Content Routing and Lookup Schemes using Global Bloom Filter for Content-Delivery-as-a-Service

Yichao Jin, Yonggang Wen, *Member, IEEE,* and Weiwen Zhang, *Student Member, IEEE*

***Abstract*—The dramatic growth of user-generated contents (UGCs) transforms the digital media value chain, and stresses current content distribution network (CDN). In order to deliver UGCs in an efficient and economical fashion, we have proposed content-delivery-as-a-service (CoDaaS) by leveraging cloud computing technology. However, due to the exponential increase of Internet traffic (especially the UGCs), traditional hashing-based content routing and lookup scheme in CDNs suffers from high delay and consequent inefficient delivery. This paper introduces a global compressed counting BF (CCBF) into CoDaaS to address this issue. By equipping it with the global CCBF, our system is able to check early on for the existence of any specific content among all the peering surrogates, before any local checking on each cache node. Based on this global CCBF, we propose two content routing and lookup mechanisms (i.e., parallel and cut-through schemes) to reduce the delay for better user experience. The comparative performance of those approaches is verified via both mathematical modeling and experimental simulation. The results show that for light traffic load, the average response time can be saved by up to 65.2% compared with traditional methods. In addition, the impacts and overheads of different synchronization schemes are also quantified to provide valuable insight for further optimizations.**

***Index Terms*—Compressed counting BF, content delivery as a service, content lookup, content routing.**

I. Introduction

THE USER generated contents (UGCs) are dominating the Internet traffic nowadays. As one of the fastest growing forms of contents, the percentage of the user generated content consumption over total Internet usage will exceed the 70 % threshold by the end of 2013 [1]. It is predicted that our current IP multimedia subsystem (IMS) will rapidly transform into IMS 2.0 [2], where the UGCs will play the most important role among all kinds of content resources.

The rapid growth as well as the long-tail nature and unique characteristics (e.g., conversational media, social interaction, etc.) of UGCs posits significant challenges for its efficient delivery over existing content distribution networks (CDNs). The leading CDNs service providers, at present, usually tailor their network architecture and operational structure mostly toward popular contents. Consequently, a profitable delivery for heavily long-tailed contents over current CDNs is difficult. Novel schemes are urgently demanded.

To tackle this challenge, we have previously proposed a content delivery scheme (i.e., content-delivery-as-a-service (CoDaaS) [3], [4]). This solution leverages the emerging cloud computing technologies to elastically allocate resources (e.g., storage and bandwidth resources, etc.) to meet the dynamic applications' demands. As a result, unnecessary system wasting can be reduced to save the UGCs delivery cost.

In addition to the cost reduction, our proposed CoDaaS also aims to provide the best possible user experience to the content consumers. Specifically, one of the quality-of-service (QoS) objectives is to minimize the user perceived request response time [5]. When contents are distributed over CDNs, the user request will be first routed to a specific server, and the server will conduct a lookup process to check the requested content locally. Those two processes have great impacts on the request response time. As a result, orchestrating those two processes is the key problem in achieving an improved user experience in CoDaaS.

In this paper, we introduce a global bloom filter (BF) into CoDaaS to reduce the request response time. Specifically, the global BF is maintained by all the VMs in CoDaaS (i.e., surrogate servers in CDNs), to indicate the existence of the cached contents. It provides the capability to make an early decision on whether the requested content is cached in the CoDaaS before the request is routed to the designated server.

Compared to traditional approaches in CDNs [6]–[8] where the two processes are normally executed in sequential order, our approach executes content routing and lookup processes in parallel. As a result, we improve the QoS by reducing the high lookup latency and the long response time due to the sequential execution order, to provide a better user experience.

The contributions of this paper are three-fold.

1) We propose two content routing and lookup schemes (i.e., parallel and cut-through), by using the global BF to reduce the response time.
2) We introduce a mathematical model based on open Jackson network to justify the potential gain of our proposed content routing and lookup schemes. We also mathematically analyze their limitations and overheads.
3) We quantify the performance gain and the overhead of our proposes schemes via a system-level simulation. Our numerical results suggest that, for light-loaded traffic (e.g., Twitter-like applications), the average request re-

Manuscript received July 30, 2012; revised December 05, 2012; accepted March 3, 2013.

The authors are with the School of Computer Engineering, Nanyang Technological University, 639798, Singapore (e-mail: yjin3@ntu.edu.sg; ygwen@ntu.edu.sg; wzhang9@ntu.edu.sg).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSYST.2013.2253041

sponse time for our proposed content routing and lookup schemes can be reduced by up to 65.2%, compared to the sequential one. Our simulation results also suggest mechanisms to optimize the proposed schemes to reduce the impact of false positive inherent to BF, and false negative and the traffic overhead, resulting from the synchronization for BF.

This paper provides fundamental insight to understanding the benefits of using global BF for content routing and lookup, and offers operational guidelines to optimize the design of CoDaaS for best possible user experience.

The remainder of this paper is organized as follows. In Section II, we review several related works. In Section III, we illustrate the system architecture of CoDaaS, and the content routing and lookup process. In Section IV, we propose two novel content routing and lookup schemes based on a global BF in comparison with the traditional one. In Section V, we introduce a mathematical model to analyze the mean response time of all three content routing and lookup schemes. In Section VI, we investigate the potential overhead of our proposed schemes. In Section VII, we outline the simulation environment and settings; then the numerical and analytical results follow. In Section VIII, we conclude the paper and point out future work.

II. Related Work

A number of studies focused on the content routing strategy in CDNs environment. At present, the most widely used content routing method in large scale CDNs is hashing-based method proposed by Karger *et al.* [6]. In this approach, the contents are routed to desirable surrogate by mapping the hash value of their URL to the IP address of each surrogate. In [9], the authors applied complex queries to select the corresponding surrogate which holds the desired content in distributed hash tables (DHTs) structure. All the requests for the same group of contents will be redirected to one designated CDN server under this mechanism. A semi-hashing-based scheme [7] was proposed to strike a balance between local hit ratio and cluster hit ratio in CDNs. It optimally splits the disk space of surrogates into two parts; one is for the popular content caching, and the other is for the collaboration with other CDN servers using hash-based solution. J. Chang *et al.* proposed an efficient content service discovery mechanism to handle with the dual-stack DHTs based cloud model [10].

Bloom filter (BF) [11] was also used intensively for network applications, under the information explosion recently [12]. In particular, the BF is a space-efficient structure to represent massive data. Cache digest [13] was an example which used BF to indicate the existence of local cached replications. It had been used in Squid web proxy cache [14] to reduce bandwidth consumption and improve response time. Fan *et al.* [15] introduced counting bloom filter (CBF) to improve the Internet cache protocol (ICP). BF was also used to route content within Internet routers [16]. Besides, the proposal of compressed BF [17] yielded a dramatic reduction in the data size that is needed to be transmitted over networks.

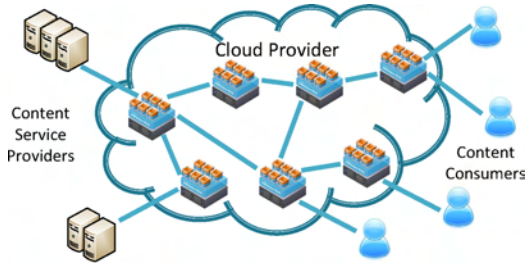


Fig. 1. Systematic overview of CoDaaS.

