Minimizing Retrieval Cost of Multi-layer Content Distribution Systems

Mathias Björkqvist, Lydia Y. Chen
IBM Research Zurich Laboratory, Switzerland
Email: {mbj,yic}@zurich.ibm.com

Xi Zhang
Texas A&M University, USA
Email: xizhang@ece.tamu.edu

***Abstract*—Content distribution systems, such as on-demand video services [6], file-sharing networks [8], [1], and content clouds [2], provide ubiquitous data access and data sharing for large numbers of end-users. To efficiently provide content access across geographic locations, content storage nodes with limited capacity are conventionally organized in a multi-layer architecture to facilitate vertical as well as horizontal peer content retrievals, each of which may have different bandwidth constraints and transport costs. Content management, i.e., caching strategies, is deployed to efficiently utilize storage nodes and further reduce network retrieval traffic and cost. An optimal system design of such a multi-layer system needs to accommodate the trade-offs between vertical communication, peer communication, storage capacity and the users' retrieval traffic, driven by caching policies. In this paper, we propose a generic optimization framework based on steady-state content diffusion to minimize the total content retrieval cost in multi-layer content distribution systems, while considering the aforementioned trade-offs. The derived optimal content diffusion can evaluate the optimality of caching policies, and dimension the size of the system and the node caching capacity. Furthermore, we develop Peer Aware Content Caching (PACC) policies based on the derived optimal content diffusion. Our simulation results show that PACC effectively caches content and minimizes vertical and horizontal content retrieval costs under different system scenarios.**

***Index Terms*—content distribution, optimization, retrieval cost.**

I. Introduction

Modern content distribution systems are developed to provide access to various types of content, such as video clips and data files, to end-users who are geographically distantly located. The amount and the variety of the data stored in such systems is growing tremendously with the current trend of user-generated content and the expansion of on-demand services [5]. To ensure the availability and satisfactory retrieval of content items, a large amount of network storage infrastructure, together with content management policies, i.e., caching policies, are deployed. The emerging of today's cloud computing [2] is partially to address scalability and cost-efficiency issues related to content access for enterprises and individual users. The challenge for providers of content distribution systems is to retain a good system design, i.e., the structure of storage nodes and storage capacity, with minimal content retrieval cost and content management overhead.

A multi-layer tree-like architecture is conventionally adopted in content distribution systems. End-users request content items, which are retrieved top-down from higher layers to the lower layers, depending on the content diffusion - that is, the availability of content items in storage nodes [7]. Caching strategies have been shown to effectively organize a huge variety of content items with widely differing popularity in limited storage space [7], [14]. Recently, peer-assisted architectures [11], [5] have enabled the cached content items to be shared among some peer nodes at the same layer. The potential benefits of peer content retrieval stem from the shorter geographic distance and lower infrastructure cost [9], which often result in lower horizontal peer retrieval costs than the vertical retrieval. As a result, various content caching policies, such as collaborative caching [5], [15], [18], are designed to take advantage of peer and vertical content retrieval in best utilizing the limited storage space.

Content popularity is commonly adopted as a criterion in caching design, such as Least Recently Used (LRU) popularity. In many cases, end-users' requests for content items have been observed to follow a Zipf-like popularity distribution [13]. In the aforementioned multi-layer content distribution system, the actual distribution of retrieval requests received at a certain layer of storage nodes depend on the content diffusion at peer nodes and nodes at lower layers. Therefore, the content diffusion and popularity at each layer is affected by complex cross-layer and inter-layer interactions, which are due to the caching policies and retrieval channels used. To minimize the total retrieval traffic of the entire multi-layer system, it is critical to characterize the content popularity and diffusion in a layer-wise fashion with respect to the system architecture and caching policies.

Depending on the nature of the content items and the system architecture, various performance measures are used as optimization criterions in the caching policies. Minimizing the retrieval latency is the main objective in web-based content systems [8], [12], whereas bandwidth consumption is one of the foremost performance criterions in content streaming systems [16], [5]. As the variety in terms of types of content and types of hosting platforms increases, unified performance measures, such as retrieval cost, that integrate bandwidth consumption and retrieval latency as well as infrastructure expenses, are very desirable when evaluating future systems. Meanwhile, as the overall system performance of a multi-layer content distribution system depends on several interrelated issues, such as the retrieval channels, the system size, the cache

size of the storage nodes, caching policies etc., a systematic approach is required to combine all the aforementioned dimensions when optimizing multi-layer systems.

