Efficient Replication in Multi-regional Peer-Supported VoD Systems

Yuval Rochman
Tel-Aviv University
Tel Aviv, Israel
yuvalroc@gmail.com

Hanoch Levy
Tel-Aviv University
Tel Aviv, Israel
hanoch@cs.tau.ac.il

Eli Brosh
Vidyo
New Jersey, USA
eli@vidyo.com

1. INTRODUCTION

Video-on-Demand (VoD) services have experienced an explosive growth in recent years, and are currently used by millions of users [3]. In traditional server-based VoD, end-user terminals download video contents from a video provider upon user's request. Since server-based systems are non-scalable and subject to server bottlenecks, a peer-supported approach, where some of the contents is provided by peer servers, has emerged as an effective solution for that problem. In this approach, peers (user terminals) use their storage space and upload bandwidth to replicate video content and serve the content to other peers when needed.

A fundamental design issue in peer-supported VoD systems is to determine the right *replica placement (allocation) strategy* so as to make best use of the upload capacity of peer servers. While the literature deals with P2P replica placement, it is mainly geared towards file sharing systems. For example, optimized the network bandwidth usage in replication; while [4] maximized file availability. These works pay little attention to the bandwidth limitation of peers, a key concern for VoD.

To this end, we consider a *multi-region* peer-supported VoD service model where peer servers are located in different regions, and serving a video across regions is more expensive than within a region. The demand distribution can vary across the regions. Service can be granted by dedicated servers as a fall back, but at higher cost. The system operations requires taking care of two tasks, which are best described as if done in two stages: First, video replicas are placed *by the central server* at the peer servers. Then, video requests submitted by the users are assigned-to and provided-by the proper peer servers. The multi-region model is a natural fit for the network of a large content provider. For example, cable operators use video hub offices in each metropolitan area (all inter-connected) to serve the local subscribers [1, 2].

The highly variable demand distribution (some videos are highly popular while very many others are lowly popular, following heavy tail distributions, see [3]) poses challenges on the system designer regarding the number of replicas that should be placed in the system; the use of a simple assignment, like the *proportional mean* (proved to be *optimal placement* under some conditions, see e.g. [5]) may be quite inefficient as shown in the following simple example: Consider a system with two video types, A and B. The demand for A is deterministic 100 (mean=100), while for B it is highly variable: $100k^2$ with probability $1/k$ and 0 with probability $1-1/k$ (mean is $100k$). Assume the system has room for 100 movie replicas. How many replicas should one store of each movie? If one follows the proportional mean placement one would place $100k/(1+k)$ copies of B and $100/(1+k)$ of A; The expected number of movies served will be then $200/(k+1)$. If, in contrast, one places 100 replicas of A , the expected number of movies served will be 100, a factor of $(k+1)/2$ *better* than that of proportional mean, a performance ratio that is *unbounded*. The problem is further complicated since there are combinatorially many allocation options to consider.

Our objective is to address this challenge and derive optimal replica allocations that minimize the system-wide video servicing costs under an arbitrary (stochastic) demand distribution. Our main result is that optimal replica placement in multi-region environments is of *max percentile* nature, namely, it is based on the tail distribution of the demand. The approach proposed, termed Max Percentile Allocation, and the results derived can be used either to practically manage peer-supported VoD systems or as theoretical bounds to evaluate the relative performance of alternative simpler approaches, typically based on the demand mean. Such a bound is especially important in light of the unbounded performance ratio demonstrated in the example given above.

Full theorem proofs are provided in a forthcoming paper.

MAMA 2012 London, United Kingdom

2. MODEL ASSUMPTIONS

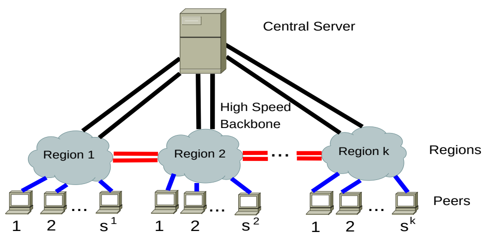

Figure 1: System topology

We consider a topology as depicted in Fig.1, consisting of k regions and one central server. Each region contains a set of *client terminals* that request movies and a set of *peer servers* (in some cases one user terminal serves both as a client and as a server) that store movie replicas and can deliver them to the requesting clients. The set of movies is enumerated by $\{1, 2, 3..., m\}$. Each movie can be replicated, and the total number of replicas to be stored in the system is s. The regions are connected to each other via a high speed backbone network. Thus, each client can retrieve movie replicas from its local region, or from a remote one.

