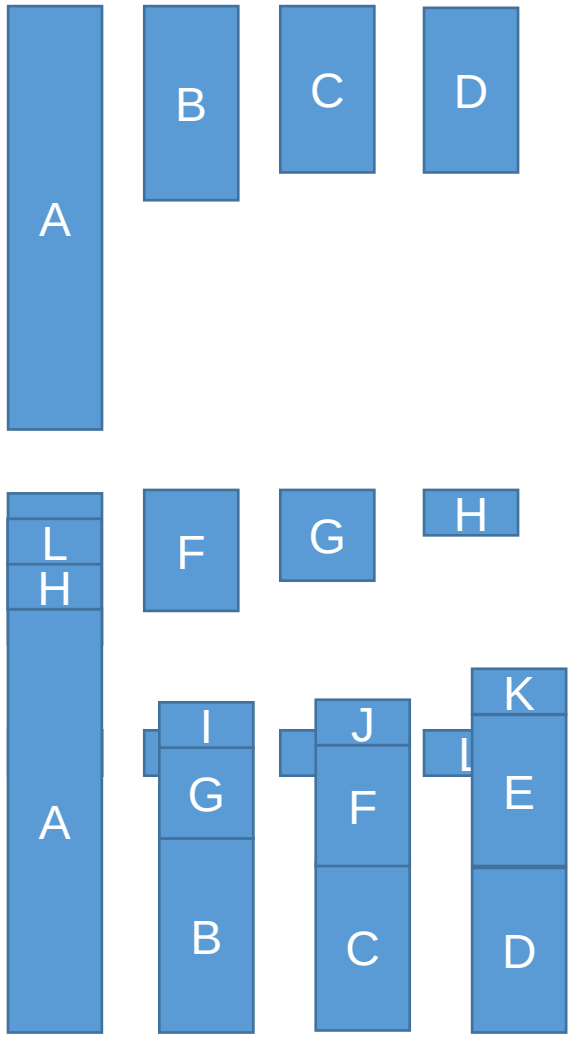
- A smaller σ will increase the number of video replicas.
- We find the smallest possible σ through binary search.

Preprocessing Stage: Replica Clustering

Algorithm 1: AVARDO replica clustering

Initialization: $P(g) = 0$, $C(g) = 0$, $\forall g \in G$;
Put all partially replicated replicas into priority queue $\mathbb{Q}$;
while $\mathbb{Q} \neq \emptyset$ **do**
    Pop top v^2 replicas with max σ^m from $\mathbb{Q}$;
    Put these v^2 replicas into priority queue $\mathbb{Q}_m$;
    while $\mathbb{Q}_m \neq \emptyset$ **do**
        Pop the replica m with max σ^m from $\mathbb{Q}_m$;
        Pop the cluster g with min $P(g)$ from $\mathbb{Q}_g$;
        Store a replica m in g: $g \leftarrow m$;
        Update parameters: $P(g)$ += σ^m, $C(g)$ += f;
    end
end

Objective: Each cluster has the same *file size* and generates similar *user traffic.*



Block Allocation and Request Dispatching

For auto scaling level $i > 0$, we write $i = kv + j$ such that $k \geq 0$ and $1 \leq j \leq v$.

Block Allocation

1. All the servers shall store fully replicated blocks.
2. For $v \in V^0$, we distribute the v^2 clusters into v servers such that each server $v \in V^0$ stores v unique clusters.
3. For server v_i such that $i \leq v$, it shall pick one unique cluster from every server $v \in V^0$ where the cluster has not been picked by the other server v_l such that $l \leq v$.
4. For server v_i such that $i = kv + j$ with $k \geq 1$, we let $G(i) = G(j)$ (i.e., server v_i and v_j have the same block replication).

Traffic Dispatching

We first consider the $i = 0$ case:

$$q_g^m = \begin{cases} 1/N^m, & \text{if } m \in g, \\ 0, & \text{otherwise.} \end{cases}$$

$$r_v^m(0) = \sum_{g \in G(v)} q_g^m, \forall m \in M, v \in V_0.$$

We then consider the $i > 0$ case:

- For the servers $v \in v_1, \ldots, v_i$, we have

$$r_v^m(i) = \frac{v}{v+i} \sum_{g \in G(v)} q_g^m, \forall m \in M.$$

- For the servers in $v \in V^0$, denoting $G_x = G(v) \cap G_{k+2}$ and $G_y = G(v) \cap G_{k+1}$, for all $m \in M$ we have

$$r_v^m(i) = \frac{j}{v+i} \sum_{g \in Gx} q_g^m + \frac{v+j}{v+i} \sum_{g \in Gx} q_g^m.$$

Optimality and Time Complexity

Time Complexity: $O(|M|\log|M| + |M||V|^2)$

- Searching for σ can be done in $O(|M|)$.
- The major component of clustering is to get the replicas from the priority queue. The time complexity is thus $O(|M|\log|M|)$.
- Computing each $r_v^m(i)$ requires constant time. Total time is $O(|M||V|^2)$.

