Modeling Hierarchical Caches in Content-Centric Networks

Zixiao Jia[†§], Peng Zhang[†§], Jiwei Huang[†§], Chuang Lin[†§], and John C. S. Lui[‡]
[†]Tsinghua National Laboratory for Information Science and Technology
[§]Dept. of Computer Science and Technology, Tsinghua University, Beijing, China
[‡]Dept. of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong
{jiazixiao, pzhang, huangjiwei, clin}@csnet1.cs.tsinghua.edu.cn, cslui@cse.cuhk.edu.hk

***Abstract*—Content-Centric Network (CCN) provides a clean-slate design for the Internet, where content becomes the primitive of communications. In CCN, routers are equipped with *content stores*, which act as caches for frequently requested content. This design enables the Internet to provide content distribution services without any application-layer support. On the other hand, as caches are integrated into routers, the overall performance of CCN will be influenced by the caching efficiency.**

This paper studies the performance issues of caches in CCN, with the aim to gain some understanding on how caches should be designed to maintain a high performance in a cost-efficient way. Specifically, we use a two-dimensional discrete-time Markov chain to model the two-layer cache hierarchy formed by CCN routers, and develop an efficient algorithm to calculate the hit ratios of these caches. Simulations validate the accuracy of our modeling method, and convey some understanding on cache design in CCN.

I. Introduction

Content-Centric Network [1] (CCN) is a new Internet architecture in which content is treated as the primitive of communication. In CCN, each piece of content contains a name that uniquely identifies it, and is transmitted in a receiver-driven way. When the requested content is returned from the source, it is remembered/cached by intermediate routers (termed as *content routers*). As a result, these content routers can serve later requests for the same content, without resorting to the source again.

Owing to its built-in caching capability, CCN enables the Internet to support content distribution services directly in its network layer, without any application-layer solutions, e.g., CDN. This feature makes CCN a good alternative for Internet Service Providers (ISPs) to offload their IP backbone traffics, which are currently overwhelmed by Over-The-Top (OTT) content like Internet videos. Fig. 1 shows a possible deployment of CCN in an ISP's network, where content routers are organized in layers, with the lowest layer accepting requests from customers and the topmost layer connecting to the IP backbone.

The deployment of CCN in ISPs' networks, however, is not that straightforward. As caches will be integrated into routers, the network performance will be influenced by the caching efficiency. For instance, if the storage volumes of these caches are too small, the benefits of employing CCN can even be offset. On the other hand, caches stand as a major infrastructure investment for ISPs, due to their high demand on both data access speed and storage volume. Thus, *how these caches should be designed cost-efficiently, while still maintaining a high performance is a problem to be examined*.

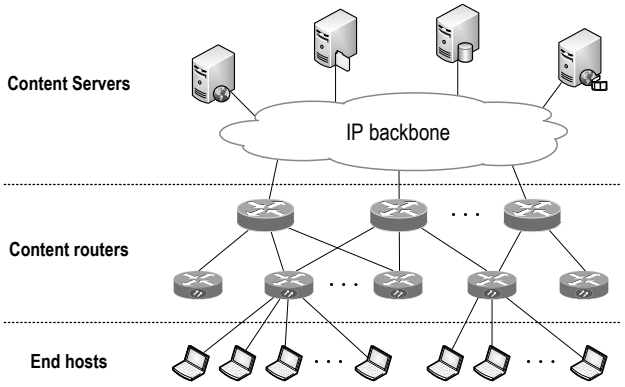


Fig. 1. The hierarchical structure of content routers in CCN.

The caching problem has been extensively studied in the area of web caching. However, caches in CCNs have some differences from web caches in the following three aspects. First, web caches are application-specific (for web browsing), while CCN caches are for general content distribution services. Second, web caches use single-path routing, while CCN allows content to be retrieved and distributed along multiple paths. Third, web caching is based on objects/files, while caching in CCN is based on chunks/packets.

There have been some efforts on studying the caching performance in CCN. However, most of them are based on trace-driven simulations or experiments. Existing modeling methods for CCN caches are either complicated to be solved or fall short in giving valuable guidance on how to design the caches.

In this paper, we treat the layers of content routers in CCN as a cache hierarchy (Fig.1), and try to develop models which can guide the design of caches at different layers. Specifically, we use discrete-time Markov chains to capture the dynamics of content occupancy in hierarchical caches, which can be used to efficiently calculate the cache hit ratios. In addition, the variables used in our model convey some meaningful information which can help us better understand the caching mechanism of CCN. The main contributions of this paper are as follows:

1) We propose probability models for two-layer cache hierarchy. These models are based on two-dimensional Markov chains, each of which contains a relatively large number of states. To obtain analytical solutions, we introduce a key variable R to decompose the two-dimensional Markov chain into a series of one-dimensional Markov chains, which can be iteratively solved.

2) Based on our models, we develop an efficient algorithm to calculate the hit ratio for each cache. Simulations show that the results are accurate with error less than 5%.

3) By analyzing the numerical results obtained from our models, we gain some understanding on (a) the impact of cache size on hit ratio, (b) "filter effect" imposed by lower-layer caches, (c) size requirements on different layers of caches, etc.