Our work differs from this pervious research in several aspects. First, we use a global BF to jointly optimize the two processes of content routing and lookup for content delivery over CoDaaS. Second, our design objective is to reduce the request response time, which can be translated into a better QoS. Third, we introduce open Jackson network [18] to generate analytical solutions for our proposed scheme. Finally, the performance gain is verified by both theoretical models and a system-level simulation over CDNsim [19].

III. System Overview

In this section, we first present a schematic overview on the CoDaaS architecture and the content routing and lookup process.

A. CoDaaS Architecture

Fig. 1 shows the architecture of the CoDaaS. There are mainly three components in this system, including a media cloud, content providers and content consumers. The media cloud consists of a list of interconnected virtual machines (VMs), which are instantiated in geographically distributed data centers. The set of VMs form a CDN overlay dynamically to cache and render contents in a profitable way. Each content provider offers a platform for users to distribute contents and browse the published contents they are interested in, with a specified service level agreement (SLA). The published contents by consumers are stored in the origin servers owned by content providers. The content consumers can experience a better QoS with the controlled content distribution. The resources consumed by the CDN overlay can be scaled up and down to meet application demand, thus reducing the total cost of ownership. More details about CoDaaS can be found in [3] and [4].

B. Content Routing and Lookup in CoDaaS

In Fig. 2, we illustrate the joint process of the content routing and lookup for CoDaaS. The content consumers are from different regions to get access to CoDaaS. When they request a content, the request, after entering the CoDaaS system, will be routed from its ingress point to the designated server. Once the request arrives, the designated server will do a content lookup in its local storage to determine whether the content has been cached. In case of a cache hit, the content will be served directly from this designated server to the users. In case of a cache miss, the content will be retrieved from the origin server and then served to the users. In this paper, we aim

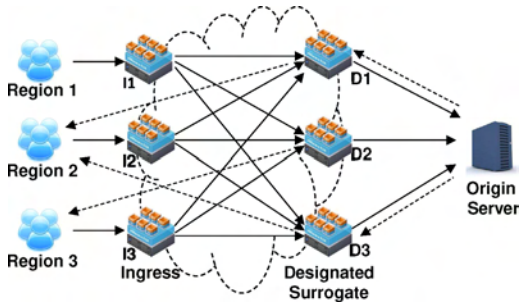


Fig. 2. Content routing and lookup scheme for CoDaaS.

to optimize these two processes to reduce the request response time, by introducing a global BF.

IV. Content Routing and Lookup Schemes

In this section, the concept of the BF will be first discussed to provide necessary foundations for the design of improved content routing and lookup schemes. Then we propose two alterative schemes, based on the usage of a global BF, to reduce the request response time.

A. Global Bloom Filter

1) *Bloom Filter and Its Variation:* In this paper, we use a CCBF, which is a variation of standard BF, for the improvement of content routing and lookup scheme in CoDaaS.

A standard BF is a bit array of m bits to represent a set of n elements $\mathbf{S}$, by using k independent hash functions $h_1, h_2...h_k$. For each element s in $\mathbf{S}$, it sets each $h_i(s), i = (1, 2, ..., k)$ bit to 1 for insertion. To check the existence of any element x, it checks whether all the $h_i(s), i = (1, 2, ..., k)$ bits are 1. Due to the hashing collision, a false positive [11] may occur with the probability as

$$f_{bf} \approx (1 - e^{-kn/m})^k. \quad (1)$$

The CBF is derived from the standard BF. It uses m fixed size integers instead of single bit for presence. For an insertion or deletion operation, the corresponding counters increase or decrease by 1. As a result, CBF provides the deletion capability. In this paper, we use four bits counters to construct our CBF, which has been shown to be sufficient for most network applications [15].

To reduce the transmission size of CBF as a message, compressed counting bloom filter (CCBF) is proposed. This paper adopts the multilayer compressed counting bloom filter (ML-CCBF) [17], which uses the run-length code to encode CBF messages. It has been proved that this encoding process can save up to 50% transmission traffic. Moreover, we also use the delta compression scheme to further reduce the transmission size by only transferring the changes for each layer in our CCBF.

2) *Synchronization Schemes:* Synchronization is a process of establishing the consistency of the global bloom filter among all the surrogates by exchanging synchronization messages. In this paper, we consider two alternative trigger schemes for the synchronization of our CCBF. One is the periodic mode, in which each surrogate broadcasts its update in a regular interval. The other is the event-driven model, in which the broadcast is triggered by some predetermined events (e.g., the reception of a fixed number of requests). These two synchronization modes will have different impact on the performance gain and the overhead, as verified in Section VII.

B. Traditional Content Routing and Lookup Schemes

Fig. 3(a) illustrates the workflow of the traditional scheme, in which content routing and lookup processes are executed sequentially. In a DHT-based scheme, the routing process takes on average $\lceil \log_2 N \rceil$ hops to reach the designated server, where N is the number of the active surrogates in CoDaaS. The content lookup process starts as soon as the request is routed to its designated server. For a cache hit, the designated server immediately replies to the user. Otherwise, it will retrieve the content from the origin server and then render it to the consumer.

C. Proposed Content Routing and Lookup Schemes

The key idea in this research is to use a global CCBF to determine whether the request content has been cached in CoDaaS, at the ingress point of each request. This early decision offers a chance for CoDaaS to retrieve the content in parallel with the request routing. Using this insight, we propose two content routing and lookup schemes for CoDaaS, including a parallel scheme and a cut-through scheme.

1) *Parallel Schemes:* Fig. 3(b) illustrates the workflow of the parallel scheme. For a cache hit, the workflow is exactly the same as the sequential process. In the case of a cache miss, the ingress directly acquires the content from its origin server. The ingress server, upon receiving the content from the origin server, renders the content to the consumer and forward a copy to the designated server for caching purpose. The parallel scheme executes content routing and content lookup in parallel, thus reducing the request response time.

2) *Cut-through Schemes:* Fig. 3(c) illustrates the workflow of the cut-through scheme. By using this scheme, in case of a cache hit, there is still no change made by this scheme compared to the sequential workflow. While in case of a cache miss, the ingress redirects the request from the user directly to the origin server. At the same time, the request is also routed to the designated server, which will retrieve a copy of the content directly from the origin server for caching purpose. As a result, the cut-through scheme reduces the request response time significantly by cutting through the CoDaaS system. Nevertheless, the drawback of this scheme is that it increases the traffic load to the origin server.

In the next three sections, we will compare the performance of these three schemes both analytically and numerically.

V. Mathematical Modeling

In this section, we develop a mathematical model to analytically present the mean response time of the aforementioned three schemes for content routing and lookup in CoDaaS. For clarity and ease of reference, we summarize the important notations used in this paper in Table I.