Theoretical Optimality Gap of AVARDO: $\nu^2\sigma$

- Lemma 1: σ is less than $1/N_{\mathrm{A}}$.
- Lemma 2: For every video cluster $g \in G$, its streaming ratio $P(g)$ is no more that $1/\nu^2 + \sigma$.
- The optimality gap, given by $\lambda_{\mathrm{op}}/\lambda - 1$, is no more than $\nu^2\sigma$.
- ν is proportional to video number, and σ is proportional to block size f.

Upper bound of optimality gap in real-world setting: less than 1%

- A nowadays video server can store more than 10^5 videos ($\sigma < 10^{-5}$).
- For auto-scaling level 0, 30 servers are more than enough ($\nu \leq 30$).
- We can further reduce σ by partitioning the video files into smaller blocks.

Contents

1. Introduction and Related Work
2. Problem Formulation and its NP-hardness
3. AVARDO: An Approximation Algorithm
4. **Illustrative Trace-driven Experimental Results**
5. Conclusion

Simulation Environment

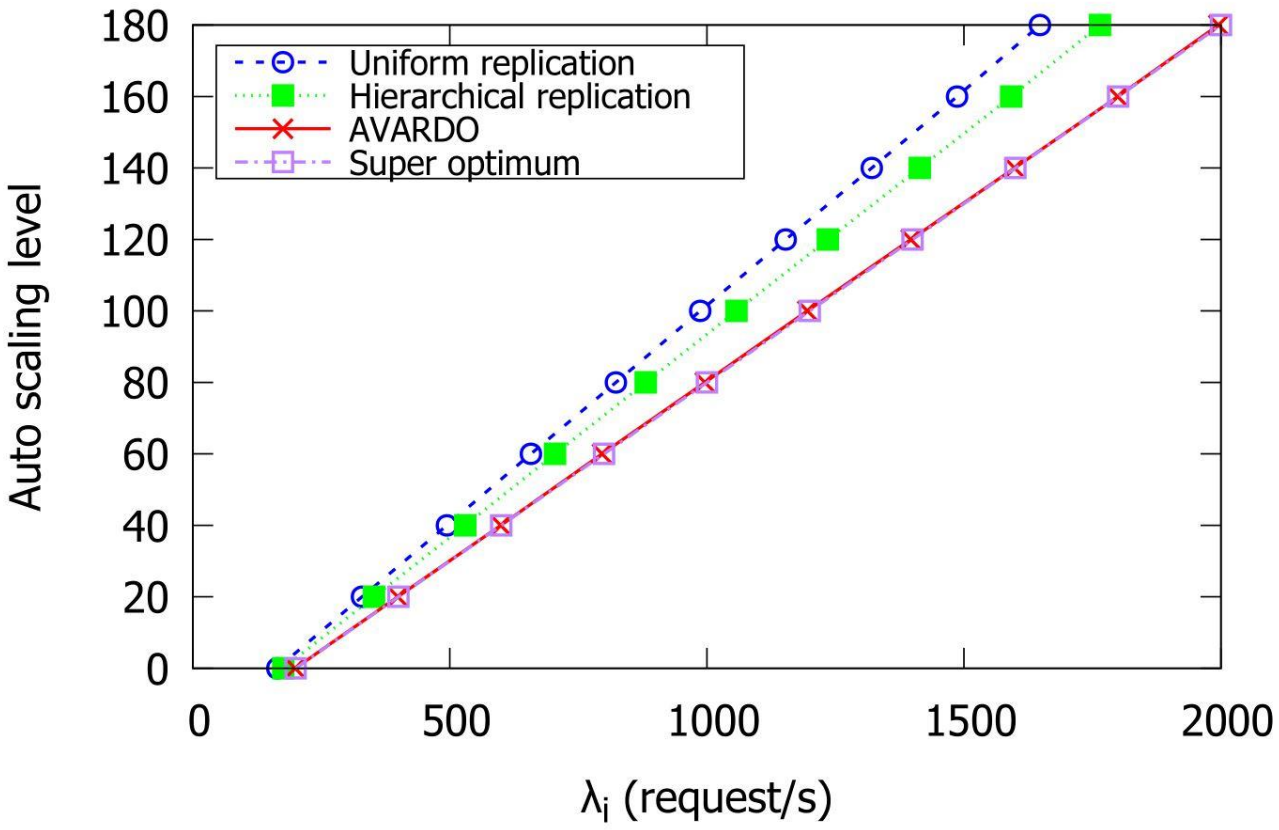
Parameter	Baseline value
• Number of blocks $\|M\|$	around 3×10^6
• block request rate λ (requests/s)	2,000
• Number of blocks in a server c/f	6×10^5
• Server streaming capacity u (Gbps)	25
• Server utilization limit μ	0.9