The rest of this paper is organized as follows. In Section II, we briefly introduce the background of CCN and make some assumptions. Section III presents two models: the leaf node model and the layer-1 node model. Section IV validates our models in terms of simulations, and reports some numerical results obtained from our models. Section V surveys some related work and Section VI concludes.

II. Background and Assumptions

A. Background

In CCN, there are two types of packets: Interest and Data. A user who is interested in a data packet broadcasts an interest packet over its connections. Any router (termed as *content router*) which receives the interest packet checks its *content store*, which is a cache of data packets. If the requested data packet is already cached in the content store, the data packet would be returned directly by that router. On the other hand, if the data packet is not found, the content router will look up in its *Forwarding Information Base* (FIB), and forward the interest packet to a list of outgoing interfaces. The router also keeps track of the pending request by inserting a record to its Pending Interest Table (PIT). When the corresponding data packet is returned, the router checks its PIT to decide which interfaces to forward this data packet. After forwarding, the router also keeps a replica of the data packet in its content store. Then, if interest packets for the same content arrive again, the router can directly return the content.

As seen above, data requests are not constrained to flow along a single path as in IP networks, but can be routed in a multi-path fashion. In addition, interest packets for the same content will be aggregated as a single PIT entry containing a list of interfaces. When the content is returned, it will be sent over all these interfaces. This enables CCN to inherently support multicast transmission.

In CCN, all contents are first splitted into packet-sized chunks (4KB as specified in current CCNx implementation [2]). To request a file, users send out a sequence of interest packets, one for each chunk. Note that these interest packets can be pipelined to reduce the response time.

TABLE I
Summary of key notations.

I_A	Indictor function which takes 1 if predicate A is true, and 0 otherwise
C	Set of all chunks in the system
K	Number of popularity ranks
C_k^i	Set of rank-k chunks requested by end hosts connected to v_i
$\alpha_k^i(\beta_k^i)$	Request ratio of rank-k chunks (a specific chunk) at node v_i
g_k^i	Size of set C_k^i
S_i	Cache size of node v_i
$p_k^i(j)$	Probability that any rank-k chunk is stored at the j^{th} slot of v_i
$b_k^i(j)$	Expected number of rank-k chunks stored in the first j slots of v_i
$h^i(h_k^i)$	Cache hit ratio of node v_i (for a specific rank k)
$\sigma_{m,n}$	Probability that node v_m forwards missed chunk requests to node v_n
δ_i	Probability that when a request comes to the system, it arrives at v_i
$R_k^{m,n}(j)$	Probability that a chunk c of rank k is not present in the cache of v_n condition that it is present in the j^{th} slot of v_m
$D_k^{m,n}(j)$	Expected number of rank-k chunks in the first j positions of v_m but not existing in node v_n

B. Assumptions

In our model, the system consists of three components: content servers, content routers and end hosts. Content routers are equipped with content stores (caches), and are organized in layers, as shown in Fig.1. We term routers of lowest layer as *leaf routers* or *leaf nodes*[1]. Leaf routers can accept data requests from end hosts that are directly connected to them. If data requests cannot be satisfied locally, they would be forwarded to the upper-layer routers, which we term as *layer-1 routers*.

Content Popularity. Let C be the collection of all chunks in the system, and these chunks are classified into K ranks of popularity. Chunks in lower ranks will be requested by end hosts with higher probabilities. For each leaf node v_i, let C_k^i be the set of rank-k chunks that can be requested by its end hosts, and define the *request ratio* α_k^i as the probability that a requested chunk belongs to rank k.

Request Arrival. We assume that the arrival of chunk requests at any leaf node conforms to the Independent Reference Model (IRM), which is widely adopted to model cache accesses [3], [4]. Specifically, let random variable X_j be the rank of the j^{th} requested chunk at a leaf router v_i, then $X_1, X_2, \ldots$ are independent and identically distributed (i.i.d.). According to the request ratio defined above, we have $P(X_j = k) = \alpha_k^i$.

Multi-path Routing. We assume that the routing policy for chunk requests is *multi-path*. Formally, let N be the number of nodes in the system, then the routing policy can be characterized using a matrix $M = (\sigma_{i,j})_{N\times N}$, where $\sigma_{i,j} \in [0, 1]$ is the probability that node v_i forwards missed requests to node v_j. Since a request can be forwarded to multiple nodes, $\sum_j \sigma_{i,j}$ can be larger than 1.

Cache Replacement. We assume a simple LRU strategy for cache replacement. Consider the cache of a router as a sequence of slots. If a requested chunk is not in the cache, it would be brought from other routers and placed at the first slot of the cache. All other chunks in the cache are pushed down one position, and the chunk in the last slot is discarded. On the other hand, if the requested chunk is already in the

[1]In our model, we would use routers and nodes interchangeably

cache, it would be brought to the top slot, and all chunks that are previously ahead of it will be pushed down one position.

III. Model Analysis

This section presents models to evaluate the performance of hierarchical caches in CCN. We characterize the caching performance with *overall hit ratio* h^i, defined as the probability that a requested chunk is stored in the cache of node v_i. In addition, we are also interested in evaluating *rank-k hit ratio* h_k^i, defined as the probability that a requested chunk of rank-k is stored in the cache of node v_i. Before moving on to the models, we introduce some more notations that will be used later.