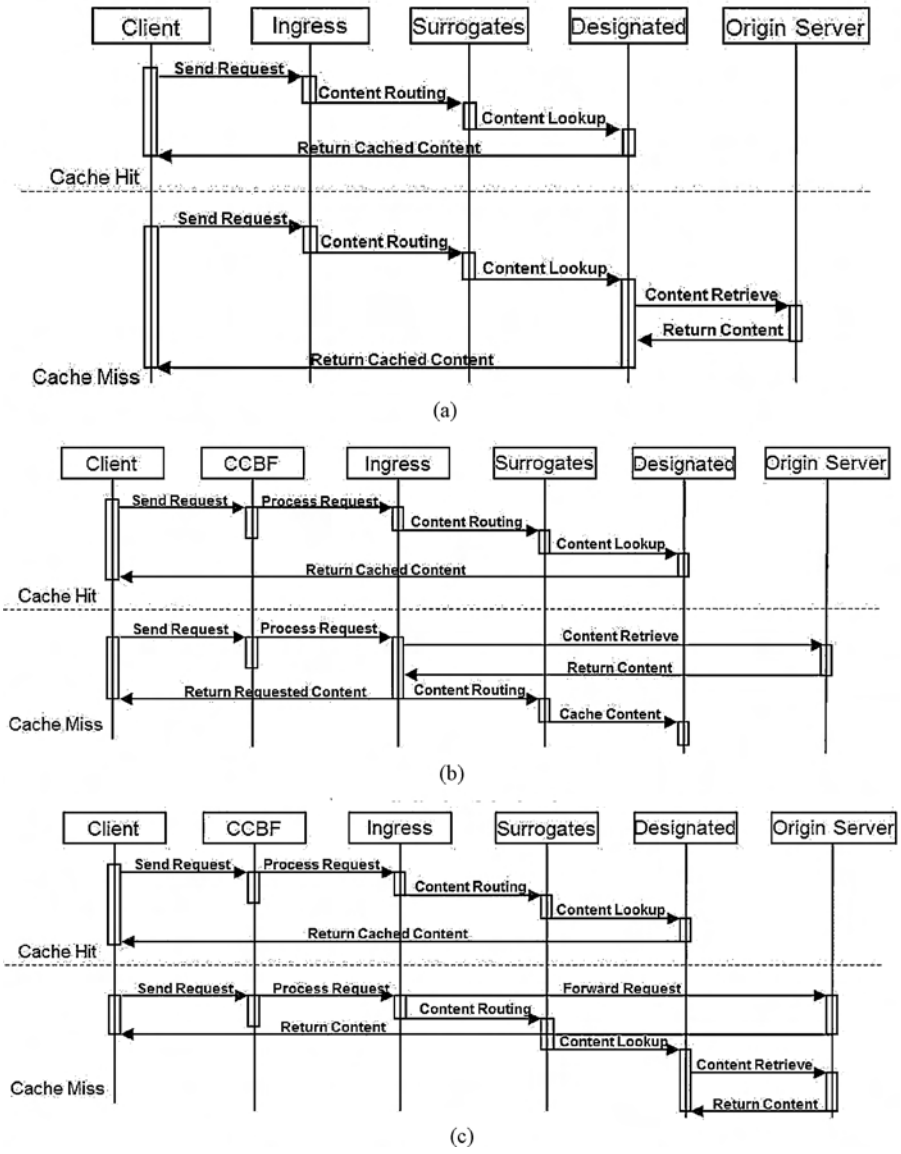


Fig. 3. Workflow diagrams for three alternative content routing and lookup schemes in CoDaaS. (a) Sequential scheme. (b) Parallel scheme. (c) Cut-through scheme.

A. Queueing Network Modeling

We use open Jackson queueing network [18] based model to characterize the service components in CoDaaS. A Jackson network is constructed by a network of queues, where the arrivals at each queue are modeled as a Poisson process, and the service times follow the exponential distribution. In an open Jackson network, there are external job arrivals as well as departures from the system. We model each service component in CoDaaS as a queue. The request from a user to consume contents equals the job arrival at the first queue. Once the user receive the first bit of the requested content, it maps to completing the job in the last queue. The aggregated queueing time at all the queues corresponds to the waiting time for available resources. The service time of a job at each queue refers to the actual process time of a request. As a result, the content routing and lookup behavior can be viewed as an open Jackson queueing network.

As indicated in Fig. 4, there are four service queues involved in the content routing and lookup process, including content routing queue, content lookup queue, content retrieve queue, and content response queue. Each queue may introduce the delay (i.e., the sojourn time) to the overall request response time. They serve as the foundations to calculate the expectation of request response time under a chosen scheme. Fig. 5 illustrates the processing models for the three content routing and lookup schemes. Based on the different workflows through different service components, we yield the analytical results on the mean response time for each scheme.

B. System Stability

Before the analysis on the content routing and lookup schemes, we have to ensure the system stability such that all the service queues are finite.

We first consider the stability condition for each surrogate. Suppose the request arrival process at ingress I_i for a particular

TABLE I
NOTATION TABLE

Symbol	Definition
N	The number of surrogate inside CoDaaS
λ_i^c	The mean arrival rate at ingress I_i for content c
μ_{ij}	The mean service rate at ingress I_i for the contents whose designated server is D_j
C_{ij}	The bandwidth capacity of the path from I_i to D_j
μ_t	The mean service rate of content routing
μ_o	The mean service rate at origin server
C_o	The bandwidth capacity at origin server
μ_l	The mean service rate of content lookup
μ_r	The mean service rate of returning content
α	The true cache hit ratio
f_p	The false positive ratio
f_n	The false negative ratio

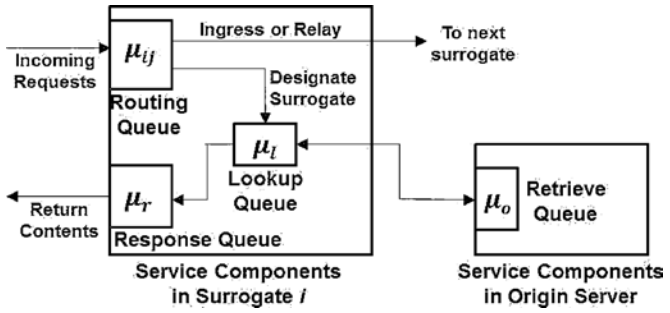


Fig. 4. Service components for content routing and lookup.

content c follows a Poisson process with the mean arrival rate λ_i^c. The ingress server I_i serves these requests with the mean service rate at μ_{ij}, at which they will be further forwarded to their designated surrogate D_j. We also assume the bandwidth capacity of the routing path from surrogate i to j is C_{ij}, $(i \neq j)$. As a result, the system stability condition is given by

$$\sum_{\forall c} \lambda_i^c < \sum_{\forall j} \{min(\mu_{ij}, C_{ij})\}. \tag{2}$$

We also need to consider the stability condition for origin server owned by content provider. In CoDaaS, the surrogate may either be the ingress, relay or the designated server to serve the requests. When surrogate i works as an ingress or a relay, it checks the routing table to determine the next hop for current requested content c. When it works as the designated server, it checks its own storage with the service rate μ_i to determine whether there is a cache hit or miss, denoted as p_c (0 for cache miss and 1 for cache hit). We refer to μ_o as the serving rate at origin server, and C_o as the bandwidth capacity at origin server. As a result, the stability condition for the origin server is given by

$$\sum_{\forall i} \sum_{\forall c} \lambda_i^c (1 - p_c) < min(\mu_o, C_o). \tag{3}$$

C. Response Time for Sequential Scheme

We first investigate the mean response time of sequential scheme as an example to further derive the results for our proposed schemes. Specifically, based on the analysis on the workflow, we first obtain the arrival rate for each queue. Then we can have the mean sojourn time for each queue. Finally, we get the mean response time of the sequential scheme.

1) *Workflow of Sequential Scheme:* Fig. 5(a) shows the workflow of sequential scheme. Each incoming request has to sequentially follow the content routing queue and the content lookup queue no matter there is a cache hit or a cache miss. Moreover, in case of a cache miss, it will take extra time to enter the content retrieve queue to obtain the requested content from origin server. For both conditions, there is a content response queue that the designated surrogates return the requested content to users after all the previous requests are served. As a result, we have

$$\lambda_1 = \lambda_2 = \lambda, \quad \lambda_3 = \beta\lambda_2, \quad \lambda_4 = \lambda_3 + \alpha\lambda_2 = \lambda_2 \tag{4}$$

where $\lambda = \sum_{\forall i} \sum_{\forall c} \lambda_i^c$. For simplicity, we assume the incoming requests are uniformly distributed among all the available surrogates. As a result, there is $\lambda = N \sum_{\forall c} \lambda_i^c$.

