# Temporal Acceleration for Cloud-CDN-Fog-Edge Hierarchy by Leveraging Proximal Object Replicas

Thepparit Banditwattanawong and Masawee Masdisornchote

***Abstract*—A classical requirement to reduce network latency drives paradigm shifts from cloud computing together with content delivery network to fog computing, and edge computing. These paradigms practically co-exist in a multi-tier structure. This paper presents novel approaches for accelerating the cloud-service response times of a cloud-CDN-fog-edge layered structure by means of hierarchical cloud caches and fine-grained replication. We also propose a unified cost-performance model that enables both temporal and monetary cost evaluation. We leverage the model to quantify the temporal effectiveness of the approaches in comparison with a classical LRU one. Results show that, when deploying in conjunction with the fine-grained replication, cloud caches significantly improve the overall response times up to 23.85%. A cloud layer is found to contribute most to the response times. Regardless of selected approaches, another finding is that the response times are the quadratical functions of the number of edge devices.**

***Index Terms*—cloud optimization, fog node, edge device, partial replication, browser cache**

## I. Introduction

IN the simplest use case of cloud computing services, edge devices including smart/IoT end devices are programmed to access origin servers on clouds. Since the cloud services are highly network dependent, a rudimentary problem cloud service consumers encounter is delayed response times. Content delivery network (CDN) [1], fog computing [2], and edge computing [3] are actively researched to mitigate user perceived latencies by placing data stores and computation as near as the consumers. Consequently, production use cases seriously employ the CDN, fog computing, and edge computing to minimize *cloud-service response costs* from end user perspective. The response costs are generic and can be specialized to either cloud-service response times or cloud-data-out monetary charges. The focus of this paper is the former cost form.

The cloud-service response times can be significantly improved by means of caching. Nowadays, the edge devices ubiquitously come with HTTP client capabilities, which include browser caches [4]. The notion of caches also recently appear in fog nodes [5]. All caching mechanisms of edge devices, fog nodes, and CDNs are however so classical that only an object hit rate (OHR) is an optimization key. Our experience [6] suggests that the best OHRs need not lead to the lowest costs. In other words, we found in [6] that the classical caches failed to optimally address the fundamental problems of cloud service accesses: the expensive and slow downloading of big objects such as video streams, disk images, electronic documents, high-resolution multimedia contents, and so on. Alternative to the classical caches is *cloud cache* [6], which is specifically optimized for both temporal and monetary cost saving.

The contributions of this paper are twofold. The paper firstly proposes a simple but sophisticated method to improve cloud-service response times in cloud-CDN-fog-edge layered environments by using a hierarchical cloud-cache architecture equipped with fine-grained replication. The architecture is subsequently evaluated by leveraging our novel unified-cost-performance model in comparison with the classical caches.

Manuscript received November 1, 2019.

T. Banditwattanawong is with the Department of Computer Science, Faculty of Science, Kasetsart University, Jatujak, Bangkok, 10900 Thailand (e-mail: thepparit.b@ku.th).

M. Masdisornchote is with the School of Information Technology, Sripatum University, Jatujak, Bangkok, 10900 Thailand (e-mail: masawee.ma@spu.ac.th).

## II. Related Work

### A. Content Delivery Network

CDNs [1], [7] improve web scalability via the network of *surrogate servers* (aka edge servers and cache servers) that offloads an origin server by delivering objects on its behalf. The surrogate servers are reverse caching proxies strategically placed across a CDN provider's distributed data centers, Internet exchange points (IXPs), and points of presence (PoPs) to improve user-perceived round-trip times. Similar to the traditional CDNs, cloud CDNs [8] have surrogate servers located across globally-distributed cloud-data centers and PoPs. When an HTTP request to a target cloud server from a client is sent to a CDN due to both the client's and a client-side proxy's cache misses, the CDN's request routing infrastructure (RRI) maps the request to the topologically closest surrogate server [9]. If a cache miss occurs and the CDN is noncooperative of its kind [10], such a surrogate will directly contact the target cloud server. Otherwise the cooperative surrogate will use ICP protocol [11], [12] to contact its sibling surrogate [13]. If all siblings in the same deployment [14] (aka edge cluster [15]) yield cache misses, the surrogate will forward the request to the cloud server [7], [13]. Nevertheless, both traditional and cloud CDNs focused on OHR by employing LRU variants [9], [16], [17].

Among CDN-performance modeling efforts, [18] analyzed mesh cache-server performance through both cache replacement and traffic models. A CDN utility in conjunction with surrogate server utilization were used in [19] as a CDN performance metric. [20] utilized a deep recurrent neural

network to find out a reach rate as the key performance metric of CDN.

### B. Fog Computing

Officially defined by [2], fog computing is a layered model enabling low-latency accesses to cloud computing resources by decentralizing storage, data processing, and computing service into *fog nodes*. The fog nodes (aka cloudlets) reside between smart/IoT edge devices and centralized cloud services. The fog nodes are either physical or virtual gateways or servers and either isolated or federated that provide data management and communication services. Like CDN, fog computing is not a mandatory layer to support the interaction between the cloud services and the edge devices.