Let S_i be the number of slots in node v_i's cache. Then, for any $j \in [1, S_i]$, define $p_k^i(j)$ as the probability that any rank-k chunk is stored at the j^{th} slot of v_i. It easily follows that $b_k^i(j) = \sum_{t=1}^{j} p_k^i(t)$ is the expected number of rank-k chunks stored in the first j slots of v_i. For simplicity of notation, let $g_k^i = |C_k^i|$, and define $\beta_k^i = \alpha_k^i / g_k^i$ as the request arrival ratio of a specific rank-k chunk at node v_i.

A. *Leaf Node Model*

Let us consider the caching model for a single leaf node v_1, and develop an expression of hit ratios h^1 and h_k^1.

By the definitions of $p_k^i(j)$ and $b_k^i(j)$, the rank-k hit ratio is:

$$h_k^1 = \frac{\sum_{j=1}^{S_1} p_k^1(j)}{\sum_{j=1}^{M} p_k^1(j)} = \frac{b_k^1(S_1)}{g_k^1}, \tag{1}$$

and the overall cache hit ratio is:

$$h^1 = \frac{\sum_{k=1}^{K} h_k^1 \cdot \alpha_k^1}{\sum_{k=1}^{K} \alpha_k^1} \tag{2}$$

To determine $b_k^1(j)$, we construct a Discrete-Time Markov Chain (DTMC), which captures the dynamics for a specific chunk $c \in C_k^1$ in the cache of v_1, as shown in Fig. 2. In this Markov chain, each state represents the slot that c occupies: state j means that c is in the j^{th} slot; state $M = S_1 + 1$ means that c is outside of the cache. A transition is triggered when a request arrives to node v_1. Let $P_{i\to j}$ be the transition probability from state i to state j. In the following, we will determine all these transition probabilities.

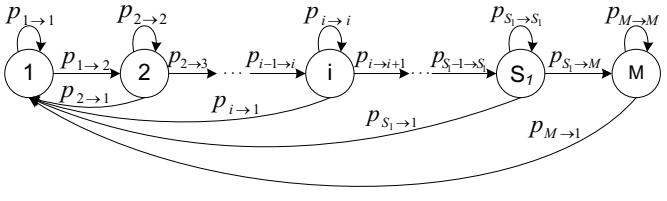


Fig. 2. The Markov chain of the leaf node Model.

$P_{i\to i},\ i \in [2, S_1]$ corresponds to the probability that any chunk in the first $i-1$ slots is requested:

$$P_{i\to i} = \sum_{t=1}^{K} \beta_t^1 \cdot b_t^1(i-1),$$

and $P_{M\to M}$ is the probability that c is not requested:

$$P_{M\to M} = 1 - \beta_k^1$$

$P_{i-1\to i},\ i \in [2, M]$ corresponds to the probability that neither c nor any other chunk in the first $i-2$ slot is requested:

$$P_{i-1\to i} = 1 - \sum_{t=1}^{K} \beta_t^1 \cdot \left[b_t^1(i-2) + I_{t=k} \right],$$

where I_A is a indictor function which takes 1 if predicate A is true, and 0 otherwise.

$P_{i\to 1},\ i \in [1, M]$ is the probability that c is requested:

$$P_{i\to 1} = \beta_k^1$$

Let $\vec{\pi} = [\pi(1), \pi(2), \ldots, \pi(M)]$ be the steady state distribution, then the balance equations are as follows:

$$\begin{cases} \pi(1) = \beta_k^1 \cdot \sum_{i=1}^{M} \pi(i) \\ \pi(i) = P_{i-1\to i} \cdot \pi(i-1) + P_{i\to i} \cdot \pi(i), \quad i \in [2, M] \\ \sum_{i=1}^{M} \pi(i) = 1 \end{cases}$$

According to our definition, $p_k^1(j) = g_k^1 \pi(j)$. Combining the above results, we have the following recursive equations:

$$p_k^1(j) = p_k^1(j-1) \cdot \frac{1 - \sum_{t=1}^{K} \beta_t^1 \cdot \left[b_t^1(j-2) + I_{t=k} \right]}{1 - \sum_{t=1}^{K} \beta_t^1 \cdot b_t^1(j-1)},\ i \in [2, M], \tag{3}$$

with the initial value $p_k^1(1) = \alpha_k^1$.

Using Eq(3), we can solve $p_k^1(j)$ for $j \in [1, S_1]$, and obtain $b_k^1(S_1)$. Then, h_k^1 and h^1 can be calculated using Eq(1) and Eq(2), respectively.

B. *Layer-1 Node Model*

Let us consider the caching model for a single layer-1 node v_0, which is connected with one leaf node v_1. We aim to evaluate the values of h^0 and h_k^0.

Different from the previous leaf node model, a cache hit of chunk c at node v_0 requires not only that c is in the cache of v_0, but also that c is not in the cache of v_1. In other words, the hit ratio of v_0 not only depends on its own state, but also depends on v_1's state. To this end, we introduce another variable to express hit ratios of node v_0.