2) *Content Routing Queue:* The first queue is the content routing queue, which directs the requests from the ingress to their designated server. We have assumed the external arrival of the requests follows a Poisson process, with an average arrival rate of λ_i^c. Together with this assumption, the service time at surrogate i is assumed to follow exponential distribution with an average service rate of μ_{ij} for those requests whose designated surrogate is server j, since the aggregation of multiple Poisson flows still follows the Poisson process. Therefore, at the content routing queue in each ingress, we have the average traffic load ρ_1 as

$$\rho_1 = \frac{\sum_{\forall c} \lambda_i^c}{\sum_{\forall j} \mu_{ij}}. \tag{5}$$

We further have the average number of jobs at each surrogate in this queue, N_1', as

$$\mathbb{E}[N_1'] = \frac{\rho_1}{1 - \rho_1} = \frac{\sum_{\forall c} \lambda_i^c}{\sum_{\forall j} \mu_{ij} - \sum_{\forall c} \lambda_i^c}. \tag{6}$$

Considering every request on average has to traverse $\lceil log_2 N \rceil$ nodes to reach its designed server, and all the surrogates are identical, we have the expected sojourn time for the content routing queue as

$$\mathbb{E}[N_1] = \lceil log_2 N \rceil \mathbb{E}[N_1']. \tag{7}$$

3) *Content Lookup Queue:* Once the requests have been routed to their designated servers, the content lookup process occurs to determine whether the requested content is cached in the system. Similarly, we assume the service time is exponentially distributed with the average service rate of μ_i at surrogate i. Supposing the hash functions are well designed, we can derive that the contents cached in CoDaaS must be evenly distributed among all the N surrogates. As a result, we have the expected utilization of content lookup queue, ρ_2, as

$$\rho_2 = \frac{\sum_{\forall i} \sum_{\forall c} \lambda_i^c}{N\mu_l} \tag{8}$$

where μ_l is the mean service rate of content lookup.

The average number of jobs N_2 in this queue is

$$\mathbb{E}[N_2] = \frac{\rho_2}{1-\rho_2} = \frac{\sum_{\forall i}\sum_{\forall c}\lambda_i^c}{N(\mu_l - \sum_{\forall i}\sum_{\forall c}\lambda_i^c)}. \quad (9)$$

4) *Content Retrieve Queue:* After the completion of content lookup, the content retrieve process is required at the origin server for the cache miss cases. Let α be the cache hit rate and thus $\beta = 1-\alpha$ be the cache miss rate. Assuming the service time is exponentially distributed with the average rate of μ_o at origin server, we have the average traffic load, ρ_3, as

$$\rho_3 = \frac{\beta\sum_{\forall i}\sum_{\forall c}\lambda_i^c}{\mu_o}. \quad (10)$$

The expected number of jobs N_3 in this queue is as

$$\mathbb{E}[N_3] = \frac{\rho_3}{1-\rho_3} = \frac{\beta\sum_{\forall i}\sum_{\forall c}\lambda_i^c}{\mu_o - \beta\sum_{\forall i}\sum_{\forall c}\lambda_i^c}. \quad (11)$$

5) *Content Response Queue:* In case of cache hit, the designated surrogates i directly returns the requested content to the users. The service time also follows exponential distribution with the average service rate of μ_i'. As a result, we have the average traffic load at content response queue as

$$\rho_4 = \frac{\sum_{\forall c}\lambda_i^c}{\mu_r} \quad (12)$$

where μ_r is the mean service rate of returning content.

The average number of jobs N_4 is as

$$\mathbb{E}[N_4] = \frac{\sum_{\forall c}\lambda_i^c}{\mu_r - \sum_{\forall c}\lambda_i^c}. \quad (13)$$

6) *Mean Response Time:* By using the Jackson network model, we obtain the expected response time R_t of each request spent in the system in sequential scheme, which can be expressed as

$$\begin{aligned}\mathbb{E}[R_t] &= N/\lambda = \sum_{k=1}^{4} N_k/\lambda \quad (14)\\ &= \lceil\log_2 N\rceil(N\mu_t - \lambda)^{-1} + (N\mu_l - N\lambda)^{-1}\\ &\quad +\beta(\mu_o - \beta\lambda)^{-1} + (N\mu_r - \lambda)^{-1}\end{aligned}$$

where $\mu_t = \sum_{\forall j}\mu_{ij}$, indicating the mean service rate of content routing at each surrogate.

D. Response Time for Parallel Scheme

The parallel scheme parallelizes the retrieve operation with the content routing and lookup process as shown in Fig. 5(b). As a result, for a cache miss, the sojourn time introduced by the content routing queue could be saved.

However, the overhead of this operation includes both false positive f_p, false negative f_n and the extra sync messages. Specifically, the cache hit ratio becomes $\alpha' = \alpha + f_p - f_n$ and the cache miss rate is $\beta' = 1-\alpha-f_p+f_n$. In case of false positive, a content retrieve process is still needed to make the response. The detailed analysis on the overhead introduced by BF will be discussed in Section VI.

Based on the analysis on the parallel scheme we have

$$\lambda_2 = \lambda_1 = \alpha'\lambda, \quad \lambda_3 = (\beta' + f_p)\lambda, \quad \lambda_4 = \lambda. \quad (15)$$

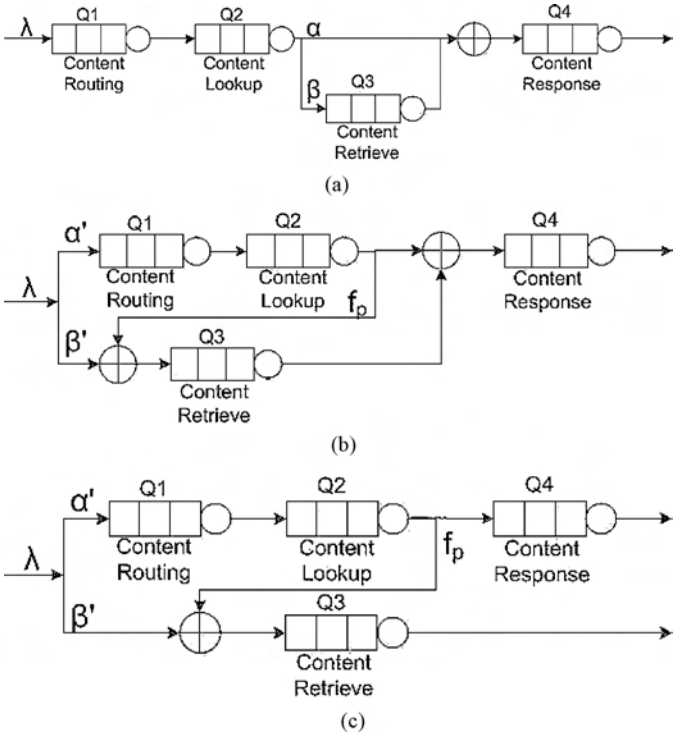


Fig. 5. Processing models for three alternative content routing and lookup schemes in CoDaaS. (a) Sequential scheme. (b) Parallel scheme. (c) Cut-through scheme.