Caching was increasingly exploited in fog [5], [21], [22]. They however aimed for OHR through either a proprietary cache eviction policy or a new peer-to-peer cache selection scheme.

As for fog performance modeling, [23] modeled average round-trip time and energy consumption. A service latency model in [24] was based on transmission and processing latencies to contrast fog computing and cloud computing suitability in IoT context.

### C. Edge Computing

Edge computing is a peripheral layer encompassing network-accessible edge devices to provide local computing to individual users [2]. Edge devices such as smart phones, smart watches, smart tablets, and various IoT end devices are capable of HTTP-based accesses to distant cloud services via web browser engines. The web browsers, operating in nonprivate mode, conventionally utilize two main types of small memory caches and disk caches [4]. The memory caches ignore almost all HTTP cache headers and stores objects only during session lifetimes. On the other hand, the disk caches completely obey HTTP cache protocol and are thus so persistent that allow object reuses across multiple sessions.

The web browsers' disk caches promote OHRs by means of either LRU or a proprietary algorithm based on object reuse and age [25], [26].

### D. Cloud Cache

Cloud cache [27] is a client-side HTTP cache establishing on a fact that small objects become no longer costly to download in cloud environments wherein big object population keeps increasing. Therefore, the cloud cache is designed to optimize cost saving ratio (*CSR*) rather than OHR by favoring big objects to persist in the cache. Cloud caches employ either of the following two cache replacement policies. *CLOUD* [6] is a profit-function based policy supporting federated clouds with nonuniform data-out monetary charges. The other policy, *SMFD* [28], [29], is a-priori request-distance based policy that universally attains near-optimal CSRs and OHRs at the same time. The cloud caches' admission controls usually follow a compulsorily-caching admission scheme realized by *CLOUD* or *SMFD*. A selectively-caching admission scheme is also allowed via *SMFD**, the variant of *SMFD* allowing optional eviction. The cloud caches' coherence simply relies on a well-established HTTP cache coherence protocol [30]. Similar to hierarchical web caches [31], cloud caches can be layered to match the hierarchical nature of the Internet consisting of ISP-level and enterprise-level shared cloud caches as well as web browser-level cloud caches [6].

### E. Web Cache Architecture Modeling

In [32], [33], not only the network but also requested document models of hierarchical and distributed web caches were presented. [34] modeled a hierarchical web-cache network by using probabilistic flowcharts. All of these models were however aware of neither local web-browser caches nor application replication.

## III. Accelerated Architecture

A production use case wherein edge devices access remote cloud services with the intermediate assistance of both CDN and fog node in a hierarchical tree topology is illustrated in Fig 1. This cloud-CDN-fog-edge architecture allows us to naturally overlay a cloud cache hierarchy to improve the cloud service's response times. The architecture is described now.

When an edge device dispatches a data object request (realized by an HTTP GET method, etc.) to a cloud origin server where target data is provided, the request is first checked in the edge device's web-browser cloud cache. If a cache miss occurs, the request will be redirected to a nearby fog node's cloud cache. If the fog-node cloud cache results in a cache miss, the request will be re-routed to a cloud-cache based CDN. Any cache miss at the CDN will result in the request routed to the cloud origin server. A reply from the origin server will be returned to the edge device via the CDN and the fog node, respectively, and stored in the cloud caches of the CDN, the fog node, and the edge devices.

On the other hand, when an edge device dispatches an application object request (in the form of an HTTP POST method, etc.) to the cloud service, the request will be

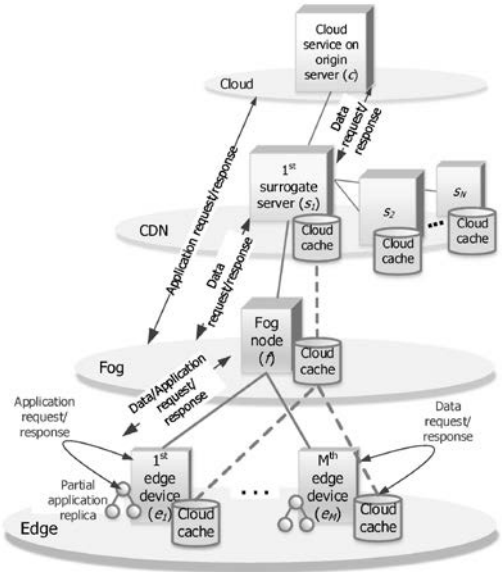


Fig. 1. Cloud cache hierarchy along cloud-CDN-fog-edge architecture.

processed within the edge device itself if fine-grained replication is locally deployed and the requested service's object replica is locally available. Basically, the fine-grained replication (FGR) incrementally and partially replicate a Java server application (implementing the cloud service) for local execution on Java-based devices (e.g., Android). To achieve the FGR, SOOM middleware [35] that enables partial and incremental replication and consistency maintenance is required. In Fig 1, fine-grained replicas are present in all edge devices. If the request cannot be served by the replicas, the request will be routed to a fog node. If the fog node offers no cloud service function that serves the service request, the request will be further routed to the cloud service bypassing CDN. Any response from the cloud service will be returned through the CDN then the fog node to the edge device.