Let $C_i(c) = j$ be the event that the chunk c is present in the j^{th} slot of node v_i; $C_i(c) = 0$ when c is not present in node v_i. Define $R_k^{m,n}(j)$ as the probability that a chunk c of rank k is not present in the cache of v_n condition that it is present in the j^{th} cache slot of v_m.

$$R_k^{m,n}(j) = \Pr\left\{ C_n(c) = 0 \mid C_m(c) = j \right\}, \quad j \in [1, S_m]$$

The variable R defined above characterizes the relation between two nodes. Specifically, $R_k^{m,n}(j)$ reflects how useful a rank-k chunk stored in the j^{th} slot of node v_m is for node v_n. Using R, we can express the rank-k hit ratio of node v_0 as:

$$h_k^0 = \frac{\sum_{j=1}^{S_0} p_k^0(j) \cdot R_k^{0,1}(j)}{g_k^0} \tag{4}$$

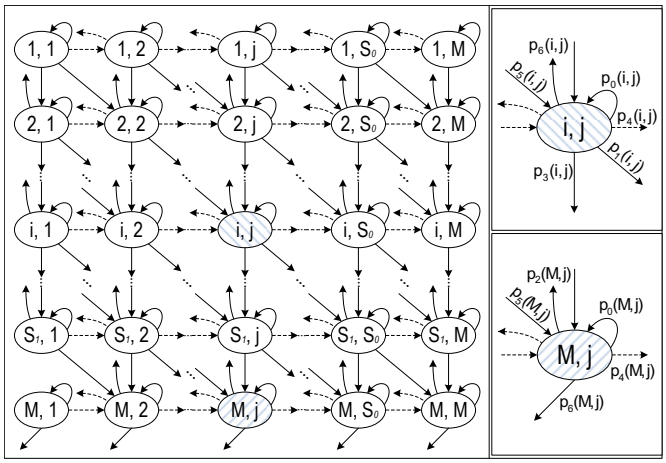


Fig. 3. The two-dimensional Markov chain of the layer-1 node model. On the right side, we list two states (i, j) and (M, j) to illustrate different state transitions.

The overall hit ratio of v_0 can be derived in the similar way as the leaf node model using Eq(2), with α_k^0 expressed as:

$$\alpha_k^0 = \frac{\alpha_k^1(1-h_k^1)}{\sum_{t=1}^{K}\alpha_t^1(1-h_t^1)}$$

Then, we define $D_k^{m,n}(j)$ as the expected number of rank-k chunks present at the first j^{th} slots of v_m, but not present in v_n's cache. $D_k^{m,n}(j)$ would be expressed as:

$$D_k^{m,n}(j) = \sum_{i=1}^{j} p_k^m(i) \cdot R_k^{m,n}(i) \tag{5}$$

By modeling the cache of v_0 using a one-dimensional Markov chain, we can derive the following recursive equation in a similar way as the leaf node model:

$$p_k^0(j) = p_k^0(j-1) \cdot \frac{1-\sum_{t=1}^{K}\beta_k^1 \cdot \left[b_t^1(S_1)+D_t^{0,1}(j-2)+R_k^{0,1}(j-1)I_{t=k}\right]}{1-\sum_{t=1}^{K}\beta_t^1 \cdot \left[b_t^1(S_1)+D_t^{0,1}(j-1)\right]},$$

where $p_k^0(1) = \alpha_k^0$.

To determine $R_k^{0,1}(j)$, we construct a two-dimensional discrete-time Markov chain, as shown in Fig. 3. In this Markov chain, state (i, j) means that c is present in the i^{th} slot of node v_1 and j^{th} slot of node v_0, where $i \in \{1, 2, ..., S_1, M\}, j \in \{1, 2, ..., S_0, M\}$. Let $\vec{\pi} = [\pi(1,1), \pi(1,2), ..., \pi(1,M), \pi(2,1), ..., \pi(M,M)]$ be the steady state probabilities. Then, $R_k^{0,1}(j)$ can be expressed as:

$$R_k^{0,1}(j) = \frac{\pi(M,j)}{\sum_{i=1}^{S_1}\pi(i,j)+\pi(M,j)} \tag{6}$$

The remaining problem is how to solve this Markov chain (please refer to Appendix A for the transition probabilities and balance equations). Since the number of states in this Markov chain is $(S_0+1)\times(S_1+1)$, solving it numerically can be difficult when the cache sizes are very large or there are multiple leaf nodes. On the other hand, analytical solutions are more desirable since they allow more efficient computation, and can be extended.

Solving this Markov chain analytically is also tricky since the transition probabilities contain variables $R_k^{0,1}(j)$, which are functions of $\vec{\pi}$. This causes a loop which prevents a straightforward solution. To address this problem, we first transform the balance equations (see Appendix A), and define $\vec{\pi}_j = [\pi(1,j), \pi(2,j), \ldots, \pi(M,j)]$ which corresponds to the steady state probabilities of the j^{th} vertical chain. Then, we have two observations: i) $\vec{\pi}_j, j > 1$ can be solved once $\vec{\pi}_{j-1}$ is determined; ii) The value of $R_k^{0,1}(j)$ only depends on the ratio of elements in $\vec{\pi}_j$.