We thus have the mean number of jobs at each queue as

$$\mathbb{E}[N_1] = \lceil log_2 N\rceil\frac{\alpha'\sum_{\forall c}\lambda_i^c}{\sum_{\forall j}\mu_{ij} - \alpha'\sum_{\forall c}\lambda_i^c} \quad (16)$$

$$\mathbb{E}[N_2] = \frac{\alpha'\sum_{\forall i}\sum_{\forall c}\lambda_i^c}{N(\mu_l - \alpha'\sum_{\forall i}\sum_{\forall c}\lambda_i^c)} \quad (17)$$

$$\mathbb{E}[N_3] = \frac{(\beta' + f_p)\sum_{\forall i}\sum_{\forall c}\lambda_i^c}{\mu_o - (\beta' + f_p)\sum_{\forall i}\sum_{\forall c}\lambda_i^c} \quad (18)$$

$$\mathbb{E}[N_4] = \frac{\sum_{\forall c}\lambda_i^c}{\mu_r - \sum_{\forall c}\lambda_i^c}. \quad (19)$$

As a result, by using parallel scheme, the mean response time R_p spent in the system can be expressed as

$$\begin{aligned}\mathbb{E}[R_p] &= \alpha'\lceil\log_2 N\rceil(N\mu_t - \alpha'\lambda)^{-1} \quad (20)\\ &\quad +\alpha'(N\mu_l - N\alpha'\lambda)^{-1} + (N\mu_r - \lambda)^{-1}\\ &\quad +(\beta' + f_p)(\mu_o - (\beta' + f_p)\lambda)^{-1}.\end{aligned}$$

E. Response Time for Cut-through Scheme

The cut-through scheme further cuts the content response time from designated surrogate by redirecting requests to origin server for cache misses as indicated in Fig. 5(c). We have the expected arrival rate at each queue as

$$\lambda_1 = \lambda_2 = \alpha'\lambda, \quad \lambda_3 = (\beta' + f_p)\lambda, \quad \lambda_4 = (\alpha' - f_p)\lambda. \quad (21)$$

In a similar way, we then have the mean number of jobs at content response queue as

$$\mathbb{E}[N_4] = \frac{(\alpha' - f_p)\sum_{\forall c}\lambda_i^c}{\mu_r - (\alpha' - f_p)\sum_{\forall c}\lambda_i^c}. \quad (22)$$

and the average number of jobs at other queues remains the same as parallel scheme.

As a result, by using cut-through scheme, the mean response time R_c spent in the system can be expressed as

$$\begin{aligned}\mathbb{E}[R_c] &= \alpha'\lceil\log_2 N\rceil(N\mu_t-\alpha'\lambda)^{-1} \\ &\quad +\alpha'(N\mu_l-N\alpha'\lambda)^{-1} \\ &\quad +(\beta'+f_p)(\mu_o-(\beta'+f_p)\lambda)^{-1} \\ &\quad +(\alpha'-f_p)(N\mu_r-(\alpha'-f_p)\lambda)^{-1}.\end{aligned} \tag{23}$$

VI. Overhead Analysis

Apart from the performance analysis, a comprehensive work also requires the solid analysis on the potential overheads and limitations. In this section, we investigate the overhead in terms of the false positive/negative, synchronization traffics, and the duplicated message introduced by cut-through scheme.

A. False Positive and False Negative

Two factors may lead to false positive, including the nature of CBF and the inconsistency problem when the cached content has been replaced. Since in CoDaaS, the cached content must be the most popular ones, and the requests follow Zipf law [20], the false positive caused by inconsistency is negligible. As a result, the false positive f_p is roughly equal with f_{bf} as

$$f_p \approx (1-e^{-kn/m})^k. \tag{24}$$

The false negative is mainly introduced by the consistency issues when surrogates synchronize the global BF. Specifically, during each synchronization, if any particular content, which has not been cached already, is requested by more than twice, and the requests are not toward the same surrogate, then false negative occurs.

We first assume the arrival requests follow Zipf law as

$$F_M(k) = \Omega k^{-\gamma} \tag{25}$$

where $F_M(k)$ is the request frequency of content of rank k, M is the total number of contents, γ is the shape parameter of Zipf distribution, and Ω is a constant given by

$$\Omega = (\sum_{s=1}^{M} s^{-\gamma})^{-1}. \tag{26}$$

In this paper we consider two synchronization methods, including periodic and event-driven styles. For the periodic synchronization, let t be the synchronization interval, and λ be the average arrival rate. Then λt is the total request number during this interval. Suppose after each synchronization, the system has already cached the most C popular contents. Thus we first have the expected number C_m that are not toward the top C contents during each synchronization interval t as

$$\mathbb{E}[C_m] = \lambda t(1-\Omega\sum_{k=1}^{C} k^{-\gamma}). \tag{27}$$

During each synchronization interval, we assume all previously requested contents will remain in the cache without replacement for simplicity. In this case, for the $(i+1)$'st request among C_m, the probability P_c that the requested content has been already cached is given by

$$P_c(i) = \sum_{k=1}^{M-C} F_M(k)(1-(1-F_M(k))^i). \tag{28}$$

It has been proved [20] that this probability $P_c(i)$, for $1 << i << M-C$ and $0<\gamma<1$, can be approximated by

$$P_c(i) \approx \Omega'(1-\gamma)^{-1}(i\Omega')^{(\gamma^{-1}-1)} \tag{29}$$

where $\Omega' = (\sum_{s=1}^{M-C} s^{-\gamma})^{-1}$. Note, when $i > M-C$, this estimation may not be accurate.

As a result, we have the expected false negative rate as

$$\begin{aligned}\mathbb{E}[f_n] &= \frac{N-1}{N\lambda t}\sum_{i=1}^{C_m} P_c(i) \\ &\approx \frac{N-1}{N\lambda t}\sum_{i=1}^{C_m}\Omega'(1-\gamma)^{-1}(i\Omega')^{(\gamma^{-1}-1)}.\end{aligned} \tag{30}$$

For the event-driven synchronization, let n be the number of requests which triggers each synchronization. As a result, to obtain the mean false negative rate under this scheme, we could simply substitute λt by n in (27) and (30).

B. Synchronization Messages

Apart from the congestion that may occur at the origin server, the synchronization messages may also introduce traffic jams inside the CoDaaS system. For each surrogate i, its outcoming traffic consists of synchronization messages, content transferring and request forwarding. The overall traffic must not exceed its bandwidth capacity to avoid the congestion.

For the parallel scheme, the content transfer occurs for both cache misses and hits. For cache misses, this traffic will be doubled. Let λ_s be the synchronization rate, S_f be the size of the CCBF, and S_r be the size of content request. Let η be the corresponding congestion factor [21], and S_c be the size of requested content. The following inequality should be satisfied to avoid congestion

$$\lambda_s\mathbb{E}[S_f]+((2-\alpha-f_p+f_n)\mathbb{E}[S_c]+\mathbb{E}[S_r])\sum_{\forall c}\lambda_i^c < \eta\sum_{\forall j}C_{ij}. \tag{31}$$

For cut-through scheme, the content transfer from surrogate to customers only happens for cache hits. Thus, the congestion free condition for cut-through scheme is given by

$$\lambda_s\mathbb{E}[S_f]+((\alpha+f_p-f_n)\mathbb{E}[S_c]+\mathbb{E}[S_r])\sum_{\forall c}\lambda_i^c < \eta\sum_{\forall j}C_{ij}. \tag{32}$$

C. Duplicated Message Introduced by Cut-through Scheme

It can be easy to find that the cut-through scheme doubles the traffic at origin server in case of cache miss. Therefore, it may cause congestion much more easily than other two schemes. The congestion will further lead to severe service