An optimal replica placement algorithm will place replicas across the regions, satisfying storage and bandwidth constraints at the peer servers, so as to minimize movie replica servicing costs. Given that a peer server can store multiple movies (typically a few), deriving the optimal solution is NP-hard, even for the simple case of a single-region. We thus restrict ourselves to consider allocations in environments where peers can store only a single movie. Such a solution can be used to provide guidelines/heuristics on the design of efficient allocations schemes for peer environments with > 1 storage.

We assume the following conditions on the system:

- The system is managed by a central server which decides on the replica allocation in the various regions[1]. It also controls the assignment of user requests to peer servers who have the requested content.
- At any given time, a peer server can only serve a single video request.
- The demand (number of requests for a movie) is static reflecting the demand at peak hours. The demand is denoted by N_i^j, which is a random variable denoting the number of requests for movie i made by peer-clients in region j. Note that $\{N_i^j\}_{j=1}^k$ are not necessarily identically distributed, and not necessarily independent of each other.

The cost of serving a request is dependent on where the request is granted. A request can be granted by a peer in its own region, by a peer in a remote region, or by the central server. In these cases the cost will be C_{loc} (local cost), C_{rem} (remote cost), C_{ser} (central server), respectively. We assume that the costs obey $C_{ser} \geq C_{rem} \geq C_{loc}$, reflecting local costs cheaper than remote ones.

Denote g_{ser}, g_{rem} and g_{loc} the number of requests granted (served) from the server, from a remote region or from a local region, respectively. The system request service cost is:

$$C = C_{ser} \cdot g_{ser} + C_{rem} \cdot g_{rem} + C_{loc} \cdot g_{loc}.$$

Let L_i^j, L_i denote the number of movie i replicas placed in region j and in the whole system, respectively; The set $L = \{L_i^j\}$ is called a *movie replica allocation* or simply an *allocation*.

The problems addressed in this paper can be stated formally as follows:

1. The *assignment problem*: Given an allocation, $L = \{L_i^j\}$, a **deterministic** demand denoted by n_i^j, and the service cost parameters $C_{ser}, C_{rem}, C_{loc}$, assign (match) the stored movie replicas to the demands as to minimize the service cost C.
2. The *allocation problem*: Given the **random** movie demand distributions $\{N_i^j\}$, the service cost parameters $C_{ser}, C_{rem}, C_{loc}$, and a matching algorithm solving the assignment problem, determine the replica allocations $L = \{L_i^j\}$, of each movie in each region[2] as to minimize the expected cost $E[C]$.

[1]The underlying assumption in this paper is that a sufficient number of peer servers is available to store the replicas in any region.

3. TRANSFORMING THE COST FUNCTION

We transform the cost function into a revenue function as done next. This is essential to facilitate the analysis.

Claim 3.1. *The following holds:*
(1) A matching algorithm M solves the assignment problem iff M maximizes the function:

$$R = R_{glo} \cdot g_{glo} + R_{loc} \cdot g_{loc}, \qquad (1)$$

where we set $R_{glo} \doteq C_{ser} - C_{rem} \geq 0$, $R_{loc} \doteq C_{rem} - C_{loc} \geq 0$, and g_{loc} denotes the number of requests granted locally within the regions and g_{glo} denotes the number of requests granted by either local or remote regions.
(2) An allocation L solves the allocation problem iff the allocation maximizes $E(R)$.

We denote Eq. (1) to be the *revenue objective function.*

4. THE ASSIGNMENT PROBLEM

Algorithm 1 below, called the *matching algorithm*, is an optimal and greedy algorithm tailored for the assignment problem.