Based on the first observation, we can solve this two-dimensional Markov chain by iteratively solving $\vec{\pi}_1, \vec{\pi}_2, \ldots, \vec{\pi}_M$ in sequence. The second observation allows us to bootstrap this iteration by setting the right side of the first transformed balanced equation (see Appendix A) to 1. In the following, we will give details of this solution.

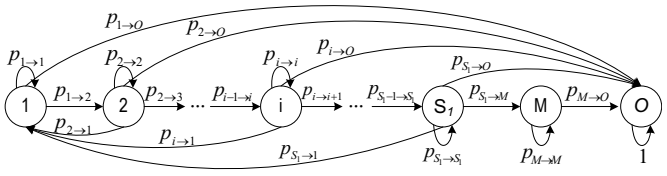


Fig. 4. MC_j, the j^{th} vertical Markov chain.

First, we vertically divide this two-dimensional Markov chain into $S_0 + 1$ one-dimensional Markov chains $\{MC_1, MC_2, ..., MC_{S_0}, MC_M\}$. Fig. 4 shows the transition graph for MC_j, where state i means that c is in the i^{th} slot of v_1; state M means that c is not in v_1. We add an absorbing state O to aggregate all the other states that c is not present in the j^{th} slot of v_0. Once the system reaches state O, the probability of moving out of O is 0. The transition probabilities of this Markov chain can be derived easily.

Then, we show how to iteratively solve Markov chains from MC_1 to MC_{S_0}. As seen in Fig. 4, the Markov chain has no steady state, since there is an absorbing state. Here, we adopt the approach given in [5] as follows.

Let φ_j^i denote the average number of times that state i is visited before the chain reaches the absorbing state O. Then, we have:

$$\varphi_j^i = q_j^i + \sum_u \varphi_j^u P_{ui}, \quad i, u \in \{1, ..., S_1, M\}, \tag{7}$$

where q_j^i is the probability that MC_j starts at state i, and P is the transition probability matrix of MC_j. Let $\vec{q}_j = \{q_j^1, q_j^2, ..., q_j^M\}$ represent the initial state distribution of MC_j. In our case, the initial-state distribution of MC_1 is $\vec{q}_1 = \{1, 0, \ldots, 0\}$; $\vec{q}_j (j > 1)$ can be calculated using the steady state probability of MC_{j-1} and the transition probabilities from

MC_{j-1} to MC_j:

$$q_j^i = \begin{cases} 0, & i = 1 \\ \varphi_{j-1}^{i-1} \cdot P_1(i-1, j-1), & i \in [2, S_1] \\ \varphi_{j-1}^{S_1} \cdot P_1(S_1, j-1) + \varphi_{j-1}^{M} \cdot P_4(M, j-1), & i = M \end{cases} \tag{8}$$

Thus, we can determine $\varphi_j^1, \ldots, \varphi_j^{S_1}, \varphi_j^M$, and solve $R_k^{0,1}(j)$ using:

$$R_k^{0,1}(j) = \frac{\varphi_j^M}{\sum_{i=1}^{S_1} \varphi_j^i + \varphi_j^M} \tag{9}$$

With $R_k^{0,1}(j)$, we can calculate $p_k^0(j)$. Repeat this process until we have all the values of $R_k^{0,1}(j)$ and $p_k^0(j)$. Finally, we can obtain h_k^0 and h^0 according to Eq(4) and Eq(2).

In the above model, we consider the case that the layer-1 node v_0 is only connected with one leaf node v_1. It is possible to extend this model to cases of multiple leaf node $v_1, v_2, \ldots, v_n$. For details, please refer to the *extended layer-1 node model* in our technical report[6].

IV. Numerical Results

In this section, we present numerical results obtained based on our models. We aim to show the accuracy of our model with simulations, and provide some understanding to guide the design of hierarchical caches in CCN. For simplicity, we will term the layer-1 node as root node.

A. Experiment Setup

We use the OMNeT++ [7], a discrete-event simulation package, to construct the content-centric networking environment [1]. Rather than implementing a full-fledged one, we only include the basic functions of CCN: multi-path routing, chunk-based caching and receiver-driven transport protocol. The forwarding table of each node is computed using the CCNdc method introduced in the CCNx project [2].

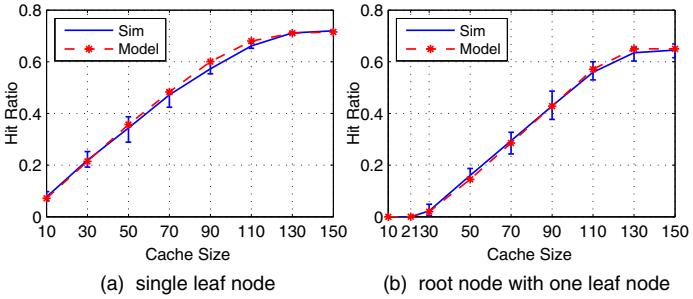
We developed a simplified version of ProWGen [8] to generate a large pool of data chunks[2]. Then, we partition these trunks into ten popularity ranks from 1 to 10. For each leaf router, we randomly sample 500 chunks from the chunk pool.

