COPACC: A Cooperative Proxy-Client Caching System for On-Demand Media Streaming *

Alan T.S. Ip[1], Jiangchuan Liu[2], and John C.S. Lui[1]

[1] The Chinese University of Hong Kong, Shatin, N.T., Hong Kong,
{tsip,cslui}@cse.cuhk.edu.hk
[2] Simon Fraser University, Vancouver, BC, Canada,
csljc@ieee.org

Abstract. Proxy caching is a key technique to reduce transmission cost for on-demand multimedia streaming. However, its effectiveness is limited by the insufficient storage space and weak cooperations among proxies and their clients. In this paper, we propose COPACC, a novel cooperative proxy-and-client caching system that combines the advantages of both proxy caching and peer-to-peer (P2P) client communications. We propose a comprehensive suite of protocols to facilitate the interactions among different network entities in COPACC. We also develop an efficient cache allocation algorithm to minimize the aggregated transmission cost of the whole system. The simulation results demonstrate that COPACC achieves remarkably lower transmission cost. Moreover, it is much more robust than a pure P2P system in the presence of node failures.

1 Introduction

Today's Internet has been increasingly used for carrying multimedia traffic, and on-demand streaming for clients is amongst the most popular networked media services. The limited server capacity, however, make efficient and scalable on-demand media streaming a challenging task. To reduce server/network loads, frequently used data is cached at proxies close to clients[1]. Streaming media, particularly those with asynchronous demands, could benefit with a significant performance improvement from proxy caching given their static nature in content and highly localized access interests. Another approach is to generalize the proxy functionalities into every client [2]. Such a P2P paradigm allows economical clients to contribute their storages for streaming. Video data originally provided by a server are spread among clients, thus amplifying the system capacity.

In this paper, we propose COPACC, a novel cooperative proxy-and-client caching system. We leverage the client-side caching to amplify the aggregated cache space and rely on dedicated proxies to effectively coordinate the communications. We develop an efficient cache allocation algorithm together with a comprehensive protocols suite to distribute video segments among the proxies and clients such that the aggregated transmission cost is minimized. As most operations are executed by dedicated proxies, the system is resilient to client failures. We also embed an efficient indexing and searching algorithm for video contents cached across different proxies or clients. COPACC also makes effective use of multicast delivery, which further reduces the cost. The simulation results demonstrate that COPACC achieves remarkably lower transmission cost as compared to proxy-based caching with limited storage space. With the assistance from dedicated proxies, it is much more robust than a pure P2P system. Moreover, It scales well to larger networks, and the cost generally reduces when more proxies and clients cooperate with each other.

* This work is supported in part by the RGC Earmarked Grant.

Fig. 1 depicts a generic architecture of COPACC. A cluster of proxies are logically connected to form overlay. The proxies and their clients are closely located with relatively low communication costs, while the proxies and the video server are located far away and incur higher costs. The video data are cached across proxies and clients of limited storage. As shown in Fig. 2, a video stream is partitioned into *prefix* and *prefix-of-suffix*. The proxies are responsible to cache the prefix of video, whereas the clients cache the prefix-of-suffix. Similar to [3], this setting helps to reduce initial playback latency. When a client expects to play a video, it initiates a playback request to its home proxy, which intercepts the request and computes a streaming schedule. It then fetches the prefix, prefix-of-suffix, as well as the remaining part of suffix, and relays them to the client.

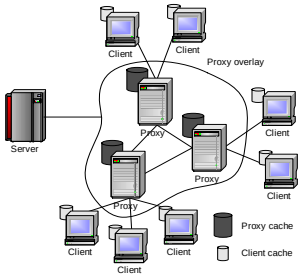

Fig. 1. The COPACC architecture.

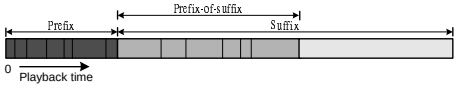

Fig. 2. Illustration of different portions of a video stream. The prefix is to be cached by proxies, while the prefix-of-suffix by clients

There are two key issues to be addressed: How to partition each video and allocate the prefixes and prefix-of-suffixes to different proxy and client? How to manage, search, and retrieve the cached data in different proxies and clients?