In this paper, we build a simple and generic content diffusion optimization framework for a three-layer content distribution system, where both vertical and horizontal retrieval is enabled. Herein, the first two layer consists of storage nodes with limited buffer space, and the third layer is the central server. The objective is to minimize the total content retrieval cost by controlling the content diffusion, subject to the entire system buffer space. We further numerically evaluate the dimensioning of the network and buffer size using the proposed optimization framework. Secondly, we propose two Peer Aware Content Caching (PACC) policies, PACC-AR and PACC-CL, which are based on the derived optimal content diffusion and which employ different degrees of peer collaboration. Specifically, PACC-AR is implemented distributively, whereas PACC-CL collaborates with peer nodes in the caching process. The effectiveness of the PACC policies are shown in simulations with various system specifications.

The remainder of this paper is organized as follows: The system specification is given in Section II. The proposed content diffusion optimization is described in Section III. The proposed PACC polices are explained in Section IV. Section V contains the experimental results. The related work is presented in Section VI, and Section VII concludes the work.

II. System and Retrieval Cost

The distribution system considered maintains a total of K content items of constant size in a three-layer hierarchy of content distribution system as shown in Figure 1. Each node at layer $j \in \{1, 2\}$ is assumed identical and has a buffer capacity of B_j, which is assumed far smaller than K, $B_j \ll K$. The central server is the top layer, which has no buffer constraint and contains all content items. Every node at layer j is vertically connected to a parent node at layer $j+1$ and to N_{j-1} child nodes at layer $j-1$. Moreover, N_j peer nodes at layer j having the same parent node are horizontally interconnected and form a so-called peer network at layer j. The request rate of a content item k, $r_k = \frac{1}{k^\alpha}, k = \{1 \dots K\}$, from end-users is assumed to follow a Zipf distribution and is estimated through off-line profiling and online estimation. Content requests are assumed uniformly distributed across all nodes. For a given stationary window, $r_k, \forall k$ is constant and we thus focus on the steady state of the content diffusion. Due to storage capacity limitations and depending on the caching policy, it may be the case that a requested content item is not cached at the local node. We refer to the average availability of a content item at storage nodes at layer j as content diffusion, denoted by π_k^j.

At any given layer, an uncached content item can either be retrieved from the peer nodes, as long as at least one copy is available in the peer network, or vertically from the parent node. Both channels are assigned retrieval costs, which are meant to capture the costs related to the network bandwidth and geographic retrieval distances [6]. We let the vertical content retrieval cost at layer j to be c_j and the peer retrieval cost at layer j to be c'_j. Note that the specific values of c_j and c'_j are system dependent, and thus considered as given input parameters of the optimization in this study. If a content item is retrieved from the local node, there is no retrieval cost associated with it. In general, we assume the horizontal retrieval cost to be less than the vertical retrieval cost, meaning $c'_j \leq c_j$.

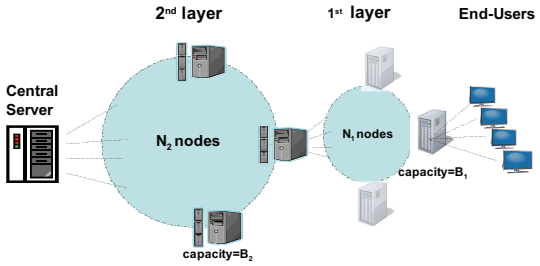


Fig. 1. A schematics of a three-layer content distribution system.

III. Content Diffusion Optimization

We propose a content diffusion optimization framework that minimizes the content retrieval costs based on the content diffusion, which is constrained by the buffer capacity. The derivation presented here is for a three-layer system, but the derivation can be extended for an arbitrary number of layers. We also discuss how the proposed optimization framework can be used to explore the dimensioning of peer network size and buffer size.