- The real-world data trace is from a leading video service website in China over 2 weeks.
- We partition the videos into the blocks of the same size of 100MB.
- When a video has multiple resolutions and bit rates, we treat them as multiple video files.

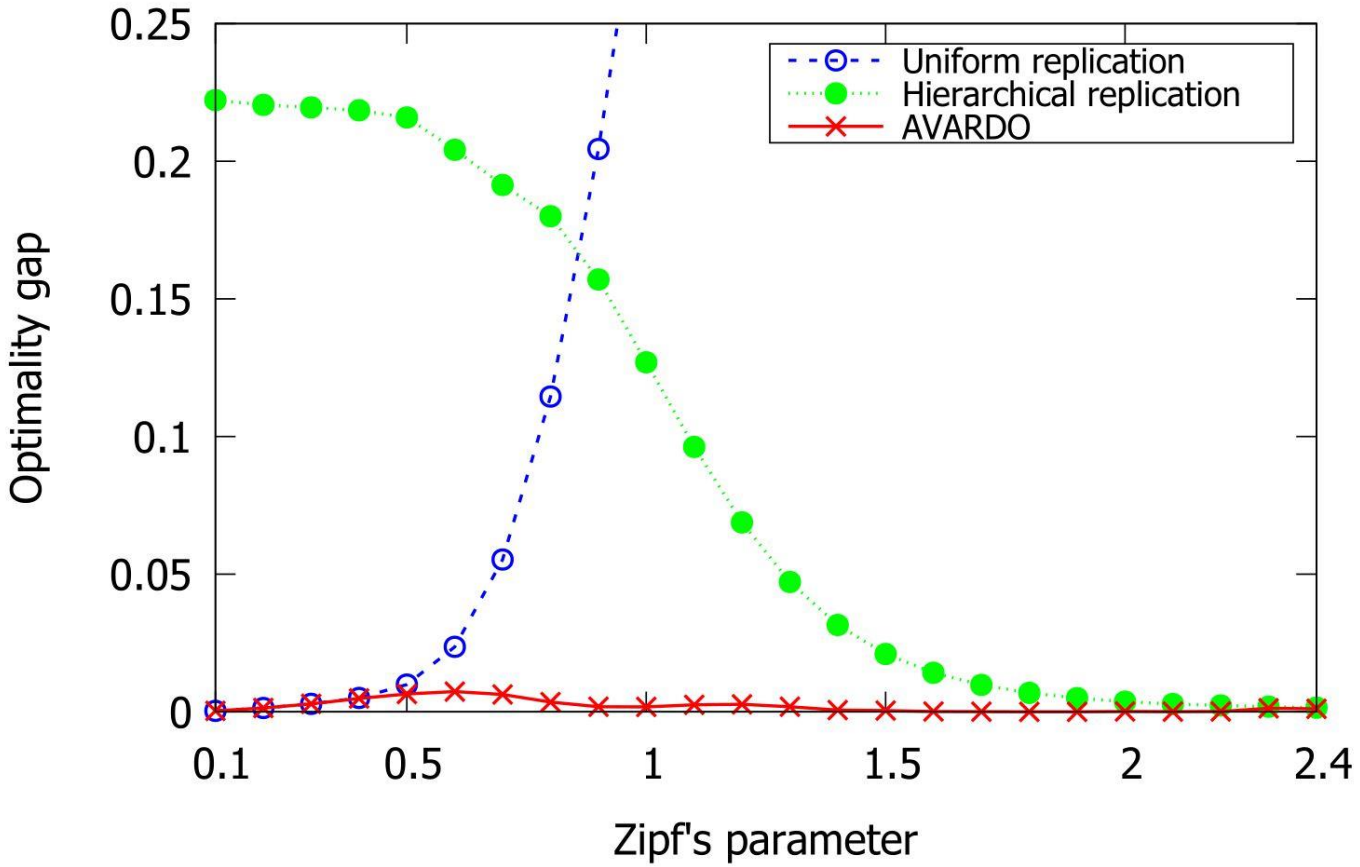
Performance Metrics

Performance Metrics	
• Request rate threshold λ_n	Optimization objective of AVARDO
• Optimality gap of λ_n	Difference between scheme performance and the theoretical performance bound
• Number of active servers	Operation cost over a given time period
• Fairness of active server utilization	Jain's Fairness Index, which is between 0 and 1 (a higher index indicates a fairer load sharing)
Comparison Schemes	
• Uniform replication	Every video has the same number of replicas. The videos are randomly stored in the servers.
• Hierarchical popularity replication	2 types of server: repository and cache. Repository servers collaboratively store all. Caches only store popular videos.
• Super optimum	Serves as the theoretical performance bound. We assume that a video can be partitioned infinitesimally (i.e., $f \to 0$).