B. Model Verification

In order to verify the accuracy of our models, we calculate hit ratios for different cache sizes, and compare them to the simulation results. To verify our three models, we consider three different kinds of nodes: (1) single leaf node, (2) root node connected with one leaf node, and (3) root node connected with multiple leaf nodes. For each case, we repeat the simulations for ten times to cover the randomness of chunk requests.

Fig. 5(a) reports the hit ratios of a leaf node with cache size ranging from 10 to 150. The error bars reflects the result variance of the 10 simulation runs. Note that our modeling results are quite close to the simulative results, with difference less than 2.5%.

[2]It is shown that ProWGen can generate workloads that are similar to empirical traces [9]



(a) single leaf node (b) root node with one leaf node

Fig. 5. The relation between hit ratio and cache size for (a) single leaf node, and (b) root node connected with one leaf node.

Fig. 5(b) shows the relationship between hit ratio and cache size for a root node connected with one leaf node. We observe that: 1) the hit ratio is close to 0 when the cache size is below 21. The reason is that the chunks stored at the first $j \leq 21$ slots are also stored in the leaf node with a high probability. As a result, $R_k^0(j)$ is negligible for these small j, and the hit ratio is thus very low according to Eq(4); 2) after the cache size of the root node becomes larger than 21, the hit ratio climbs quickly, but will not increase much after the size exceeds 130. This is because these slots at the rear of the cache are mostly occupied by cold contents which are seldom requested (see the next experiment for details). These content contributes little to the overall hit ratio according to Eq(2).

By the above two observations, our model can be employed to find the two threshold values s_1 and s_2 (here $s_1 = 21$, $s_2 = 130$). With the cache size of root node set between s_1 and s_2, we can expect a quick growth of overall hit ratio.

It is also seen that the shape of curves in Fig. 5(a) and (b) are remarkably different, owing to the fact their request arrival patterns are different. For the leaf node, requests arrive at it according to the IRM assumption (see Section II); while the root node is fed with non-IRM requests. This confirms the "filter effect" previously studied in [10]: lower-level nodes selectively filter out chunk requests, and the resultant requests are no longer independent of each other.

We continue to consider the scenarios of multiple leaf nodes. Fig. 6 plots the hit ratio of the root node when the leaf node number $n = 2, 3, 4, 5$. As we apply approximation in the calculation[6], the accuracy drops a little, but the error is still bounded by 5%. Note that the hit ratios in these cases are all above 0 even when the cache size is very small. This is due to the independence of chunk requests from leaf nodes, i.e., the chunk recently requested by node v_1 can still be requested by v_2 within a short period. Another interesting observation is that as the number of leaf nodes increases, the shapes of curves become more and more similar to that of the leaf node in Fig. 5(a). This may imply that the request arrival at root node tends to follow IRM again, i.e., the "filter effect" becomes less noticeable with the increase of leaf node number.

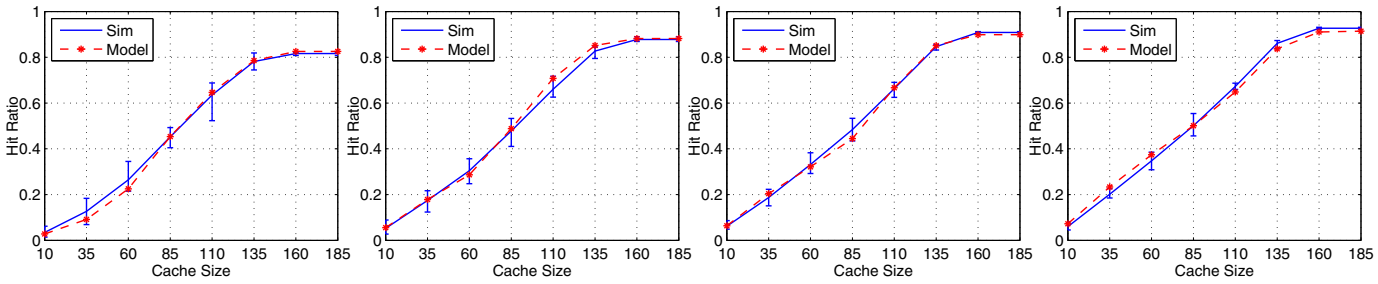


Fig. 6. The relation between hit ratio and cache size of a root node, when the number of leaf nodes is (a) $n = 2$, (b) $n = 3$, (c) $n = 4$, and (d) $n = 5$.

C. *How Large Cache Do We Need for the Root Node?*

In the previous experiments, we have mentioned the "filter effect" imposed by leaf nodes, and gained the intuition that the root node should have a relatively large cache to maintain a relatively high hit ratio. In this experiment, we justify this claim by studying the minimum cache size for the root node to maintain a hit ratio above 0.5. We vary the number of leaf nodes connected to the root node from 1 to 6, and the results are shown in Fig. 7. Note that each point is averaged over ten runs by varying the input (randomly generated chunk requests). We have the following two observations:

1) The root node requires a much larger cache size than the leaf nodes. As shown in Fig. 7, The ratio is 3× when there is only one leaf node, and 2× when there are six leaf nodes. This may indicate that using the same replacement policy in hierarchical caches without any cooperations is inefficient. If the root node employs a different replacement policy, we may be able to reduce the required cache size for the root node.