## IV. Unified Cost Performance Model

This section analyzes the generic-cost-performance model of the accelerated architecture in Fig. 1. The resulting model is applicable to both temporal and monetary costs incurred by any number of edge devices making requests to a cloud service through a single isolated fog node and any number of CDN surrogates. According to the architecture, the expected cloud-service response-cost ($\mathbb{E}_c$) of a data- and/or application-request aggregation ($\sigma$) generated by edge devices within a time unit is defined as

$$\begin{aligned}\mathbb{E}_c &= C_e P_e \lambda_e \\ &+ C_{ef}(1-P_e)\lambda_{ef} \\ &+ P_f(C_f\lambda_{ef} + C_{fe}\lambda_{fe}) \\ &+ C_{fs}(1-P_f)\lambda_{fs} \\ &+ P_s\left(C_s\lambda_{fs} + (C_{sf}+C_{fe})\lambda_{se}\right) \\ &+ C_{sc}(1-P_s)\lambda_{sc} \\ &+ P_c\left(C_c\lambda_{sc} + (C_{cs}+C_{sf}+C_{fe})\lambda_{ce}\right)\end{aligned} \quad (1)$$

where $P_e$ is a probability $\sigma$ will be purely served within the edge devices, $C_e$ is a cost per request the edge devices locally respond $\sigma$, $\lambda_e$ is a rate $\sigma$ is generated at the edge devices, $C_{ef}$ is a transfer cost per request from the edge layer to an isolated fog node, $\lambda_{ef}$ is the rate of $\sigma$ subset falling back from the edge layer to the fog node because requested objects are unavailable in the edge devices (i.e., cache misses), $P_f$ is a probability the $\sigma$ subset (i.e., fallback requests) will be responded at the fog node, $C_f$ is a response cost per request at the fog node, $C_{fe}$ is a cost a response is sent from or forwarded by the fog node to the edge, $\lambda_{fe}$ is the rate of responses to the edge by the fog node, $C_{fs}$ is a transfer cost per request from the fog node to a CDN surrogate, $\lambda_{fs}$ is the rate of requests falling back from the fog node to the CDN surrogate, $P_s$ is a probability the fallback requests will be responded at the CDN, $C_s$ is a response cost per request at the CDN, $C_{sf}$ is a cost a response is sent from or forwarded by the CDN to the fog node and the edge respectively, $\lambda_{se}$ is the rate of responses by the CDN toward the edge, $C_{sc}$ is a transfer cost per request from the CDN to a cloud origin server, $\lambda_{sc}$ is the rate of requests falling back from the CDN to the origin server, $P_c$ is a probability the fallback requests will be eventually processed by the origin server and equals 1 if the server is available when the requests arrive, $C_c$ is a cost the origin server responds each request, $C_{cs}$ is a cost a response is sent back from the origin server to the edge via the CDN and the fog, and $\lambda_{ce}$ is the rate of responses by the origin server.

Remark that processing costs to evaluate cache misses at every cache level, the costs of both response forwarding decision and missing object caching at the CDN and the fog node, and the costs of both response displaying and missing object caching at the edge are neither significant nor in-line in cloud-service response times, thus omitted. Moreover, since the response cost is to be measured within a single time unit, an assumption that cached objects do not expire meanwhile applies. Such an assumption is supported by [36] : within much longer time frame 3 weeks, 95% of requests originating at mobile devices can be served by an unconditional HTTP GET method. In other words, objects requested by the edge mostly last long.

Since the edge devices can make requests to both data and application objects (hopefully served by cloud caches and fine-grained replicas, respectively) to the cloud service, the cost components of (1) are derived one by one.

An edge response cost incurred within the time unit equals

$$C_e P_e \lambda_e = \sum_{i=1}^{M} \max\left(C_{e_i}^{(d)} P_{e_i}^{(d)} \lambda_{e_i}^{(d)}, C_{e_i}^{(a)} P_{e_i}^{(a)} \lambda_{e_i}^{(a)}\right) \quad (2)$$

where notations $(d)$ and $(a)$ distinguish data object requests and application object requests and the total number of edge devices is $M$. $P_{e_i}^{(d)}$ and $P_{e_i}^{(a)}$ are a data OHR and an application OHR, respectively, at $e_i$. As both data and application object requests are executed concurrently by multiprocessing computer devices, the max function is utilized.

An edge-to-fog request transmission cost is

$$C_{ef}(1 - P_e)\lambda_{ef} = \sum_{i=1}^{M}\left(C_{e_i f}^{(d)}(1 - P_{e_i}^{(d)})\lambda_{e_i}^{(d)} + C_{e_i f}^{(a)}(1 - P_{e_i}^{(a)})\lambda_{e_i}^{(a)}\right). \quad (3)$$

Note that costs arising at the edge devices and between the fog node and the edge devices are serial in the default model (i.e., an overall cost paid by all $M$) to project the entire-system response cost of $M$ devices. These serial costs can alternatively be summarized into an average, min, or max value to reflect a response cost an individual edge device undergoes.