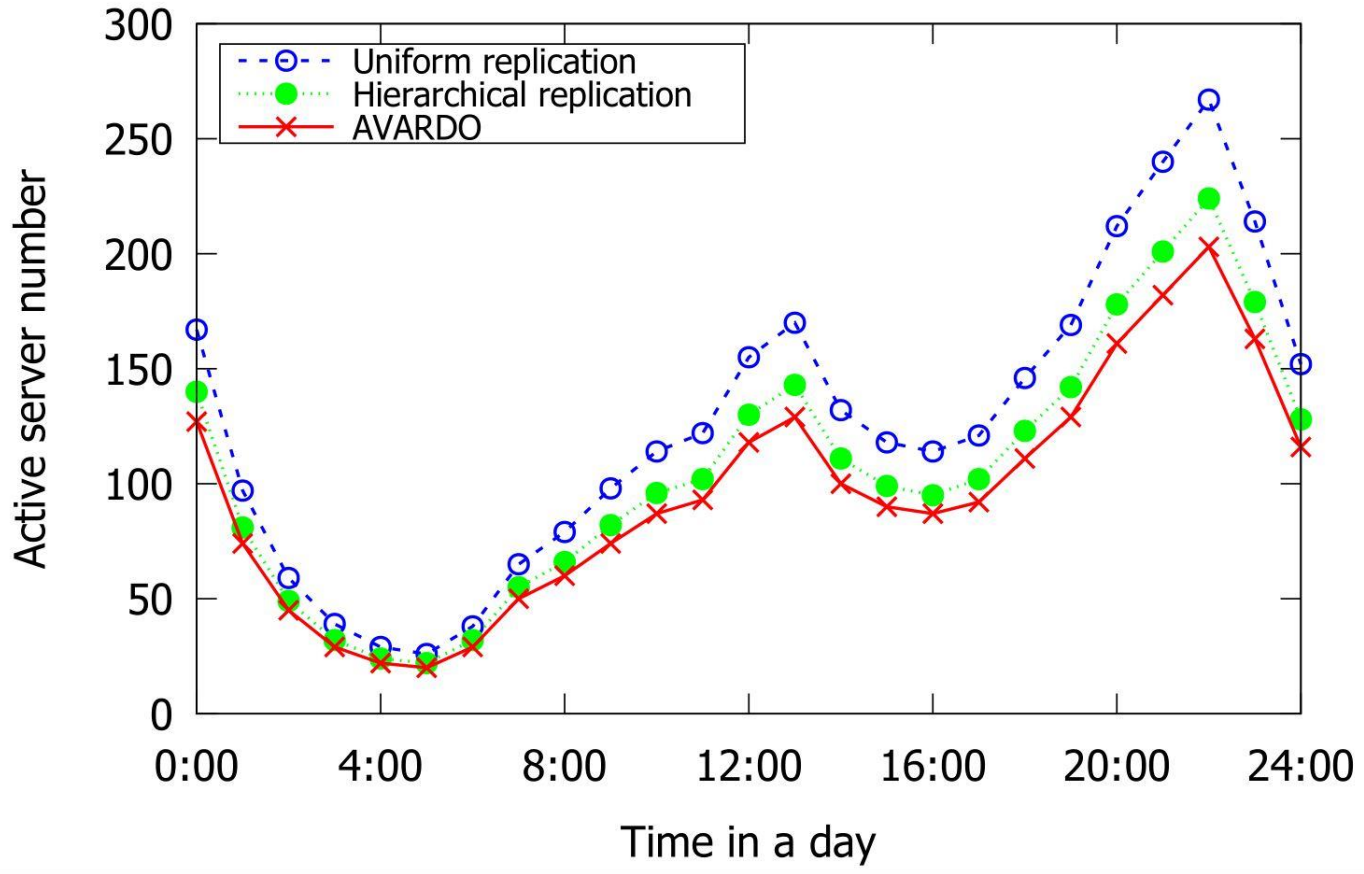
Asymptotic Optimality



Outperform State-of-the-art Schemes



Closely Optimal Over a Typical Day



Contents

1. Introduction and Related Work
2. Problem Formulation and its NP-hardness
3. AVARDO: An Approximation Algorithm
4. Illustrative Trace-driven Experimental Results
5. **Conclusion**

Conclusion

Problem Formulation and NP-hard analysis	• Formulation the problems as multi-objective mixed-integer linear programming • Prove that AVARD problem is NP-hard
Stack-based approximation algorithm with provable performance	• AVARDO: Auto-scaling Video Allocation and Request Dispatching Optimization • A novel and closely-optimal approximation algorithm with proven optimality gap • Stack-based approach to minimize overhead
Extensive trace-driven experimental results	• Real-world VoD data traces • Outperform the state-of-the-art schemes • Significantly lower optimality gap (often 1/20)

Selected References

[1] J. Yang, Z. Yao, B. Yang, X. Tan, Z. Wang, and Q. Zheng, “Software-defined multimedia streaming system aided by variable-length interval in-network caching,” IEEE Transactions on Multimedia, vol. 21, no. 2, pp. 494–509, Feb 2019.

[2] E. Bourtsoulatze, N. Thomos, J. Saltarin, and T. Braun, “Content-aware delivery of scalable video in network coding enabled named data networks,” IEEE Transactions on Multimedia, vol. 20, no. 6, pp. 1561– 1575, June 2018.

[3] J. Tang, X. Tang, and J. Yuan, “Traffic-optimized data placement for social media,” IEEE Transactions on Multimedia, vol. 20, no. 4, pp. 1008–1023, April 2018.