2) The requirement for cache size is lower when there are more leaf nodes. This is because different nodes have different popularity assignment (α_k^i) in our model, and the requests for chunks of different popularity tend to be distributed more uniformly with the increase of leaf nodes.

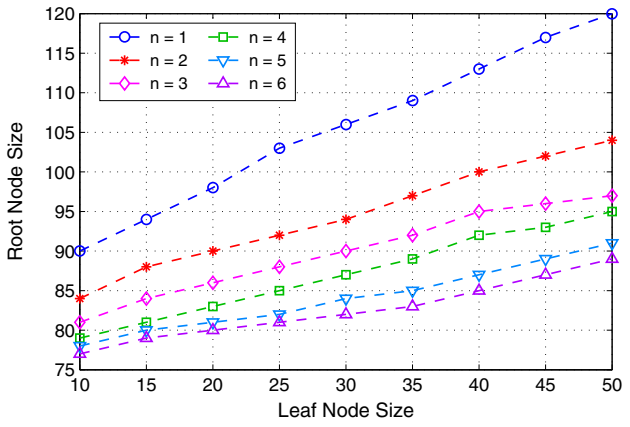


Fig. 7. The cache size requirements for root node when connected with different number of leaf nodes.

V. Related Work

Content caching has been extensively studied in the research area of web cache. Similar as caches in CCN, web caches can store the recently requested file for a short time so that subsequent requests can be satisfied locally. Dan *et al.* [11] study the LRU and FIFO cache replacement strategies for a single cache node which is fed with IRM file requests. As the computation complexity of directly calculating cache hit probability is very high, they develop an approximated approach to estimate the hit probability. Che *et al.* [3] propose another approximation approach to evaluate the performance of two-level caches by assuming LRU replacement policy and IRM request arrival. They approximate the evaluation by assuming a constant value for inter-arrival times between two consecutive requests for a same document without cache misses. This assumption is claimed to be good when they are a large number of files. Based on [11], Rosensweig *et al.* [4] analyze a more general cache network where there is no fixed topology. The problem of multi-cache problem is decomposed into a set of single-cache problems, which can be solved independently. However, they assume that the cache miss stream to be IRM, which would affect the accuracy of their model.

There are some efforts on analyzing the caching performance in Content-Centric Networks, but they are mostly based on simulations or experiments [12], [13], [14], [15]. However, for a fundamental understanding of the performance of CCN caches, some analytical models are still needed. Towards this goal, Psaras *et al.* [16] try to use Continuous-Time Markov Chain (CTMC) to capture the caching dynamics. They first introduce a simple model for one node, and then extend the model to multiple nodes. However, as the time is continuous, the model is not easy to solve, and not much results are given. Carofiglio *et al.* [17] take another approach by assuming the requests arrivals conform to Markov Modulated Rate Process (MMRP). But to make the chunks misses to be independent (similar to IRM), they assume the file size to be "memoryless", i.e., geometrically distributed.

To sum up, though web cache has been extensively studied, they can not be directly applied to CCN hierarchical caching systems due to features including chunk-based caching and multi-path request routing. In addition, existing continuous

models are complicated and are short in giving valuable guidance for cache design.

VI. Conclusion

This paper models and evaluates the performance of hierarchical caches formed by CCN content routers. The main characteristic that we study is cache hit ratio, including overall hit ratio and rank-k hit ratio. Our models can be used to efficiently calculate the hit ratio for up to two layers of caches in CCN. The accuracy of the proposed models is validated through extensive experiments. In addition, our models reveal the relationship between cache size and hit ratio for different layers of caches. This can help ISPs to strategically choose the right size for the CCN caches. Our findings also gives some valuable understanding on the "filter effect" imposed by lower-layer caches in CCN.

Acknowledgements

This research was funded by the National Grand Fundamental Research 973 Program of China (No. 2010CB328105, No. 2009CB320504), and the National Natural Science Foundation of China (No. 60932003, No. 61020106002). We would like to thank the anonymous reviewers for their helpful comments and suggestions.

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. of ACM CoNEXT*, 2009.

[2] CCNx project. http://www.ccnx.org/.

[3] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: modeling, design and experimental results," *IEEE Journal on Selected Areas in Communications*, 2002.

[4] E. J. Rosensweig, J. Kurose, and D. Towsley, "Approximate models for general cache networks," in *Proc. of IEEE INFOCOM*, 2010.

[5] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*, 2nd ed. John Wiley & Sons, 2001.

[6] "Modeling hierarchical caches in content-centric networks," Tech. Rep., 2012. [Online]. Available: http://qos.cs.tsinghua.edu.cn/zxjia/MHC-CCN-12.pdf

[7] OMNeT++ homepage. http://www.omnetpp.org/.

[8] M. Busari and C. Williamson, "ProWGen: A synthetic workload generation tool for simulation evaluation of web proxy caches," *Computer Networks*, 2002.

[9] O. Saleh and M. Hefeeda, "Modeling and caching of peer-to-peer traffic," in *Proc. of IEEE ICNP*, 2006.

[10] C. Williamson, "On filter effects in web caching hierarchies," *ACM Trans. on Internet Technology (TOIT)*.