At the isolated fog node, a fog response cost is

$$\begin{aligned}P_f(C_f\lambda_{ef} + C_{fe}\lambda_{fe}) = \max&\begin{pmatrix} C_f^{(d)} P_f^{(d)} \sum_{i=1}^{M}(1 - P_{e_i}^{(d)})\lambda_{e_i}^{(d)}, \\ C_f^{(a)} P_f^{(a)} \sum_{i=1}^{M}(1 - P_{e_i}^{(a)})\lambda_{e_i}^{(a)} \end{pmatrix} \\ &+ P_f^{(d)} \sum_{i=1}^{M} C_{fe_i}^{(d)}(1 - P_{e_i}^{(d)})\lambda_{e_i}^{(d)}) \\ &+ P_f^{(a)} \sum_{i=1}^{M} C_{fe_i}^{(a)}(1 - P_{e_i}^{(a)})\lambda_{e_i}^{(a)})\end{aligned} \quad (4)$$

where $P_f^{(d)}$ is a data OHR and $P_f^{(a)}$ is a probability to find the requested functional objects of the cloud service running within the fog node.

A fog-to-CDN request-transfer cost is

$$C_{fs}(1 - P_f)\lambda_{fs} = C_{fs}^{(d)}(1 - P_f^{(d)}) \sum_{i=1}^{M}\left(1 - P_{e_i}^{(d)}\right)\lambda_{e_i}^{(d)} + C_{fs}^{(a)}(1 - P_f^{(a)}) \sum_{i=1}^{M}\left(1 - P_{e_i}^{(a)}\right)\lambda_{e_i}^{(a)}. \quad (5)$$

Let $N$ be the number of cooperative surrogate servers, a CDN response cost is

$$P_s(C_s\lambda_{fs} + (C_{sf} + C_{fe})\lambda_{se}) = \sum_{j=1}^{N} P_{s_j}^{(d)} C_{s_j}^{(d)}\left(1 - P_f^{(d)}\right) \sum_{i=1}^{M}\left(1 - P_{e_i}^{(d)}\right)\lambda_{e_i}^{(d)} + C_{sf}^{(d)} \sum_{j=1}^{N} P_{s_j}^{(d)}\left(1 - P_f^{(d)}\right) \sum_{i=1}^{M}\left(1 - P_{e_i}^{(d)}\right)\lambda_{e_i}^{(d)} + \sum_{j=1}^{N} P_{s_j}^{(d)}\left(1 - P_f^{(d)}\right) \sum_{i=1}^{M} C_{fe_i}^{(d)}\left(1 - P_{e_i}^{(d)}\right)\lambda_{e_i}^{(d)} \quad (6)$$

where $\sum_{j=1}^{N} P_{s_j}^{(d)}$ is an overall data OHR at the CDN. If the surrogate servers are noncooperative, $N$ is simply set to 1 in (6)-(8). Note that the CDN does not process application object requests.

A CDN-to-cloud request-transfer cost is

$$C_{sc}(1 - P_s)\lambda_{sc} = C_{sc}^{(d)}(1 - \sum_{j=1}^{N} P_{s_j}^{(d)})(1 - P_f^{(d)}) \sum_{i=1}^{M}\left(1 - P_{e_i}^{(d)}\right)\lambda_{e_i}^{(d)} + C_{sc}^{(a)}(1 - P_f^{(a)}) \sum_{i=1}^{M}\left(1 - P_{e_i}^{(a)}\right)\lambda_{e_i}^{(a)}. \quad (7)$$

Lastly, let $P_c$ be 1 and the origin server responds or acknowledges every request. A cloud response cost, including the cost of transmitting responses back to the edge devices, is

$$P_c(C_c\lambda_{sc} + (C_{cs} + C_{sf} + C_{fe})\lambda_{ce}) = \max\begin{pmatrix} C_c^{(d)}(1 - \sum_{j=1}^{N} P_{s_j}^{(d)})(1 - P_f^{(d)}) \sum_{i=1}^{M}\left(1 - P_{e_i}^{(d)}\right)\lambda_{e_i}^{(d)}, \\ C_c^{(a)}(1 - P_f^{(a)}) \sum_{i=1}^{M}\left(1 - P_{e_i}^{(a)}\right)\lambda_{e_i}^{(a)} \end{pmatrix} + C_{cs}^{(d)} + C_{sf}^{(d)} + \sum_{i=1}^{M} C_{fe_i}^{(d)}(1 - \sum_{j=1}^{N} P_{s_j}^{(d)})\left(1 - P_f^{(d)}\right) \sum_{i=1}^{M}\left(1 - P_{e_i}^{(d)}\right)\lambda_{e_i}^{(d)} + C_{cs}^{(a)} + C_{sf}^{(a)} + \sum_{i=1}^{M} C_{fe_i}^{(a)}\left(1 - P_f^{(a)}\right) \sum_{i=1}^{M}\left(1 - P_{e_i}^{(a)}\right)\lambda_{e_i}^{(a)} \quad (8)$$

## V. EVALUATION

We evaluated our architecture acceleration by enlisting the cost performance model based on the scenario in Fig 1.

### A. Configuration

We simulated the architecture that utilized three acceleration strategies one by one to speed up cloud-service response times as follows.