Let us first derive the layer-wise retrieval traffic. Let the probability that a copy of content item k is locally cached at a layer j node be π_k^j, which ranges between $[0, 1]$. Due to the assumption of symmetric nodes, $\pi_k^j \, \forall k$ is identical for all nodes at layer j. The retrieval traffic of content item k from a single node at layer 1 is the product of the content request rate and content unavailability, $r_k(1 - \pi_k^1)$. Note that we normalize r_k throughout this study, so that $\sum_k r_k = 1$. The content retrieval requests can be fulfilled by peers with the probability $1 - (1 - \pi_k^1)^{N_1 - 1}$, meaning that at least one of $N_1 - 1$ peer nodes has such a content item. On the other hand, an uncached content item k is retrieved vertically from the parent node with the probability $(1 - \pi_k^1)^{N_1 - 1}$. The total retrieval traffic at a first-layer node is thus $\sum_k r_k\{(1-\pi_k^1)(1-(1-\pi_k^1)^{N_1-1})+(1-\pi_k^1)^{N_1}\}$. The retrieval cost incurred at the total of $N_1 N_2$ first-layer nodes is the product of the content retrieval traffic and the corresponding retrieval costs, c_1 and c'_1,

$$N_1 N_2 \sum_k r_k \{c'_1(1 - \pi_k^1) + (c_1 - c'_1)(1 - \pi_k^1)^{N_1}\}.$$

The request for content item k is received by a second-layer node at the rate $N_1 r_k (1 - \pi_k^1)^{N_1}$, which is essentially the vertical retrieval traffic for content item k generated by N_1 nodes at the first layer. The retrieval traffic of content item

k generated at the second layer is the product of the local diffusion and the request rate received from the first layer, $N_1 r_k(1-\pi_k^1)^{N_1}(1-\pi_k^2)$. Some items can be horizontally retrieved from the peer nodes with probability $1-(1-\pi_k^2)^{N_2-1}$, whereas others can only be retrieved vertically from the central server with probability $(1-\pi_k^2)^{N_2-1}$. The retrieval cost for content item k at each second-layer node is

$$N_1 r_k(1-\pi_k^1)^{N_1}(1-\pi_k^2)\{c_2'(1-(1-\pi_k^2)^{N_2-1})+c_2(1-\pi_k^2)^{N_2-1}\},$$

where c_2 and c_2' are the horizontal peer retrieval and vertical central server retrieval cost, respectively.

The total content retrieval cost, RC, of entire system is

$$\begin{aligned} RC = \quad & N_1 N_2 \textstyle\sum_k r_k\{c_1'(1-\pi_k^1) + (c_1 - c_1')(1-\pi_k^1)^{N_1} + \\ & (1-\pi_k)^{N_1}\{c_2'(1-\pi_k^2) + (c_2 - c_2')(1-\pi_k^2)^{N_2}\}\}. \end{aligned}$$

As each node at layer j has a capacity to cache B_j content items, the sum of the entire content diffusion (availability) is $\sum_k \pi_k^j = B_j$ in the long run. We summarize the content diffusion optimization, denoted by **O**, as

$$\begin{aligned} (\mathbf{O}) \quad & \min RC(\pi_k^1 \ldots \pi_K^1, \pi_1^2 \ldots \pi_K^2) \\ & \textstyle\sum_k \pi_k^1 = B_1 \\ & \textstyle\sum_k \pi_k^2 = B_2 \\ & 0 \le \pi_k^1, \pi_k^2 \le 1. \end{aligned}$$

To solve a general **O**, we suggest to resort to CONOPT [3] due to the intractability of obtaining a closed-form solution from high-order functions. As for the **O** with degenerate cost structure, i.e., $c_2 = c_2'$, we can obtain a simple optimal structure for content diffusion.

Lemma 3.1: When $c_2 = c_2'$, the optimal policy is to statically cache the B_2 of content which receives the highest content request rate $r_k(1-\pi_k^1*)^N$.