[11] A. Dan and D. Towsley, "An approximate analysis of the lru and fifo buffer replacement schemes," *ACM SIGMETRICS Perform. Eval. Rev.*

[12] G. Carofiglio, V. Gehlen, and D. Perino, "Experimental evaluation of memory management in content-centric networking," in *Proc. of IEEE International Conference on Communications(ICC)*, 2011.

[13] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing," *Technical Report*, 2011.

[14] ——, "On sizing CCN content stores by exploiting topological information," in *Proc. of IEEE INFOCOM Workshops*, 2012.

[15] C. Fricker, P. Robert, J. Roberts, and N. Sbihi, "Impact of traffic mix on caching performance in a content-centric network," in *Proc. of IEEE INFOCOM Workshops*, 2012.

[16] I. Psaras, R. G. Clegg, R. Landa, W. K. Chai, and G. Pavlou, "Modelling and evaluation of CCN-caching trees," *NETWORKING 2011*, 2011.

[17] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino, "Modeling data transfer in content-centric networking," in *Proc. of IEEE International Teletraffic Congress(ITC)*, 2011.

Appendix
Transition Probabilities and Balance Equations of the Markov Chain in Section III-B

Transition Probabilities

i) For $1 \le i \le S_1, 1 \le j \le S_0$:

$$\begin{cases} P_0(i,j) = \sum_{t=1}^{K} \beta_t^1 \cdot [b_t^1(i-1) + I_{i=1}] \\ P_1(i,j) = 1 - \sum_{t=1}^{K} \beta_t^1 \cdot [D_t^{0,1}(j-1) + b_t^1(S_1) - p_t^1(i) + I_{t=k}] \\ P_3(i,j) = \sum_{t=1}^{K} \beta_t^1 \cdot [D_t^{0,1}(j-1) + b_t^1(S_1) - b_t^1(i)] \\ P_6(i,j) = \beta_t^1 \\ P_2(i,j) = P_4(i,j) = P_5(i,j) = 0 \end{cases}$$

ii) For $i = M, 1 \le j \le S_0$:

$$\begin{cases} P_0(M,j) = \sum_{t=1}^{K} \beta_t^1 \cdot [b_t^1(S_1) + D_t^{0,1}(j-1)] \\ P_4(M,j) = 1 - \sum_{k=1}^{K} \beta_t^1 \cdot [b_t^1(S_1) + D_t^{0,1}(j-1) + I_{t=k}] \end{cases}$$

iii) For $j = M$:

$$P_3(i,M) = 1 - \sum_{t=1}^{K} \beta_t^1 \cdot [b_t^1(i-1) + I_{t=k}]$$

iv) For $i = M, j = M$:

$$P_0(M,M) = 1 - \beta_k^1$$

Balance Equations

$$\begin{cases} \pi(1,1) = \sum_{u=2}^{S_1} P_6(u,1)\pi(u,1) + \sum_{u=1}^{M} P_6(M,u)\pi(M,u) + P_0(1,1)\pi(1,1) \\ \pi(i,1) = P_3(i-1,1)\pi(i-1,1) + P_0(i,1)\pi(i,1) & i \in [2,S_1] \\ \pi(M,1) = P_0(M,1)\pi(M,1) \\ \pi(1,j) = \sum_{u=2}^{M} P_6(u,j)\pi(u,j) + P_0(1,j)\pi(1,j) & j \in [2,M] \\ \pi(i,j) = P_3(i-1,j)\pi(i-1,j) + P_0(i,j)\pi(i,j) \\ \qquad + P_1(i-1,j-1)\pi(i-1,j-1) & i,j \in [2,S_0] \\ \pi(M,j) = P_1(S_1,j-1)\pi(S_1,j-1) + P_4(M,j-1)\pi(M,j-1) \\ \qquad + P_3(S_1,j)\pi(S_1,j) + P_0(M,j)\pi(M,j) & j \in [2,M] \\ \sum_{i=1}^{M}\sum_{j=1}^{M} \pi(i,j) = 1 \end{cases}$$

Transformed Balance Equations with $\pi(i,j)$ changed to π_j^i

$$\begin{cases} (1 - P_0(1,1))\pi_1^1 - \sum_{u=2}^{S_1} P_6(u,1)\pi_1^u = \sum_{u=1}^{M} P_6(M,u)\pi_u^M \\ (1 - P_0(i,1))\pi_1^i - P_3(i-1,1)\pi_1^{i-1} = 0 & i \in [2,S_1] \\ (1 - P_0(M,1))\pi_1^M = 0 \end{cases}$$

$$\begin{cases} (1 - P_0(1,j))\pi_j^1 - \sum_{u=2}^{M} P_6(u,j)\pi_j^u = 0 & j \in [2,M] \\ (1 - P_0(i,j))\pi_j^i - P_3(i-1,j)\pi_j^{i-1} & i,j \in [2,S_0] \\ \qquad = P_1(i-1,j-1)\pi_{j-1}^{i-1} \\ (1 - P_0(M,j))\pi_j^M - P_3(S_1,j)\pi_j^{S_1} & j \in [2,M] \\ \qquad = P_1(S_1,j-1)\pi_{j-1}^{S_1} + P_4(M,j-1)\pi_{j-1}^M \end{cases}$$