一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
NS2021-127(2022-03)

共起度を考慮した複数Webキャッシュサーバに対するキャッシュ制御方式

他力　誠人†　　三角　真†　　上山　憲昭††

† 福岡大学工学部電子情報工学科　〒 814–0180 福岡市城南区七隈 8–19–1
†† 立命館大学　情報理工学部　〒 525–8577 滋賀県草津市野路東 1 丁目 1–1
E-mail: †tl171289@cis.fukuoka-u.ac.jp, ††mmisumi@fukuoka-u.ac.jp, †††kamiaki@fc.ritsumei.ac.jp

あらまし　Web ページの待ち時間を削減する方法として，Web ページを構成する複数のオブジェクトを並列取得することで，Web ページ表示の待ち時間を削減する HTTP/2 や HTTP/3 が標準化され広く普及している．HTTP/2 や HTTP/3 の Web ページの表示待ち時間削減効果は，同一の配信サーバから取得するオブジェクト集合に対して有効な反面，多数の配信サーバからのオブジェクトの取得については各配信サーバにコネクションを作成しなければならないため効果が低い．一方で筆者らは，任意の個数のオブジェクト組が複数の Web ページ内で出現することを示す共起という概念を提案し，オブジェクト組が出現する WEB ページの数を共起度と定義し，約 8,000 の Web ページ URL から取得したオブジェクトについて共起度の分析と，共起度のキャッシュ制御法への適用可能性を示している．そこで本稿では，HTTP/2 や HTTP/3 の並列取得効果の向上を目的とし，複数の HTTP/2 や HTTP/3 に対応したキャッシュサーバが存在するとき，共起を考慮したキャッシュ制御を行うことで，共起するオブジェクト組を構成するオブジェクトを，単一のキャッシュサーバに可能な限り集約する手法を提案する．また計算機シミュレーションにより，その有効性を確認する．
キーワード　HTTP/2，Web オブジェクト，共起

Cache Control Method for Multiple Web Cache Servers Based on Co-Occurrence Degree

Masato TARIKI†, Makoto MISUMI†, and Noriaki KAMIYAMA††

† Fukuoka University　8–19–1, Nanakuma, Jounan, Fukuoka 814–0180
†† The University of Tokyo　1–1–1, Nojihigashi, Kusatsu, Shiga 525–8577
E-mail: †tl171289@cis.fukuoka-u.ac.jp, ††mmisumi@fukuoka-u.ac.jp, †††kamiaki@fc.ritsumei.ac.jp

Abstract　As a method of reducing the waiting time of a web page, HTTP/2 and HTTP/3, which reduce the waiting time of displaying a web page by acquiring multiple objects that make up the web page in parallel, have been standardized and widely used. The effect of reducing the response time by HTTP/2 and HTTP/3 is effective for the object set acquired from the same object server. However, when obtaining objects from many object servers, and client host connects to each object server, the effect is low because of using many HTTP connections. To solve this issue, we proposed the concept of co-occurrence, which indicates that an arbitrary number of object sets appear in multiple webpages, and defined the number of web pages in which object sets appear as the degree of co-occurrence. It shows the applicability of the co-occurrence degree to cache control methods. Therefore, in this paper, for improving the parallel acquisition effect of HTTP/2 and HTTP/3, when there are multiple cache servers that support HTTP/2 and HTTP/3, we propose a cache control method that considers co-occurrence. The proposed method aggregates the objects that make up the co-occurring object set into a single cache server as much as possible. In addition, its effectiveness is confirmed by computer simulation.
Key words　HTTP/2, web object, co-occurrence

1. はじめに

世界中の多くの人々が毎日インターネットを利用しており，インターネットの中でも特に Web 閲覧サービスはインターネット上で最も普及したサービスの一つである．現在，Web ページは静止画やテキストといった静的なオブジェクトから広告などの動的なオブジェクトなどのさまざまなオブジェクトから構成されており，複雑性が増してきている [3]．そのような中，インターネット利用者はより快適な Web 品質を求めており，多くの研究者が様々な観点から研究を行っている．Web 応答時間，つまりアクセスした Web ページが表示されるまでの利用者の待ち時間の削減もその研究の一つである [14]．Web ページの待ち時間を削減する方法として HTTP/2 や HTTP/3 がインターネット上で標準化され広く普及している．従来使用されてきた HTTP/1.1 に SPDY といった WEB ページの表示を高速化するためのプロトコルを移行することによって [9] [18] [19]，HTTP/2 や HTTP/3 は HTTP/1.1 の互換性を持たせながらも様々な性能を向上させた [12]．