Proof: When $c_2 = c_2'$ and after some algebraic manipulation, one can get $RC = N_1 N_2 \sum_k r_k\{c_1'(1-\pi_k^1) + (c_1 - c_1')(1-\pi_k^1)^{N_1})\} + c_2 r_k(1-\pi_k^1)^{N_1}(1-\pi_k^2)$. Thus, RC is clearly a linear function of π_k^2 with coefficients $c_2 r_k(1-\pi_k^1)^{N_1}$. For given r_k and π_k^1, to minimize RC, one needs to have $\pi_k^2 = 1$ for B_2 content items, which have the highest coefficient, $r_k(1-\pi_k^1)^{N_1}$. It also holds true for π_k^1*, which in turn minimizes RC. Thus, statically caching B_2 content items with the highest $r_k(1-\pi_k^1*)^N$ minimizes the retrieval cost. ■

A. Sensitivity Analysis: Buffer and Peer

In principle, the proposed diffusion optimization can be used for evaluating system dimensions, such as optimal size of peer network and caching capacity, by assuming optimal diffusion (π_k^j*) can be obtained by some caching policies. For instance, with a fixed total caching capacity at layer j, Γ_j, one can find a peer network size and the buffer size that can best utilize the horizontal and vertical communications by modifying the diffusion constraints in **O** as

$$\sum_k \pi_k^j = \frac{\Gamma_j}{N_j}.$$

A numerical derivative of RC can be taken with respect to N_j to evaluate whether the current system dimensions of layer j can be further improved.

An important premises of applying the proposed diffusion optimization numerically in actual system design is to obtain accurate estimations for system-dependent retrieval costs, which is out of scope of the paper. In a larger peer network, the content can be retrieved from a closer neighboring node with a higher probability and possibly higher bandwidth [6]. On the other hand, a smaller peer network with bigger individual buffers might be more affected by bandwidth limits, because of higher traffic loads. We present the numerical results in Section V to illustrate how the retrieval cost affects optimal sizing.

IV. Peer-Aware Content Caching

The objective of the content management policy here is to attain the minimal retrieval cost in O, which is equivalent to achieving the optimal content diffusion π_k^j* defined in the proposed diffusion optimization. As nodes are assumed to be uniformly identical, we can straightforwardly derive the optimal number of content item k at the layer j peer network as

$$M_k^j* = N_j \pi_k^j * .$$

As a result, keeping the number of content item copies at M_k^j, $\forall k, j$ for the entire system can also minimize the system-wide content retrieval cost. The centralized management of content copies over the entire network is not trivial and can cause significant overhead [4]. Moreover, it is challenging to fairly distribute M_k^j* copies among N_j nodes so that the loads of the individual network nodes are balanced. An unfair and unbalanced distribution of M_k^j* copies could discourage peers from sharing content and further break down the system [10]. To achieve an optimal M_k^j* with a minimum overhead of copy control and peer collaboration, we propose the Peer-Aware Content Caching (PACC) policy, which can be implemented in a distributed or a semi-distributed fashion.

When receiving a content request, the PACC policy first decides whether or not to cache the content item for future use. If a new item is decided to be cached and the buffer is full, an existing content item is discarded. The optimal content diffusion π_k^j*, the peer content distribution, and the local content popularity are criteria considered in PACC. In general, PACC keeps a stand-alone copy of a content item k at a layer j node if the derived $\pi_k^j* \ge T^{j-max}$, where T^{j-max} is a threshold value. For example, when $T^{j-max} = 1$, PACC keeps a stand-alone copy of content item k, which has $\pi_k^j* = 1$. PACC does not keep content item k at the layer j peer network if its optimal diffusion is less than the threshold, $\pi_k^j* \le T^{j-min}$. For instance, when $T^{j-min} = 0.3$, PACC does not keep content items which have $\pi_k^j* = 0.3$.

For content items with an optimal diffusion between $[T^{j-min}, T^{j-max}]$, two variations of PACC, PACC-AR and PACC-CL, employ different degrees of selfishness and collaboration for caching and discarding. The intuitive values

for $[T^{j-min}, T^{j-max}]$ are [0,1], which are adopted in our experimental evaluations. Nevertheless, T^{j-min} and T^{j-max} are control parameters that can be tuned according to system specifications and parameters. The definitions of AR and CL are as follows:

- AR (**A**lways cache, **R**andomly discard)
  The PACC-AR algorithm always caches requested content and discards local content randomly. PACC-AR is a purely distributed policy without explicit peer collaboration. Note that as the threshold values for applying AR are determined by the proposed content diffusion optimization, PACC-AR is definitely peer-aware. Always-caching increases the probability of keeping more popular content at the local node, whereas random discarding ensures that every node discards some popular content and thus keeps some unpopular content. Consequently, all content items can be circulated among all peers, even though every node manages its own buffer selfishly according to its local requests.
- CL (**C**ollaboratively cache, **L**RU discard)
  The PACC-CL algorithm explicitly collaborates with peers to keep M_k^{j-max} copies of content item k, and discards local content following a Least Recently Used (LRU) policy. It only maintains M_k^{j-max} copies of content item k ($T^{j-min} < \pi_k < T^{j-max}$) across the peer network, where M_k^{j-max} is the ceiling of average allocatable free buffer after excluding stand-alone and never-cached content items. For example, the system has $N_j = 20$ and $B_j = 10$ to cater $K = 200$ content items. According to π_k^j*, PACC decides to keep content items with id $k = 1-2$ and never cache content items with id $k = 101-200$. We then use $M_k^{j-max} = \lceil \frac{(10-2)\cdot 20}{100} \rceil = 2$ to make caching decision for content items with id $k = 3-100$. An existing cached content item is discarded if it is the least recently requested item in the local buffer. Collaborative caching can ensure that every node caches less popular content, whereas LRU discarding increases the probability of keeping more popular content locally. Thus, all content copies can be distributed evenly among all peers.

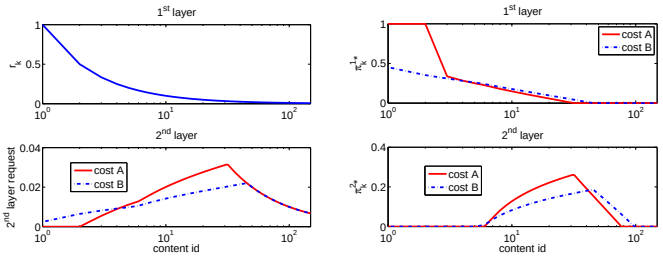
Algorithm 1 PACC-AR and PACC-CL

if $\pi_k^j* > T^{j-max}$ **then**
  keep content item k locally
else
  if $T^{j-min} < \pi_k^j* < T^{j-max}$ **then**
    AR: always cache, randomly discard
    CL: collaboratively cache, LRU discard
  end if
else
  if $\pi_k^j* < T^{j-min}$ **then**
    never cache.
  end if
end if

V. Numerical and Simulation Results

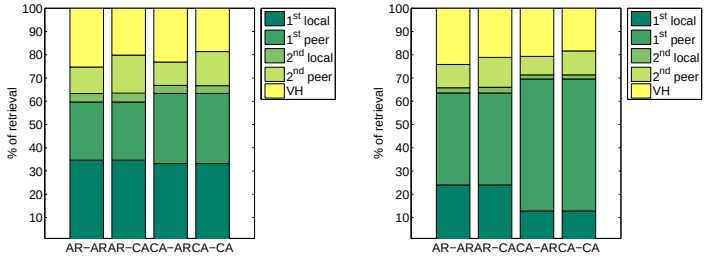
We built an event-driven simulator in the Java environment to validate the proposed content diffusion optimization and the proposed PACC policies, PACC-AR and PACC-CL. In the simulator, incoming requests are processed atomically so that individual requests do not interfere with one another. First we compare different combinations of the PACC policies in a three-layer system with respect to different cost functions. We then present the numerical results for the first layer with varying buffer size (B) and number of peer nodes (N).

A. Retrieval Traffic Costs



(a) Content request rates. (b) Optimal content diffusion.

Fig. 2. A three-layer system under different cost function A and B.



(a) Cost A (b) Cost B

Fig. 3. The distribution of content retrieval traffic under different cost structures.