- *LRU cache* : The CDN, fog, and edge layers totally used LRU caches.
- *Cloud cache* : The CDN, fog, and edge layers uniformly employed cloud caches.
- *Cloud cache with FGR* : The CDN, fog, and edge layers were equipped with cloud caches, and the edge layer additionally employed FGR.

TABLE I
SIMULATION PARAMETERS AND THEIR EMPIRICAL VALUES

| Var. | Value | Var. | Value | Var. | Value |
|---|---|---|---|---|---|
| M | 1-20 | $C_f^{(d)}$ | 5E-4-0.05 | $C_{s_j}^{(d)}$ | 2.5E-4-0.025 |
| $C_{e_i}^{(d)}$ | 0.001-0.1 | $P_f^{(d)}$ | 0.05-0.5 | $C_{sf}^{(d)}$ | 2E-4-1.5E-4 |
| $P_{e_i}^{(d)}$ | 0.05-0.65 | $C_f^{(a)}$ | 0.05-0.75 | $C_{sf}^{(a)}$ | 0.01-0.3 |
| $\lambda_{e_i}^{(d)}$ | 1-110 | $P_f^{(a)}$ | 0.35-0.85 | $C_{sc}^{(d)}$ | 0.001-0.0015 |
| $C_{e_i}^{(a)}$ | 0.1-1.5 | $C_{fe_i}^{(d)}$ | 1E-5-0.31 | $C_{sc}^{(a)}$ | 0.1-3 |
| $P_{e_i}^{(a)}$ | 0.25-0.75 | $C_{fe_i}^{(a)}$ | 0.001-0.03 | $C_c^{(d)}$ | 2.5E-4-0.025 |
| $\lambda_{e_i}^{(a)}$ | 1 | $C_{fs}^{(d)}$ | 1E-4-1.5E-4 | $C_c^{(a)}$ | 0.01-0.15 |
| $C_{e_if}^{(d)}$ | 1E-5-1.5E-5 | $C_{fs}^{(a)}$ | 0.01-0.3 | $C_{cs}^{(d)}$ | 0.001-0.0015 |
| $C_{e_if}^{(a)}$ | 0.001-0.03 | $P_{s_j}^{(d)}$ | 0.14 | $C_{cs}^{(a)}$ | 0.1-3 |

The simulation relied on model parameter values in Table I. The values were estimated by using various practical tools and previous empirical results [34], [35], [37], [38]. We measured the average $P_{e_i}^{(d)}$ value of each $e_i$ by using a web browser's LRU-OHR extension. $M$ was limited to 20 to prevent overloading the only fog node and end-to-end network-bandwidth saturation. $C_{e_i}^{(d)}$ and $\lambda_{e_i}^{(d)}$ values for every $e_i$ were captured by using [38] when users loaded web pages causing one or more requests for each web page's embedded object(s). The unit of $\lambda$ is requests per sec. Experience from [35] suggested $C_{e_i}^{(a)}$, $P_{e_i}^{(a)}$, and $\lambda_{e_i}^{(a)}$ values for a Java-enterprise cloud-server application (implemented with JBoss Enterprise Application Platform, for example) running on an off-premise private-cloud server. The application consisted of both replicable and database-accessing nonreplicable portions. A $C_{e_i}^{(a)}$ value was slightly larger than a $C_{e_i}^{(d)}$ value as invoking the application caused the loading of several dependent modules. A $P_{e_i}^{(a)}$ range was higher but narrower than that of $P_{e_i}^{(d)}$ as, based on our experience, most users routinely accessed a small and repeated set of server application functions to complete their pieces of work. A $\lambda_{e_i}^{(a)}$ value relied on user speed to interact with the server application. Both $C_{e_if}^{(d)}$ and $C_{e_if}^{(a)}$ values were $M$-shared wireless-LAN latencies for sending 66B-1KB server-input data and 5KB-200KB server-input Java objects, respectively. A $C_f^{(d)}$ value halved the $C_{e_i}^{(d)}$ value as potentially using double I/O speed. A $P_f^{(d)}$ value reflected a shared-cache OHR in serving all $e_i$ in the same vicinity and had its value taken from [29]. A $C_f^{(a)}$ value was derived from the $C_{e_i}^{(a)}$ value by assuming the fog node was only twice as fast as each $e_i$ due to higher loads. A $P_f^{(a)}$ value relied on the $P_{e_i}^{(a)}$ one but occurred on cloud service functions frequently used by all $e_i$ and thus placed on the fog node. A $C_{fe_i}^{(d)}$ value was wireless LAN latencies for sending/forwarding an 85B to 2MB response. A