2 Optimal Cache Allocation Problem (CAP)

The optimal cache allocation problem (CAP) can be formulated as

$$\textbf{CAP}: \min\ Cost(\{p_j^i\},\{q_{j,k}^i\}),$$
$$\text{s.t. } p_j^i, q_{j,k}^i \geq 0, j \in [1 \dots H], k \in [1 \dots K_j];\quad \sum_{i=1}^{N} p_j^i \leq s_j^p;\quad \sum_{i=1}^{N} q_{j,k}^i \leq s_{j,k}^c;$$
$$\sum_{j=1}^{H} p_j^i + \sum_{j=1}^{H} \sum_{k=1}^{K_j} q_{j,k}^i \leq V^i,$$

where $Cost(\{p_j^i\},\{q_{j,k}^i\})$ is the total transmission cost given prefix allocation $\{p_j^i\}$ and prefix-of-suffix allocation $\{q_{j,k}^i\}$; the second and third constraints follow the cache space limit of proxy j and that of client k of proxy j, respectively; the forth constraint applies because we do not consider replication.

2.1 Single Proxy with Client Caching

A single proxy system is nice that the total transmission cost depends only on how the video streams are partitioned. We, therefore, combine the cache of all the clients to form an aggregated cache space, and derive the minimum transmission cost by finding the optimal values of $\{P^i\}$ and $\{Q^i\}$ subject to cache space constraints S^p and S^c. We define an auxiliary cost function $C^i(P^i, Q^i)$, which is the cost for delivering video i with prefix size P^i and prefix-of-suffix size Q^i. Note that $Cost(\{p_j^i\},\{q_{j,k}^i\})$ is now equal to $\sum_{i=1}^{N} C^i(P^i, Q^i)$. The problem can then be solved by *dynamic programming*. It is applicable with arbitrary cost function $C^i(P^i, Q^i)$, which can be instantiated given a specific transmission scheme. As an example, assume both a server-to-client and a client-to-client transmissions are unicast-based and relayed by a proxy, $C^i(P^i, Q^i)$ can be derived as $\lambda f^i \cdot [w^{c\leftrightarrow p} P^i + 2w^{c\leftrightarrow p} Q^i + (w^{s\rightarrow p} + w^{c\leftrightarrow p})(V^i - P^i - Q^i) + w^{in}(P^i + Q^i)]$, where the first four terms in the second part respectively represent the costs for retrieving prefix, prefix-of-suffix, the remaining suffix, and the internal cost of the proxy.

2.2 Multiple Proxies with Client Caching

A multiple proxies system is much more complex as it involve interactions among several proxies and clients, and the unit transmission costs for the proxy-to-proxy and client-to-proxy links can be heterogeneous. In fact, we formally prove that CAP is NP-hard in this general case (see [4]). We show this by transforming the optimal resource allocation problem (RAP), which is known as NP-hard [5], to CAP in polynomial time. We thus resort to a practically efficient heuristics, which consists of two phases: first, it partitions the prefix and prefix-of-suffix for each video; second, given the partitions, it allocates the segments of prefixes and prefix-of-suffixes to the proxies and clients.

1) Partitioning of prefix and prefix-of-suffix: In this phase, we approximate the system by a single proxy system with aggregated proxy cache space S^p and aggregated client cache space S^c. An approximate solution of $\{P^i\}$ and $\{Q^i\}$ can be directly obtained using the dynamic programming algorithm.

2) Allocation to proxy and client caches: In this phase, we further partition the prefix and prefix-of-suffix, and allocate them to the proxies and clients. Since the allocation for prefixes to proxy caches is independent from that for prefix-of-suffixes to client caches, we separate the two allocation problems and solve them individually. The optimal prefix allocation problem (**PA**) is formulated as

$$\begin{aligned} &\mathbf{PA}: \min \ \textstyle\sum_{i=1}^{N}\sum_{j=1}^{H} W^p(i,j,p_j^i) \\ &s.t. \ \ \textstyle\sum_{j=1}^{H} p_j^i = P^i,\ i \in [1 \ldots N]; \ \ \sum_{i=1}^{N} p_j^i \le s_j^p,\ j \in [1 \ldots H]. \end{aligned}$$

As $W^p(i,j,p_j^i)$ can be instantiated as $\sum_{j'=1}^{H} p_j^i [w_{j,j'}^{p \to p} + w_{j'}^{c \leftrightarrow p}] \lambda_{j'} f_{j'}^i$ for unicast delivery, the formulation of **PA** can be relaxed as a linear programming problem by re-writing it to $\min \sum_{i=1}^{N}\sum_{j=1}^{H} \bar{W}^p(i,j) \cdot p_j^i$.

We should also consider the optimal suffix-of-prefix allocation problem (**SA**) for client cache. Obviously, both the problem **SA** and the cost function itself have similar structure as that of problem **PA**. Thus, we omit the derivation of **SA** here. Notes that the linear programming relaxation also applies for **SA**. The optimizations shown above can also be applied to multicast delivery (see [4]).