The system considered here has a central server and two layers of network storage, with $N_1 = 10$, $B_1 = 5$, $N_2 = 5$, $B_2 = 10$, to provide $K = 150$ content items. The content request rate of the end-users is r_k, and the Zipf distribution has $\alpha = 1$. The two sets of retrieval costs considered are (a) lower and (b) higher peer retrieval cost, which are summarized in Table I. Figures 2 (a) and (b) show the request rates and optimal content diffusion derived from (**O**) with respect to each layer. The optimal diffusion in both layers has a similar trend, which is driven by the number of content requests received. As we can see that the requests for content items received in the first layer have a higher variance than in the second layer, the optimal diffusion π_k^1* has a larger variance than π_k^2* in general. The other observation is that the optimal content distribution here is to have content items with smaller k in the first layer and content items k with medium popularity in the second layer. The remainder of the content items are retrieved from the central server.

TABLE I
COST FUNCTIONS

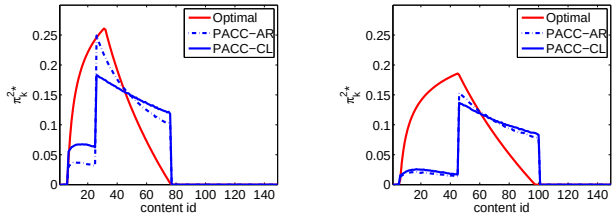
	c_1'	c_1	c_2'	c_2
Cost A	1	2	2.5	3
Cost B	0.25	1.25	1.5	2.5

We apply both PACC-CL and PACC-AR in different combinations on the first and second layers. For each cost function we thus have four combinations on the first and second layer, namely AR-AR, AR-CL, CL-AR and CL-CL, with the following parameters:

- Cost A
  At the first layer, PACC keeps content items with id $k = 1-2$ statically, and never caches a content item with id $k > 25$. For content items with id $k = 3-25$, PACC-AR and PACC-CL policies are applied. Specifically, PACC-CL here keeps at most two copies ($M_k^{1-max} = 2$) of such content items in the peer network. At the second layer, PACC-CL and PACC-AR collaboratively cache content items with id $k = 7-76$, and PACC-CL tries to keep at most two copies ($M_k^{2-max} = 2$) of them. The remainder of the content items are not cached.
- Cost B
  For cost B, PACC keeps no static copies for either layer. At the first layer, both PACC policies collaborate for content items with id $k = 1-45$, and PACC-CL keeps at most one copy ($M_k^{1-max} = 1$) of such content items in the peer network. At the second layer, PACC-AR and PACC-CL collaboratively cache content items with id $k = 6-100$, and PACC-CL again keeps at most one copy ($M_k^{2-max} = 1$) at the second-layer peer network. The remainder of the content items are not cached.

The resulting retrieval traffic distribution is shown in Figure 3, where 1^{st} local and 2^{nd} local denotes that content items are retrieved locally from the first and the second layer nodes, respectively. PACC-CL is observed to have a lower local retrieval rate, but higher peer traffic. Applying CL-CL in both cost functions achieves the lowest central server traffic, especially in the case of a lower peer retrieval cost. We can also see that both PACC caching policies retain higher percentages of local retrieval and peer retrieval in the first-layer nodes than in the second-layer nodes. This can be explained by the content diffusion distribution and requests, shown in Figure 2, which are step-like functions. The parameters of the PACC-CL and PACC-AR policies at the second layer are currently based on the predicted second-layer requests rates, shown in Figure 2 (a). They are essentially a function of the predicted optimal diffusion of the first layer. We believe that as the accuracy of the predicted second-layer request rate improves, PACC can potentially achieve lower central server retrieval traffic in the second layer.

The total retrieval cost normalized by the predicted optimal cost is summarized in Table II. The lowest content retrieval cost is obtained for both cost functions when the PACC-CL caching policy is used on both layers. On other hand, PACC-AR caching on both layers has a higher retrieval cost, but without peer collaboration overhead. The overhead of applying PACC-CL is lower when the number of peers is smaller. The trade-off between PACC-CL and PACC-AR is in terms of saving of retrieval cost and the collaboration overhead. For Cost A, AR-CL has a decent retrieval cost but only needs to maintain collaborative caching with four other peer nodes, i.e., far less than collaborating at the first layer. For Cost B, CL-AR has a more prominent retrieval cost saving than AR-CL, and lower caching collaboration overhead than CL-CL. Depending on the system specification, PACC-CL and PACC-AR can be used interchangeably to achieve the desired trade-off. Overall, both PACC-CL and PACC-AR are very effective in achieving the predicted optimal retrieval cost.