$C_{fe_i}^{(a)}$ value equaled the $C_{e_if}^{(a)}$ one. Both $C_{fs}^{(d)}$ and $C_{fs}^{(a)}$ values were 10 times as expensive as the $C_{e_if}^{(d)}$ and $C_{e_if}^{(a)}$ values, respectively, due to the congestion of the Internet connection between the fog node and the CDN as revealed by a traceroute command. $P_{s_j}^{(d)}$ was the OHR of a shared LRU cache and also came from [29]. A $C_{s_j}^{(d)}$ value was twice lower than the $C_f^{(d)}$ value as exploiting sophisticated method and platform. Both $C_{sf}^{(d)}$ and $C_{sf}^{(a)}$ values equaled the $C_{fs}^{(d)}$ and $C_{fs}^{(a)}$ ones due to symmetric Internet bandwidth. $C_{sc}^{(d)}$, $C_{cs}^{(d)}$, $C_{sc}^{(a)}$, and $C_{cs}^{(a)}$ values were identical and 10 times as costly as the $C_{fs}^{(d)}$ value due to the origin server was further away from the edge than the fog node. A $C_c^{(d)}$ value equaled the $C_{s_j}^{(d)}$ one owing to no I/O optimization but lower utilization. A $C_c^{(a)}$ value was 5 times lower than the $C_f^{(a)}$ value. In addition, to derive the OHRs of cloud caches, we multiplied each of $P_{e_i}^{(d)}$, $P_f^{(d)}$, and $P_{s_j}^{(d)}$ of the LRU caches by the factor of 1.10 as suggested by our previous empirical results in [29].

$\mathbb{E}_c$ was finally calculated by letting edge devices independently generate request streams for one second, a CDN employ noncooperative surrogate servers (i.e., $N$ = 1) as in a commercial CDN [10], and a fog node be isolated instead of federated. Herein, we carried out the serial scheme of the model. For any ranging values, we used their arithmetic means.

### *B. Results and Findings*

Fig 2 compares cloud-service response times based on varied $M$ values. Though all of the strategies imposed the quadratic growth of response times with respect to $M$, the cloud cache with and without FGR strategies outperformed the LRU cache one for every $M$ value and even by far when $M$ notably increased. To be more precise, the response-time growth rates ($\frac{d\mathbb{E}_c}{dM}$) of LRU cache, cloud cache, and cloud-cachewith-FGR strategies were represented by $3.2162M+3.2336$, $2.8812M+3.2863$, $2.8750M+2.4719$, respectively. This finding serves as the practical design guideline of edge device density per fog node.

Fig 3 shows time portions constituting each strategy's response time when $M$ = 1. Both LRU-cache and cloud-cache strategies yielded the same time spends in descending order : cloud response (8), fog node response (4), CDN-to-cloud transfer (7), edge device response (2), CDN response (6), fog-to-CDN transfer (5), and edge-to-fog transfer (3). With the cloud-cache-with-FGR strategy, the order of time spends came out the same as the others except for the edge device response and the CDN-to-cloud transfer. This was because not only some application object requests were early responded by FGR at the edge (thus raising the edge-device response time) but also the FGR cut down the number of application object requests processed at the cloud (thus lessening the CDN-to-cloud transfer time). In overall, the cloud-cache-with-FGR strategy defeated the LRU cache one by 23.85%.

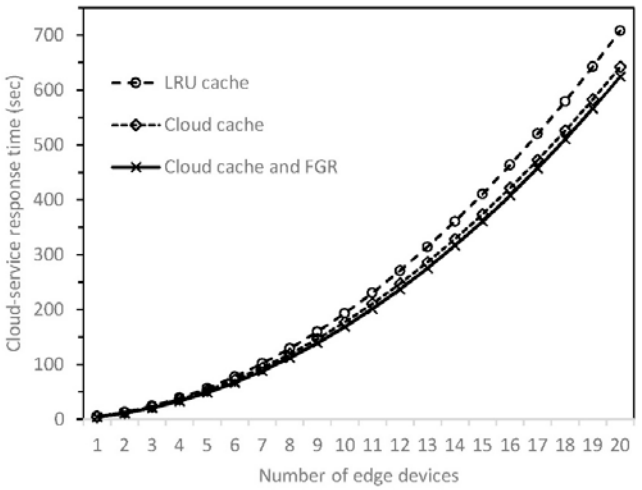


Fig. 2. Cloud service response time.

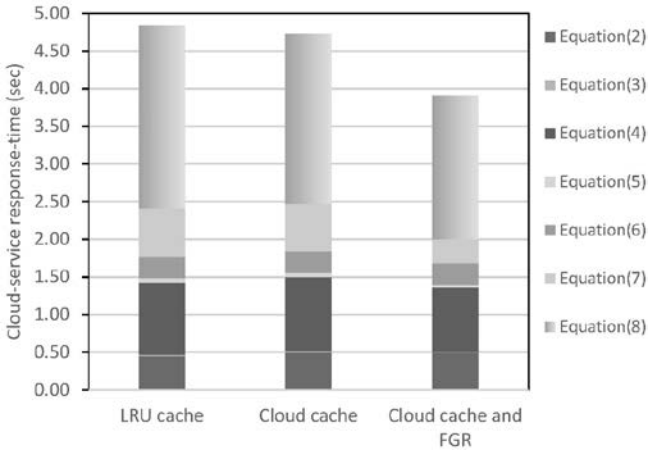


Fig. 3. Cloud-service response-time anatomy for 1 edge device.