Algorithm 1 The matching algorithm

```
Require: An allocation of peer-servers ,$L = \{L_i^j\}$ and demand $n_i^j$, $i = 1, ..., m$, $j = 1, ..., k$.
1: for all movie $i$ do
2:   for all region $j$ do
3:     Take $\min(L_i^j, n_i^j)$ requests from the $j^{th}$ region of the $i^{th}$ movie, and match them to $\min(L_i^j, n_i^j)$ peer-servers in the $j^{th}$ region, containing the $i^{th}$ movie.
4:   end for
5:   Let $n_i^{rem}$ be the number of movie $i$ requests, which we did not match in Step 3, and let $L_i^{rem}$ be the number of unmatched movie $i$ peer-servers.
     Then, match the remaining $\min(n_i^{rem}, L_i^{rem})$ requests to the remaining $\min(n_i^{rem}, L_i^{rem})$ peer-servers.
6: end for
```

An efficient implementation of the matching algorithm, operating in linear time, $O(s + n)$, is given in our article. We prove that the algorithm derives an optimal assignment between the demands and the peer-servers, and that it yields the following expression for its revenue:

$$R = R_{glo} \sum_{i=1}^{m} \min(L_i, n_i) + R_{loc} \sum_{i=1}^{m} \sum_{j=1}^{k} \min(L_i^j, n_i^j). \qquad (2)$$

5. UNLIMITED ALLOCATION PROBLEM

Using Claim 3.1 and Eq. (2), the unlimited problem can be stated as the following maximization problem:

$$\text{Find } \max_L E(R^L) \text{ such that } \sum_{i=1}^{m} \sum_{j=1}^{k} L_i^j \leq s,$$

[2]The formulation accounts only for the number of replicas in a region since the storage of each server is one, and thus all servers in a region are interchangeable

where the revenue function is:

$$E(R^L) =$$

$$R_{glo}\sum_{i=1}^{m} E_{N_i}(\min(L_i, N_i)) + R_{loc}\sum_{i=1}^{m}\sum_{j=1}^{k} E_{N_i^j}(\min(L_i^j, N_i^j)).$$

The greedy algorithm solving the Unlimited Allocation Problem is called GeMurMap (Generalized Multi Region Max Percentile) and serves as the basis for solving the limited problem (Section 6). The iterative algorithm will assign in every step the "best" movie to the "best" region. More formally, we define the following definitions:

Definition 1. (1) Let L be an allocation. Then the *marginal allocation* of adding movie i_0 to region j_0 is allocation L' such that $L'^{j_0}_{i_0} = L^{j_0}_{i_0} + 1$ and $L'^{j}_{i} = L^{j}_{i}$ for $i \neq i_0$ or $j \neq j_0$. (2) The *marginal revenue* of adding movie i_0 to region j_0 is $\delta^{j_0}_{i_0}(L) = E(R^{L'}) - E(R^L)$.

The GeMurMap algorithm is given below:

Algorithm 2 GeMurMap algorithm

```
1: Initiate an allocation $L = \{L_i^j | 1 \leq i \leq m, 1 \leq j \leq k\}$
   such that $L_i^j = 0$ for every movie $i$ and region $j$.
2: repeat
3:   for all movie $i_0$ do
4:     for all region $j_0$ do
5:       Find $(i_0, j_0) = \arg\max_{(i,j)} \delta_i^j(L)$.
6:       Add replica $i_0$ to region $j_0$.
7:     end for
8:   end for
9: until $\sum_{i=1}^m L_i = s$.
10: return the optimal allocation $L$.
```

In the paper we construct an equivalent very efficient algorithm with complexity of $O(s \log sk + m \log mk)$. This results from the fact that every discrete non-negative random variable satisfies $E(X) = \sum_{k=0}^{\infty} \Pr(X \geq k)$, implying $\delta_i^j(L) = R_{glo}\Pr(N_i \geq L_i + 1) + R_{loc}\Pr(N_i^j \geq L_i^j + 1)$, which can be calculated in $O(1)$.