[4] H. Zhao, Q. Zheng, W. Zhang, B. Du, and H. Li, “A segment-based storage and transcoding trade-off strategy for multi-version VoD systems in the cloud,” IEEE Transactions on Multimedia, vol. 19, no. 1, pp. 149–159, Jan 2017.

[5] G. Gao, Y. Wen, W. Zhang, and H. Hu, “Cost-efficient and QoS-aware content management in media cloud: Implementation and evaluation,” in Proc. International Conference On Communications (ICC). IEEE, 2015, pp. 6880–6886.

[6] L. De Cicco, S. Mascolo, and V. Palmisano, “QoE-driven resource allocation for massive video distribution,” Ad Hoc Networks, 2019.

[7] W. Iqbal, A. Erradi, and A. Mahmood, “Dynamic workload patterns prediction for proactive auto-scaling of web applications,” Journal of Network and Computer Applications, vol. 124, pp. 94 – 107, 2018.

[8] C. Valliyammai and R. Mythreyi, “A dynamic resource allocation strategy to minimize the operational cost in cloud,” in Emerging Technologies in Data Mining and Information Security, A. Abraham, P. Dutta, J. K. Mandal, A. Bhattacharya, and S. Dutta, Eds. Springer Singapore, 2019, pp. 309–317.

[9] J. Nino-Mora, “Resource allocation and routing in parallel multi-server ˜ queues with abandonments for cloud profit maximization,” Computers and Operations Research, vol. 103, pp. 221 – 236, 2019.

[10] H. Zhao, J. Wang, Q. Wang, and F. Liu, “Queue-based and learning-based dynamic resources allocation for virtual streaming media server cluster of multi-version VoD system,” Multimedia Tools and Applications, Apr 2019.

Thank You!

Any Questions?