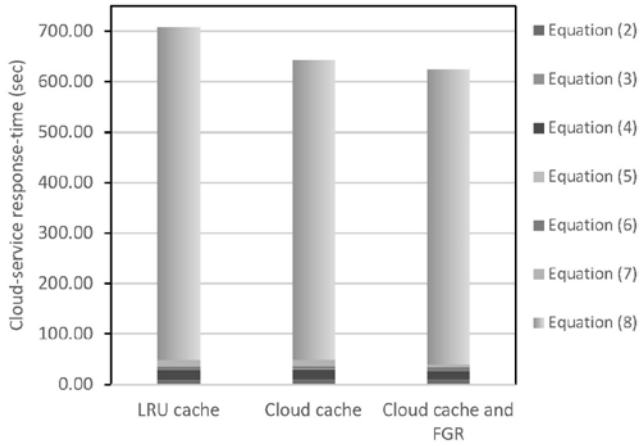


Fig. 4. Cloud-service response-time anatomy for 20 edge devices.

Fig 4 breaks down time fragments when $M$ was raised to 20. Obviously, cloud response took the majority time portions. An overall cloud-service response time improved by 13.37% when the cloud-cache-with-FGR strategy contributed instead of the LRU one. A response time gap between the cloud cache strategy and the cloud-cache-with-FGR one became narrower than that of Fig 3. This was because more workloads were shifted to the high-latency cloud layer. Not showed here, we also found through our experiment that a fundamental

observation in [31], the greater fallback requests to higher-level caches the longer response times, would not be true if the higher levels were more powerful such as fast processing power, high-speed I/O, and wider network bandwidth.

## VI. Conclusion

This paper proposes a pair of cloud cache-based approaches, cloud caches and cloud caches with FGR, to improve cloud-service response times in cloud-CDN-fog-edge hierarchy. They are evaluated by using a novelly unified cost-performance model to compare temporal cost with pure LRU cache approach. Since the cloud caches deliver near-optimal OHR, faster response times can be expected than the LRU caches. Furthermore, we point out an architectural opportunity to optimize edge computing with FGR to reduce application response times and cloud and fog workloads. Compared to a traditional LRU cache, cloud cache together with FGR accelerate cloud-service response time by 23.85% for a single edge device and 13.37% for 20 edge devices. The response time increases of LRU cache and cloud-cache-with-FGR strategies are $3.2162M+3.2336$ and $2.8750M+2.4719$, respectively.

## References


[1] Cloudflare, Inc. (2019) What is a cdn? [Online]. Available: https://www.cloudflare.com/learning/cdn/what-is-a-cdn/

[2] M. Iorga, L. Feldman, R. Barton, M. J. Martin, N. S. Goren, and C. Mahmoudi, "Fog computing conceptual model," National Institute of Standards and Technology, Tech. Rep. Special Publication (NIST SP) - 500-325, March 2018.

[3] C. Mahmoudi, A. Battou, and F. Mourlin, "Formal definition of edge computing: An emphasis on mobile cloud and iot composition," in *Proceedings of The Third IEEE International Conference on Fog and Mobile Edge Computing*, June 2018.

[4] D. Gash. (2019) Http caching. [Online]. Available: https://developers.google.com/web/fundamentals/performance/getstarted/httpcaching-6

[5] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5031–5044, May 2019.

[6] T. Banditwattanawong, M. Masdisornchote, and P. Uthayopas, "Multiprovider cloud computing network infrastructure optimization," *Future Generation Computer Systems*, vol. 55, pp. 116 – 128, 2016.

[7] A. Vakali and G. Pallis, "Content delivery networks: status and trends," *IEEE Internet Computing*, vol. 7, no. 6, pp. 68–74, Nov 2003.

[8] Google Cloud. (2019) Cloud cdn documentation. [Online]. Available: https://cloud.google.com/cdn/docs/

[9] A. Sundarrajan, M. Feng, M. Kasbekar, and R. K. Sitaraman, "Footprint descriptors: Theory and practice of cache provisioning in a global cdn," in *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '17. New York, NY, USA: ACM, 2017, pp. 55–67.

[10] M. Pathan and R. Buyya, "A taxonomy of cdns," in *Content Delivery Networks, Lecture Notes in Electrical Engineering*, Jan 2008, pp. 33–77.

[11] Akamai. (2019) Akamai developer : Caching. [Online]. Available: https://developer.akamai.com/article/content-caching

[12] D. Wessels and K. Claffy. (1997) Internet cache protocol (icp), version 2. [Online]. Available: https://tools.ietf.org/html/rfc2186

[13] ——. (1997) Application of internet cache protocol (icp), version 2. [Online]. Available: https://tools.ietf.org/html/rfc2187

[14] E. Nygren, R. K. Sitaraman, and J. Sun, "The akamai network: a platform for high-performance internet applications." *Operating Systems Review*, vol. 44, pp. 2–19, Jan 2010.

[15] Akamai. (2019) Content caching. [Online]. Available: http://developer.akamai.com/legacy/learn/Caching/ContentCaching.html

[16] G. Neglia, D. Carra, M. Feng, V. Janardhan, P. Michiardi, and D. Tsigkari, "Access-time aware cache algorithms," in *2016 28th International Teletraffic Congress (ITC 28)*, vol. 01, 2016, pp. 148–156.