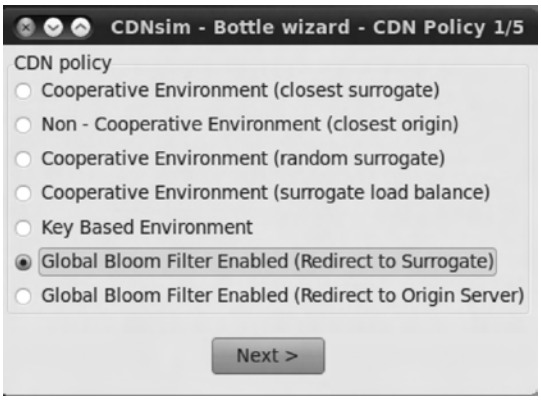


Fig. 6. Snapshot of CDNsim.

delay. To avoid the congestion at the origin server, the following inequality must be satisfied:

$$2(1-\alpha-f_p+f_n)\mathbb{E}[S_c]\sum_{\forall i}\sum_{\forall c}\lambda_i^c < \eta C_o. \tag{33}$$

Therefore, we obtain the threshold of cache hit rate α and the size of requested content S_c from this inequality for congestion avoidance as

$$\alpha > 1-f_p+f_n-\frac{\eta C_o}{2\mathbb{E}[S_c]\sum_{\forall i}\sum_{\forall c}\lambda_i^c}, \tag{34}$$

$$\mathbb{E}[S_c] < \frac{\eta C_o}{2(1-\alpha-f_p+f_n)\sum_{\forall i}\sum_{\forall c}\lambda_i^c}. \tag{35}$$

VII. Simulation and Results

This section first presents the details on simulation methodologies. Then false positive/negative is investigated to provide suitable settings for the following experiment on mean response time, where both analytical and numerical results of different schemes are obtained. Finally, the impact of synchronization frequency on mean response time is discussed.

A. Simulation Methodology and Setup

1) *Simulation Tool:* We use CDNsim [19] to build our simulation environment. Specifically, CDNsim is a discrete event simulation tool based on OMNeT++ library particularly designed for CDNs. We modify some basic modules to fit our implementations and add the three proposed content routing and lookup policies into this tool. Fig. 6 demonstrates a snapshot of our simulation tool.

2) *Simulation Setting:* We simulate $N = 50$ homogenous VMs as geographically distributed surrogates to construct the media cloud inside the CoDaaS. All the surrogates are coordinated by a distributed hash table (DHT) based ring using Chord protocol [22]. The modulo function is used to hash the identification of both contents and surrogates. The cache capacity of each surrogate is set at 2% of the total size of the contents stored in origin server. The least recently used (LRU) strategy is employed as the cache replacement policy. We use GT-ITM [23] to generate a real Internet topology model, transit-stub model with 1008 routers dispersed at different areas. There is only one origin server, which hosts 50 000 unique content objects, which are mainly comments and low-resolution photos, with the total size of approximately 5GB retrieved from a social website. Fig. 7(a) presents the cumulated distribution density (CDF) of those contents. The service capacity is 600 $reqs/s$ for each surrogate, and 1200 $reqs/s$ for the origin server. We assume the size of each request occupies 60 Bytes (i.e., $S_r = 60$ $Bytes$). Besides, we define the link capacity as $C_o = 100$ MB/s for origin server, and $\sum_{\forall j} C_{ij} = 100$ MB/s for surrogate i.

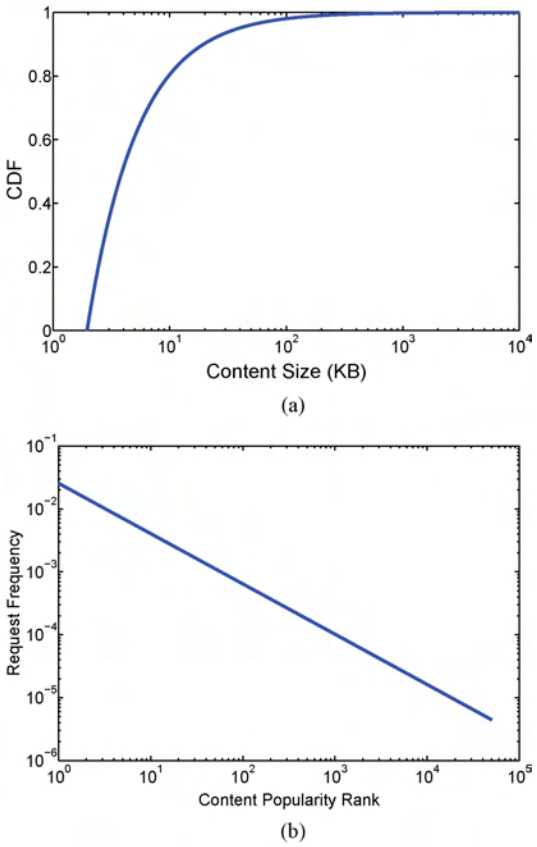


Fig. 7. Distribution of (a) content size and (b) popularity.

3) *Request Traffic Pattern Setting:* We generate requests from 100 client groups with the mean interval time at 0.01 second (i.e., $\sum_{\forall i}\sum_{\forall c}\lambda_i^c = 100$ $reqs/s$). The distribution of the interval time of all the request follows exponential distribution, and the distribution of the requested content follows Zipf law to make the access pattern much closer to realistic ones. Fig. 7(b) plots the frequency of requested contents in terms of their popularity ranks, where the parameter $\gamma \approx 0.79$. As a result, with this setting, the stability of our CDN system can be ensured by satisfying both (2) and (3).

4) *Bloom Filter Setting:* The hash functions of the BF are built by dividing a 128-bit MD5 signature [24] for each URL into four 32-bit words, and the modulus of each word by the number of counters are the hash values. Each surrogate on average takes charge of caching 1000 distinct objects, so it needs a counter BF with maximum volume at 20KB (i.e. 40 000 counters) to keep the false positive rate below 1% for Zipf distributed enquires. Therefore, the global BF requires 1MB spaces in total. In our simulation, we further compress

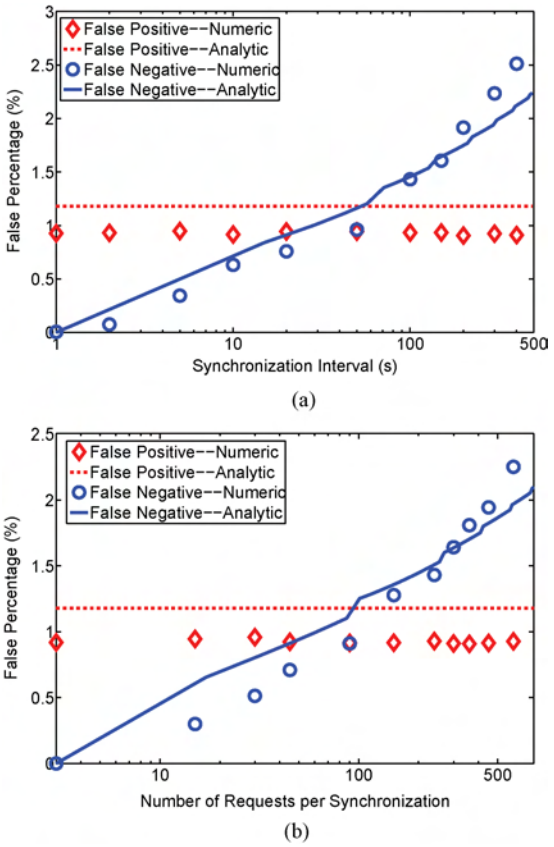


Fig. 8. False positive/negative rate versus sync frequency. (a) Periodic scheme. (b) Event-driven scheme.