(a) Cost A (b) Cost B.

Fig. 4. Second-layer content diffusion for a three-layer system, in which PACC-CL is applied at the first-layer nodes.

TABLE II
RETRIEVAL COST NORMALIZED BY THE PREDICTED OPTIMAL COST

(1^{st} layer -2^{nd} layer)	Cost A	Cost B
AR-AR	106.26%	99.70%
AR-CL	104.21%	96.20%
CL-AR	103.35%	90.56%
CL-CL	100.64%	87.93%

B. Impact of Buffer and Peer Network Sizes

In this subsection, we present an example of numerical evaluations for different sizes of peer nodes and buffers at the first layer. The total available storage capacity is $\Gamma = 100$ and the individual buffer size of a node is $B_1 = \frac{100}{N_1}$. We compare three sets of $(B_1, N_1) = \{(5, 20), (10, 10), \text{and } (50, 2)\}$. We set the peer retrieval cost $c_1' = 1$, and vertical retrieval cost $c_1 = 1 + \beta\frac{1}{\sqrt{N_1}}$. The difference between c_1 and c_1' is an increasing function of N_1, meaning that the vertical retrieval cost is higher in a larger system than in a small system. We use three different values of $\beta = \{2, 6, 10\}$. The optimal total retrieval cost computed in the content diffusion optimization is illustrated in Figure 5.

When $\beta = 2$, a smaller number of peer nodes with bigger buffers can have the lowest retrieval cost because of the higher difference in vertical and horizontal retrieval costs. On the contrary, when $\beta = 10$, a larger peer network with small individual buffers can achieve the lowest retrieval cost. As for $\beta = 6$, the total retrieval cost differences among different dimensions of network and buffer sizes are insignificant. A general observation from our numerical experiments is that

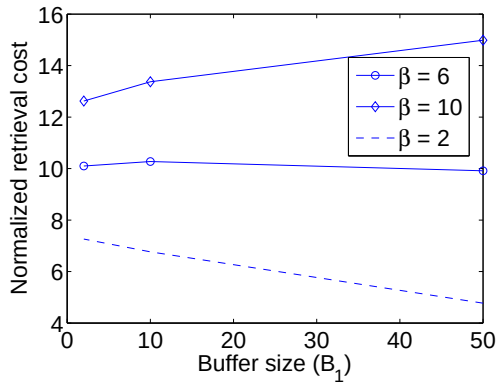


Fig. 5. The optimal content retrieval cost when retrieval cost difference is a decreasing function of an increasing N_1, $c_1 - c_1' = \beta\frac{1}{\sqrt{N_1}}$.

the optimal design is highly impacted by the cost function and content request rates, which are very system specific. As demonstrated by our results, we believe the proposed simple content optimization framework can efficiently provide performance evaluations to explore the design space of multi-layer content distribution systems.

VI. Related studies

Prior studies on caching strategies mainly focus on optimizing the performance of specific content distribution configurations, e.g., web proxy caching [18], [17], IPTV video caching [14], [7] and distributed file systems [8]. The performance characteristics of interest are mainly access latency and bandwidth consumption, depending on specific system and application scenarios. Few studies have investigated cost optimization using collaborative caching. Cha et al. [6] empirically compared the cost of different IPTV architectures to distribute video content. Borst et al. [5] formulated a content retrieval cost optimization using integer programming, which is not applicable when evaluating the performance impact from a system perspective, i.e., the dimensions of system size and storage capacity. Our proposed optimization framework is capable of evaluating the overall content retrieval cost of a multi-layer system, as well as providing a sensitivity analysis of the system parameters. From the perspective of a single layer, the proposed PACC policies show strong resemblances to caching policies proposed in earlier studies [5], [15]. As PACC is based on a multi-layer optimization, it is not only collaborating with peer nodes, but also higher-layer and lower-layer nodes, which are not explicitly considered by earlier proposed collaborative caching policies.