[17] Amazon Web Services. (2019) Measuring cloudfront performance. [Online]. Available: https://aws.amazon.com/th/blogs/networking-andcontent-delivery/measuring-cloudfront-performance/

[18] M. Garetto, E. Leonardi, and V. Martina, "A unified approach to the performance analysis of caching systems," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 1, no. 3, pp. 12:1–12:28, May 2016. [Online].

[19] S. UlIslam, H. A. Khattak, F. F. Qureshi, V. Chang, and J. M. Pierson, "Leveraging utilization as performance metric for cdn enabled internet of things," *Measurement*, 2019.

[20] W. Zhang, Z. Lu, Z. Wu, J. Wu, H. Zou, and S. Huang, "Toy-iot-oriented data-driven cdn performance evaluation model with deep learning," *Journal of Systems Architecture*, vol. 88, pp. 13 – 22, 2018.

[21] F. Jiang, Z. Yuan, C. Sun, and J. Wang, "Deep q-learning-based content caching with update strategy for fog radio access networks," *IEEE Access*, vol. 7, pp. 97 505–97 514, 2019.

[22] Z. Li, J. Chen, and Z. Zhang, "Socially aware caching in d2d enabled fog radio access networks," *IEEE Access*, vol. 7, pp. 84 293–84 303, 2019.

[23] E. Baccarelli, P. G. V. Naranjo, M. Scarpiniti, M. Shojafar, and J. H. Abawajy, "Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study," *IEEE Access*, vol. 5, pp. 9882–9910, 2017.

[24] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the suitability of fog computing in the context of internet of things," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 46–59, Jan 2018.

[25] Mozilla Developer Network. (2019) Browser storage limits and evictioncriteria. [Online]. Available: https://developer.mozilla.org

[26] The Chromium Projects. (2019) Disk cache. [Online]. Available: https://www.chromium.org/developers/design-documents/networkstack/disk-cache

[27] T. Banditwattanawong, "From web cache to cloud cache," in *Advances in Grid and Pervasive Computing*, ser. Lecture Notes in Computer Science, R. Li, J. Cao, and J. Bourgeois, Eds. Springer Berlin / Heidelberg, 2012, vol. 7296, pp. 1–15.

[28] ——, "The optimality and complexity of offline cache replacement policies for nonuniform objects," *International Journal of Future Computer and Communication*, vol. 7, no. 3, pp. 63–67, Sep 2018.

[29] ——, "The empirical discovery of near-optimal offline cache replacement policies for nonuniform objects," in *Recent Advances in Information and Communication Technology 2018*. Springer International Publishing, 2019, pp. 286–296.

[30] M. N. R. Fielding and J. Reschke. (2014) Hypertext transfer protocol (http/1.1): Caching. [Online]. Available: http://www.ietf.org/rfc/rfc7234.txt

[31] A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrell, "A hierarchical internet object cache," in *Proceedings of the 1996 Annual Conference on USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 1996, pp. 13–13.

[32] P. Rodriguez, C. Spanner, and E. W. Biersack, "Analysis of web caching architectures: hierarchical and distributed caching," *IEEE/ACM Transactions on Networking*, vol. 9, no. 4, pp. 404–418, Aug 2001.

[33] R. E. Abdouni Khayari, A. Musovic, A. Lehmann, and M. B¨ohm, "A model validation study of hierarchical and distributed web caching model," in *Proceedings of the 2010 Spring Simulation Multiconference*. Society for Computer Simulation International, 2010, pp. 98:1–98:6.

[34] C. Chiang, M. Ueno, M. T. Liu, and M. E. Muller, "Modeling web caching schemes for performance studies," in *Proceedings 2000 International Conference on Parallel Processing*, 2000, pp.243–250.

[35] T. Banditwattanawong, S. Hidaka, H. Washizaki, and K. Maruyama, "Soom: Scalable object-oriented middleware for cooperative and pervasive computings," *IEICE Transactions on Communications*, vol. 90-B, pp.728–741, 2007.

[36] I. Papapanagiotou, E. M. Nahum, and V. Pappas, "Smartphones vs. laptops: Comparing web browsing behavior and the implications for caching," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 1, pp. 423–424, Jun. 2012.

[37] Wireshark. (2019) Wireshark. [Online]. Available: https://www.wireshark.org

[38] Google. (2019) Chrome devtools. [Online]. Available: https://developers.google.com/web/tools/chrome-devtools/

**Thepparit Banditwattanawong** was born in Bangkok, Thailand. He received B.Eng. degree (Honors) in computer engineering from King Mongkut's Institute of Technology Ladkrabang, Thailand and M.Eng. degree from Asian Institute of Technology, Thailand. He obtained his Ph.D. degree in informatics from the National Institute of Informatics (NII), The Graduate University for Advanced Studies, Tokyo, Japan.

He is currently an Assistant Professor with the Department of Computer Science, Kasetsart University, Bangkok, Thailand. His main areas of research interests include computer network optimization, cloud computing, and distributed computing.

**Masawee Masdisornchote** was born in Bangkok, Thailand. She received B.S. in statistics from Silpakorn university, Thailand and M.S. in information technology from King Mongkut's Institute of Technology Ladkrabang, Thailand.

She is currently an Assistant Professor and the director of Business Computer department at Sripatum University, Thailand. Her main research area is data science.