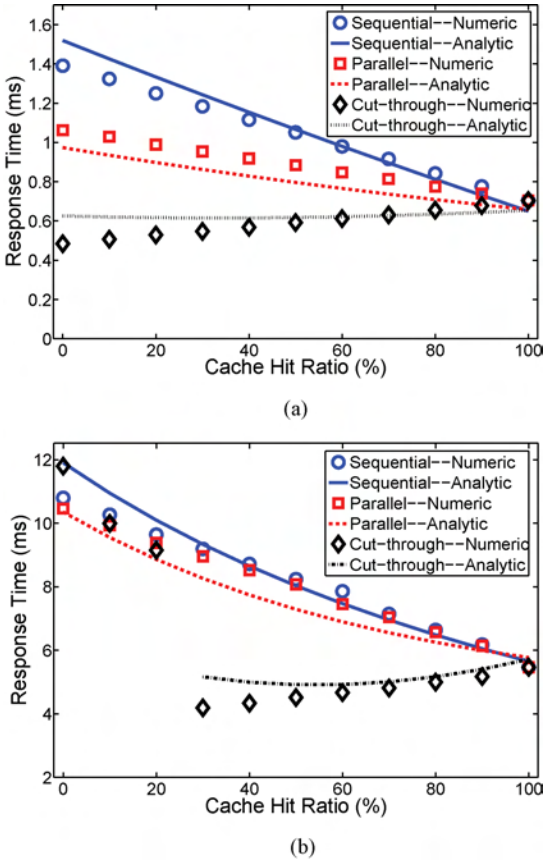


Fig. 9. User response time versus cache hit rate under two scenarios. (a) Popularity of content is inversely proportional to its size (light traffic). (b) Popularity of content is proportional to its size (heavy traffic).

this global counting BF by using ML-CCBF [17] method to reduce the transmission size. Besides, the synchronization messages are also compressed by delta compression algorithm to save the network bandwidth.

B. False Positive and False Negative

Fig. 8 shows the relationship between synchronization frequency and false positives/negatives under two synchronization mechanisms (i.e., periodic and event-driven synchronization) both analytically and numerically. The analytic results are derived from (24) and (30), while the numeric results are generated from our simulations. It can be seen that the analytic results roughly present the same style as the numeric ones. In addition, the results also indicate that the two synchronization schemes follow almost the same trend. This can be attributed to the fact that for a given incoming pattern, the expected inter-arrival time is fixed, as we discussed in Section VI.

As the interval increases, the false negative percentage increases accordingly, while the false positive rate maintains at a certain level since it is independent with the number of incoming requests. The relationship between the frequency of synchronization and the false positive/negative is roughly linear in a log-log graph. When the synchronization frequency is extremely high (i.e., one synchronization after every single request is served), there is no false negative at all, but this setting could generate intensive synchronization traffic. When the frequency is higher than once per 200 seconds for periodic scheme or once per 400 requests for event-driven scheme, the false negative can be controlled below 2 %. In this case the traffic load could also be kept at a relatively low level. In the next experiment, we will use this setting to investigate the mean response time of each scheme.

C. Improvement on Mean User Response Time

We simulate two scenarios to test our system. One is that the popularity of an object is inversely proportional to its size, that is, the smallest content is most popular. The other is that the largest content has the highest popularity. Hence, the average size of requested content S_c for these two scenarios is around 32KB and 170KB respectively, to simulate different level of traffic load on data plane. A set of cache hit rates ranging from 0% to 100% are applied by initializing the global BF and the cached content in each surrogate. As a result, this simulation aims to test the mean response time of different content routing and lookup scheme in function of the cache hit rate. The simulation lasts for 1000 seconds.

Fig. 9 shows the performance in terms of the mean response time of the three content routing and lookup schemes under light traffic loads, both numerically and analytically. The numeric results are generated from the simulations, and the analytic results are derived from (14), (20), and (23).

Fig. 9(a) presents the results of the light traffic scenario. The analytic parameters are set as $\lambda = 100\ req/s$, $\mu_t = 300\ req/s$, $\mu_l = 500\ req/s$, $\mu_o = 1200\ req/s$, and $\mu_r = 60\ req/s$ according

to our simulation settings. The average response time of cut-through scheme is the lowest, while the traditional scheme costs the highest delay, among all the three mechanisms. When there is no cache hit, the gap between our proposed schemes and the traditional one is the largest for all the cases. The parallel scheme and the cut-through scheme are able to reduce the mean response time by 23.6% and 65.2% compared with that required by the sequential one according to the results. As the cache hit ratio goes up, the performances of the three schemes get closer, because the workflows are the same for cache hit cases. Finally, when all the requests are cache hit, the results from the three policies arrive at roughly the same point.

An interesting observation in Fig. 9(a) is that the mean response time of both traditional sequential scheme and parallel scheme decreases gradually, while the time required by cut-through scheme increase, as the cache hit ratio raises. It indicates that as long as there is no traffic congestion (i.e., (35) is satisfied), it is advantageous to redirect the missed requests to origin server to minimize the response time.

Fig. 9(b) shows the performances of the high traffic load scenario. The parameters are set as μ_t = 100 req/s, μ_l = 500 req/s, μ_o = 260 req/s, and μ_r = 8 req/s, due to the larger size of requested contents compared with the first scenario. In this case, the parallel scheme still needs less response time than that of the sequential scheme for all the cases, while the cut-through scheme suffers from the congestion when the true cache hit rate is lower than 30%. However, cut-through scheme performs most efficiently again when the cache hit rate passes the congestion threshold as derived from (34). It may motivate us to adopt an adaptive algorithm, which adopts the cut-through scheme when there is no congestion, and dynamically changes into the parallel scheme when the cache miss rate exceeds the threshold.

D. Impact of Synchronization Traffic

Fig. 10 shows the impact of synchronization traffic on the mean response time for our proposed parallel and cut-through schemes. We experiment with a set of synchronization intervals based on both periodic and event-driven schemes to generate different levels of traffic. This simulation is under the condition that the cache hit rate is 40%.

Fig. 10(a) shows how the frequency of periodic scheme affects the response time. The response time is constrained by the congestion inside the CoDaaS when the frequency is high (i.e., higher than once per second). In contrast, when the frequency is extremely low (i.e., lower than once per 1000 seconds), the incurred false negative causes more faking cache misses. This further introduces congestion at origin server, and thus increases the response time.

Fig. 10(b) shows how the frequency of event-driven scheme affects the response time. Specifically, the response time is much longer when the frequency is higher than once per five requests for both schemes. The reason can be attributed to the traffic jam inside CoDaaS as discussed in (31) and (32). As the synchronization interval continues to increase, the synchronization traffic becomes lighter, but on the other hand, the false negative increases. Such combined effects keep the response time at roughly a certain level.