For proving the algorithm optimality we will use the following definitions:

Definition 2. (1) Let L and L' be two allocations. Then the allocation are called *equivalent* iff the total number of movie i replicas in those allocation is the same, i.e $L_i = L'_i$. (2) Let L be an allocation. Then L is called a *global maxima allocation* if for every equivalent allocation L' we have $E(R^L) \geq E(R^{L'})$.

The *optimal* allocation, which is the allocation with the highest revenue, must be a global maxima allocation; otherwise ,there is an equivalent allocation with a higher revenue.

In the article we prove the following theorem:

THEOREM 5.1. *If GeMurMap adds replica movie i_0, then the result allocation maximized $\sum_{j=1}^{k} E(\min(L^j_{i_0}, N^j_{i_0}))$ among the equivalent allocations. Moreover, in every iteration in GeMurMap we reach a global maxima allocation.*

In the paper we show that the theorem follows from the fact that the function $f_i^j(x) = \Pr(N_i^j \geq x)$ is monotone. We then use Theorem 5.1 and the fact that $\delta_i^j(L)$ (as well as an important variation of $\delta_i^j(L)$) is monotone decreasing to prove that GeMurMap returns the optimal allocation.

6. THE LIMITED ALLOCATION PROBLEM

The Limited Allocation Problem is similar to the unlimited one, with an additional restriction that the number of servers is in region j is limited by s^j. Formally:

$$\text{Find } \max_{L} E(R^L) \text{ such that } \sum_{i=1}^{m} L_i^j \leq s^j \text{ for } 1 \leq j \leq k.$$

In the paper we prove that this problem can be solved via a graph representation $G = (V, E)$ minimum-cost flow problem with $O(s^3)$ vertices. The difficulty is that minimum-cost flow algorithms operate in at least $O(|V|^2)$. And thus the overall implementation is impractical (at least $O(s^6)$).

Thus, we give a heuristic implementation of the allocation algorithm. The algorithm, called Limited GeMurMap, is based on the unlimited algorithm and is given next:

Algorithm 3 Limited GeMurMap algorithm

```
1: Initiate an allocation $L = \{L_i^j | 1 \leq i \leq m, 1 \leq j \leq k\}$
   such that $L_i^j = 0$ for every movie $i$ and region $j$.
2: repeat
3:   for all movie $i_0$ do
4:     for all region $j_0$ such that $\sum_{i=1}^m L_i^j \neq s^j$ do
5:       Find $(i_0, j_0) = \arg\max_{(i,j)} \delta_i^j(L)$.
6:       Add replica $i_0$ to region $j_0$.
7:     end for
8:   end for
9: until $\sum_{i=1}^m L_i = s$.
10: return the optimal allocation $L$.
```

We give an equivalent implementation of the algorithm in $O(s \log sk + m \log mk)$. Although we show that the algorithm is not optimal, we can prove that if the regions are "symmetrical" (i.e $\{N_i^j | 1 \leq j \leq k\}$ are identically distributed for all movie i and $s^j = s/k$ for every region j) then the algorithm returns the optimal allocation.

Other results: In our work we further deal with: 1) Multiple Hierarchies, 2) online behavior, 3) Heuristics for multiple movies in a server, and 4) optimality properties.

7. REFERENCES

[1] M. S. Allen, B. Y. Zhao, and R.Wolski. Deploying Video-on-Demand Services on Cable Networks. In *ICDCS*, Toronto, Canada, June 2007.

[2] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. K. Ramakrishnan. Optimal content placement for a large-scale vod system. In *ACM CoNEXT*, Philadelphia, USA, Dec 2010.

[3] M. Cha, H. Kwak, P. Rodriguez, Y. Y. Ahn, and S. Moon. I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System. In *ACM Internet Measurement Conference*, New York, NY, USA, October 2007.

[4] J. Kangasharju, K. W. Ross, and D. A. Turner. Optimizing file availability in peer-to-peer content distribution. In *IEEE INFOCOM*, Anchorage, Alaska , USA, May 2007.

[5] S. Tewari and L. Kleinrock. Proportional replication in peer-to-peer networks. In *IEEE INFOCOM*, Barcelona, Spain, April 2006.