3 Cooperative Proxy-Client Caching Protocol

As shown in Fig. 1, COPACC operates as a two-level overlay, where the first level consists of all the proxies, and the second level consists of each proxy and its own clients. The interactions among different entities in this two-level overlay are specified by a cooperative proxy-client caching protocol, which consists of three subprotocols. 1) *Cache allocation and organization* protocol specifies the election of proxy coordinator, which executes the optimal cache allocation algorithm and disseminates the lookup information using simplest hashing. 2) *Cache lookup and retrieval* protocol defines the discovery and retrieval of cache between proxies. 3) *Client access and integrity verification* protocol performs verification operation, which detects forged video data through a simple signature mechanism. The details of these subprotocols are not presented here.

4 Performance Evaluation

A primary design objective of COPACC is to reduce the transmission cost. Fig. 3 plots the transmission cost as a function of the total cache space, where the proxies and clients respectively contribute half of the total cache size. The cache sizes are normalized by the total size of the video repository, and the transmission costs are normalized by the corresponding cost of a system with no cache.

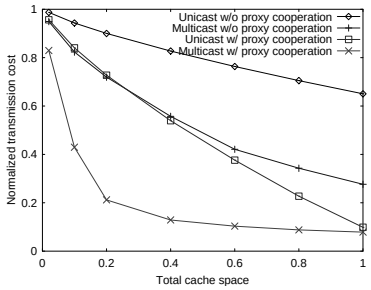

Fig. 3. Transmission cost as a function of the total proxy-client cache space.

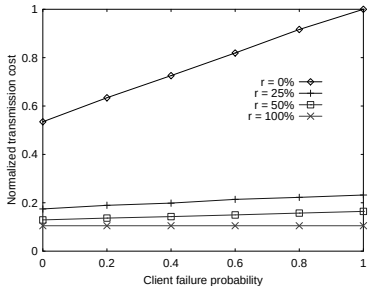

Fig. 4. Transmission cost versus client failure probability.

Not surprisingly, increasing the total space reduces transmission cost. With unicast, the cost decreases linearly, while with suffix multicast, it decreases much faster. When the total cache space is 0.2, the cost with suffix multicast has been reduced to 0.2; in other words, a 20% cache space leads to a 80% cost reduction, which implies that batching the requests from local clients can avoid a significant amount of remote transmissions. It is also clear that the cost with cooperative proxies is much lower, particularly when multicast is also enabled in local paths.

The robustness in the presence of client failures is also a critical concern in COPACC. In Fig. 4, we show the transmission cost as a function of different client failure probabilities. We vary, r, the fraction of the total proxy cache space in the total cache space from 0% to 100%. When $r = 0\%$, COPACC degenerates to a pure P2P system, and, when $r = 100\%$, it degenerates to a pure proxy-based system. We can see that, when there is no client failure, the costs for different r are quite close if there are certain cache existed in proxies. More importantly, the cost of the pure proxy-based system remains unchanged when increasing client failures, and that for $0\% < r < 100\%$ is also very stable. For illustration, even if r is 25%, the transmission cost only slightly increases with an increase of failure probability; when the failure probability is 1, the cost remains a low as 0.22. To the contrary, the cost of the pure P2P system quickly increases and reaches 1 (the cost of a zero-cache system), when all clients fail. Such results demonstrate that the use of dedicated proxies with suffix batching remarkably improves the robustness and resilience of COPACC in the presence of client failures.

References

1. Liu, J., Xu, J.: Proxy Caching for Media Streaming over the Internet. IEEE Communications (2004)
2. Cui, Y., Li, B., Nahrstedt, K.: oStream: Asynchronous Streaming Multicast in Application-Layer Overlay Networks. IEEE JSAC **22** (2004)
3. Wang, B., Sen, S., Adler, M., Towsley, D.: Optimal Proxy Cache Allocation for Efficient Streaming Media Distribution. In: Proc. IEEE INFOCOM'02, NY (2002)
4. Ip, A.T.S., Liu, J., Lui, J.C.S.: COPACC: A Cooperative Proxy-Client Caching System for On-Demand Media Streaming. Technical Report (2004, CUHK, http://www.cs.sfu.ca/∼jcliu/Papers/TR-COPACC-Final/TR-COPACC.pdf)
5. Katoh, N., Ibaraki, T., Mine, H.: Notes on the Problem of the Allocation of Resources to Activities in Discrete Quantities. Journal of Operational Research Society **31** (1980) 595–598