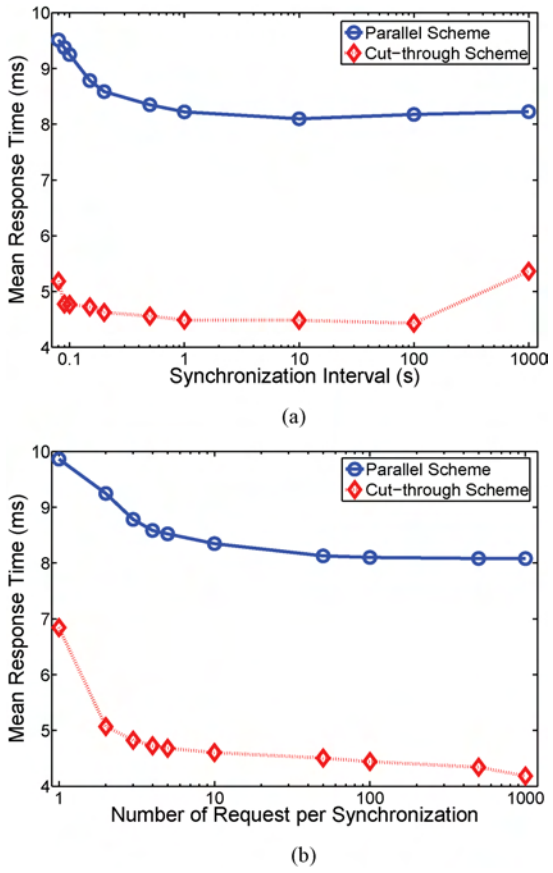


Fig. 10. Sync frequency versus user response time. (a) Periodic scheme. (b) Event-driven scheme.

VIII. Conclusion and Future Works

In this paper, we proposed the parallel and the cut-through content routing and lookup schemes by using a global CCBF. Mathematical models based on Jaskson network model were presented to analyze both the user perceived response time, and the overheads of importing this global BF, including false positive, false negative, synchronization cost, and possible congestions. Simulations were used to prove the performance of those metrics. The numerical results matched well with our analytic models, indicating our proposed schemes can achieve a saving up to 65.2% on mean response time.

Our future work will aim to develop a congestion-aware adaptive algorithm to further improve the efficiency of our proposed content routing and lookup scheme for different traffic loads. We also plan to consider conditions when the virtualization technologies in CoDaaS are utilized, so that the service rate and storage capacity can be dynamically adjusted.

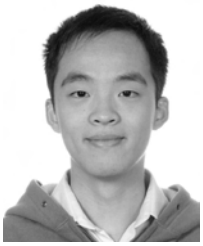
References

[1] P. Verna. (2009). “A spotlight on UGC participants,” Tech. Rep. [Online]. Available: http://www.emarketer.com/Article.aspx?R=1006914

[2] K. Chang, C. Chen, J. Chen, and H. Chao, “Challenges to next generation services in IP multimedia subsystem,” *J. Inform. Process. Syst.*, vol. 6, no. 2, pp. 129–146, 2010.

[3] Y. Wen, G. Shi, and G. Wang, “Designing an inter-cloud messaging protocol for content distribution as a service (CoDaaS) over future Internet,” in *Proc. ACM CFI*, Jun. 2011, pp. 91–93.

[4] Y. Jin, Y. Wen, G. Shi, G. Wang, and A. Vasilakos, "CoDaaS: An experimental cloud-centric content delivery platform for user-generated contents," in *Proc. IEEE ICNC*, Jan. 2012, pp. 934–938.
[5] J. Chen, S.-H. Chan, and V. Li, "Multipath routing for video delivery over bandwidth-limited networks," *IEEE J. Selected Areas Commun.*, vol. 22, no. 10, pp. 1920–1932, Dec. 2004.
[6] D. Karger, A. Sherman, A. Berkheimer, B. Bogstad, R. Dhanidina, K. Iwamoto, B. Kim, L. Matkins, and Y. Yerushalmi, "Web caching with consistent hashing," *Comput. Netw.*, vol. 31, nos. 11–16, pp. 1203–1213, May 1999.
[7] J. Ni and D. Tsang, "Large scale cooperative caching and application-level multicast in multimedia content delivery networks," *IEEE Commun. Mag.*, vol. 43, no. 5, pp. 98–105, May 2005.
[8] M. Pathan, C. Vecchiola, and R. Buyya, "Load and proximity aware request-redirection for dynamic load distribution in peering CDNs," in *On The Move to Meaningful Internet Systems: OTM 2008*. Berlin, Germany: Springer-Verlag, 2008, pp. 62–81.
[9] M. Harren, J. Hellerstein, R. Huebsch, B. Loo, S. Shenker, and I. Stoica, "Complex queries in dht-based peer-to-peer networks," in *Proc. IPTPS*, 2002, pp. 242–259.
[10] J. Chang, H. Chao, J. Chen, and C. Lai, "An efficient service discovery system for dual-stack cloud file service," *IEEE Syst. J.*, vol. 6, no. 4, pp. 584–592, 2012.
[11] B. H. Bloom, "Space/time tradeoffs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.
[12] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet Math.*, vol. 1, no. 4, pp. 485–509, May 2004.
[13] A. Rousskov and D. Wessels, "Cache digests," *Comput. Netw. ISDN Syst.*, vol. 30, pp. 2155–2168, Nov. 1998.
[14] Squid. (2012). *Optimising Web Delivery* [Online]. Available: http://www.squid-cache.org/,Tech.Rep.
[15] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: A scalable wide area web cache sharing protocol," *IEEE/ACM Trans. Netw.*, vol. 8, no. 3, pp. 281–293, Jun. 2000.
[16] M. Lee, K. Cho, K. Park, T. Kwon, and Y. Choi, "Scan: Scalable content routing for content-aware networking," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2011, pp. 1–5.
[17] D. Ficara, S. Giordano, G. Procissi, and F. Vitucci, "Multilayer compressed counting bloom filters," in *Proc. IEEE INFOCOM*, 2008, pp. 311–315.
[18] J. R. Jackson, "Jobshop-like queueing systems," *Manage. Sci.*, vol. 10, no. 1, pp. 131–142, 1963.
[19] K. Stamos, G. Pallis, A. Vakali, D. Katsaros, A. Sidiropoulos, and Y. Manolopoulos, "CDNsim: A simulation tool for content distribution networks," *ACM Trans. Model.Comput.Simulat.*, vol. 20, pp. 1–40, Apr. 2010.
[20] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proc. IEEE INFOCOM*, 1999, pp. 126–134.
[21] R. Banner and A. Orda, "Multipath routing algorithms for congestion minimization," *IEEE/ACM Trans. Netw.*, vol. 15, no. 2, pp. 413–424, Apr. 2007.
[22] I. Stoica, R. Morris, D. Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for Internet applications," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 17–32, 2003.
[23] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an Internet work," in *Proc. IEEE INFOCOM*, Mar. 1996, pp. 594–602.
[24] R. L. Rivest, "The MD5 message digest algorithm," *Request for Comments (RFC) 1321*, Apr. 1992.

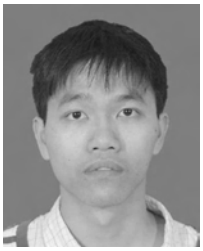


Yichao Jin received the B.S. and the M.S. degrees from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2008 and 2011, respectively. He is currently pursuing the Ph.D. degree at the School of Computing Engineering, Nanyang Technological University, Singapore.

His current research interests include cloud computing and content delivery networks.

Yonggang Wen (M'08) received the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Boston, MA, USA, in 2008.

Currently, he is an Assistant Professor at the School of Computer Engineering, Nanyang Technological University, Singapore. Prior to his present position, he has held research and development positions in networking companies in the USA, including Cisco and Lucent. His current research interests include cloud computing, content networking, and green networks.



Weiwen Zhang (S'12) received the Bachelor's degree in software engineering and Master's degree in computer science from the South China University of Technology, Guangzhou, China, in 2008 and 2011, respectively. Currently, he is pursuing the Ph.D. degree at the School of Computer Engineering, Nanyang Technological University, Singapore.

His current research interests include cloud computing and mobile computing.