VII. Conclusion Remark

In this study, we first proposed a content diffusion optimization framework to minimize the content retrieval cost in a three-layer content distribution system. The framework captures the critical system parameters, including the content request rate, the layer-wise storage capacity, the layer-wise peer network size and the content retrieval cost. The derived content diffusion is used to investigate the trade-off between vertical and peer retrieval costs. Secondly, we developed two peer-aware caching policies, PACC-AR and PACC-CL, whose parameters are based on the derived optimal content diffusion. PACC-AR is purely distributed, whereas PACC-CL requires peer collaboration in the caching process. Our experiments show that both come close to achieving the optimal diffusion and, moreover, also achieve fairly low retrieval costs, especially PACC-CL. As PACC has been developed on the proposed optimization framework, it can easily be adjusted for various system specifications and cost functions. In general, we observed that the optimal content diffusion and the optimal system dimensioning highly depend on the retrieval cost structure. We plan to conduct experiential studies to quantify retrieval cost functions in the future and validate our results on real networks.

References

[1] http://wua.la.

[2] http://www.cloudbook.net/reservoir-gov.

[3] http://www.conopt.com.

[4] V. Aggarwal, A. Feldmann, and C. Scheideler. Can ISPs and P2P Users Cooperate for Improved Performance? *SIGCOMM Comput. Commun. Rev.*, 37(3):29–40, 2007.

[5] S.C. Borst, V. Gupta, and A. Walid. Distributed Caching Algorithms for Content Distribution Networks. In *Proceedings of IEEE INFOCOM*, 2010.

[6] M. Cha, P. Rodriguez, S. Moon, and J. Crowcroft. On Next-Generation Telco-Managed P2P TV Architectures. In *Proceedings 7th International Workshop on Peer-to-Peer Systems (IPTPS)*, 2008.

[7] Y. Chen, M. Meo, and A. Scicchitano. Caching Video Content in IPTV systems with Hierarchical Architecture. In *Proceedings of IEEE International Conference on Communications (ICC)*, 2009.

[8] M. Dahlin, O. Y. Wang, T. Anderson, R. Wang, and D. Patterson. Cooperative Caching: Using Remote Client Memory to Improve File System Performance. In *Proceedings of USENIX Conf. on Operstaiong Syst. Design and Implimentation (OSDI)*, 1994.

[9] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross. Measurement Study of a Large-Scale P2P IPTV System. *IEEE Transactions on Multimedia*, 9(8):1672–1687, 2007.

[10] A. Karthik, A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica. Load Balancing in Dynamic Structured P2P Systems. In *Proceedings of IEEE INFOCOM*, 2005.

[11] S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, and M. Chiang. Performance Bounds for Peer-Assisted Live Streaming. *SIGMETRICS Perform. Eval. Rev.*, 36(1):313–324, 2008.

[12] J. Ni and D.H.K Tsang. Large-scale Cooperative Caching and Application-level Multicast in Multimedia Content Delivery Networks . *IEEE Communication Magazine*, 43(5):98–105, 2005.

[13] T. Qiu, Z. Ge, S. Lee, J. Wang, Q. Zhao, and J. Xu. Modeling Channel Popularity Dynamics in a Large IPTV System. In *Proceedings of ACM SIGMETRICS*, pages 275–286, 2009.

[14] L. Sofman, B. Krogfoss, and A. Agrawal. Optimal Cache Partitioning in IPTV Network. In *Proceedings of the 11th Communications and Networking Simulation symposium*, pages 79–84, 2008.

[15] S. Tewari and L. Kleinrock. On Fairness, Optimal Download Performance and Proportional Replication in Peer-to-Peer Networks. In *Proceedings of IFIP Networking*, pages 709–717, 2005.

[16] C. Wu, B. Li, and S. Zhao. Multi-channel Live P2P Streaming: Refocusing on Servers. In *Proceedings of IEEE INFOCOM*, 2008.

[17] J. Yang, W. Wang, and R. Muntz. Collaborative Web Caching Based on Proxy Affinities. In *Proceedings of ACM Sigmetrics*, pages 78–89, 2000.

[18] P. Yu and E. MacNair. Performance Study of a Collaborative Method for Hierarchical Caching in Proxy Servers. *Comput. Netw. ISDN Syst.*, 30(1-7):215–224, 1998.