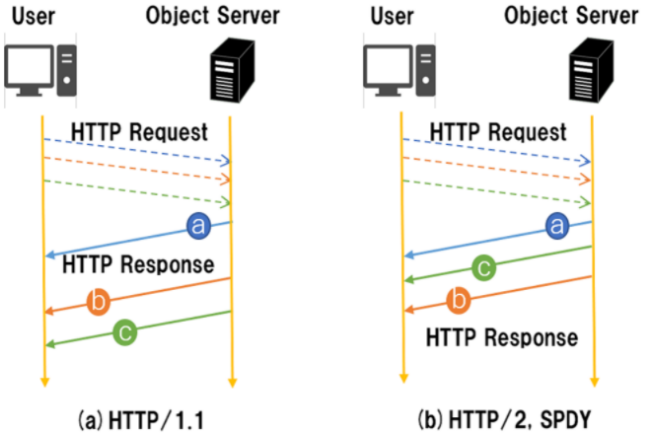

図 1　HTTP/1.1 と HTTP/2 (SPDY) の比較

図 1 は HTTP/1.1 と HTTP/2 (SPDY) の比較である．黄色の矢印がユーザの待ち時間を表し，アルファベット a，b，c がユーザがリクエストしたオブジェクトである．

HTTP/1.1 はサーバに要求が到着した順でしかサーバはオブジェクトを送信できないため，例のようなオブジェクト b のサーバでの処理に時間を要する場合，後続の要求であるオブジェクト c も影響を受けて送信時刻が遅れてしまう．それに対し HTTP/2 や HTTP/3 は，各パケットが属する HTTP Response が識別できる情報をヘッダに入れることで，任意の順番でサーバはオブジェクトの送信が可能となり，図 1(b) に示すようにオブジェクト c を先に送信することでトータルでの遅延時間の低減が可能となる．このことから HTTP/2 や HTTP/3 はオブジェクトを 1 つのサーバから並列的に取得するうえで効率の良い手法である [2]．しかし 1 つのサーバではなく複数のサーバからオブジェクトを並列的に取得するうえで，HTTP/2 や HTTP/3 は効果を発揮できない [7]．

図 2 の (a) は複数のサーバから青，橙，緑オブジェクトを取得する際，(b) は単一のサーバからこれらオブジェクトを取得する際の HTTP/2 の働きを示す．1 つのサーバからオブジェクトを取得する場合は，サーバと利用者の端末をつなぐコネクションを 1 つしか使用しないが，複数のサーバを利用する場合は利用するサーバの数だけコネクションが増加する．ひとつのサーバから複数のオブジェクトを取得する場合は，前述したように並列的に取得することができる HTTP/2 や HTTP/3 の効果を発揮できる．しかし複数コネクションが存在する場合は，複数のサーバ間にそれぞれ作成されたコネクションを共有できず，青，橙，緑それぞれのオブジェクトが別々のコネクションで転送されるため，一つのサーバを利用した時と比較すると HTTP/2 や HTTP/3 の効果が低下する．

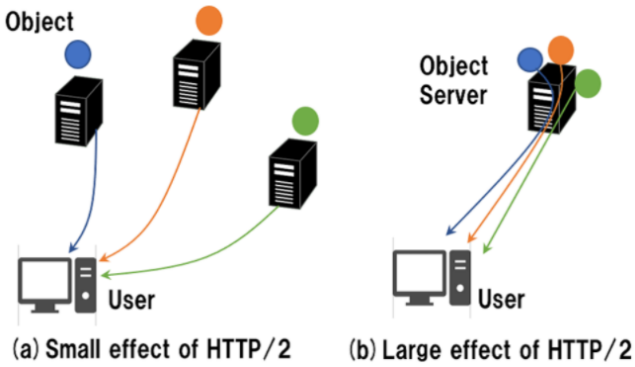

図 2　HTTP/2 による単一・複数サーバからのオブジェクト取得

1.1 HTTP/2・HTTP/3 と CDN

本研究で共起に着目した理由として CDN（Content Delivery Network) の存在がある [15] [16] [17]．CDN はネットワーク上の世界中ありとあらゆる場所に設置されたキャッシュサーバにオリジンサーバ (配信者) から得たオブジェクトのコピーをキャッシュし，ユーザの近くに存在するキャッシュサーバからオブジェクトを配信することで，ユーザの Web 応答待ち時間を低減させる仕組みである．また，オリジンサーバと同じオブジェクトを持ったキャッシュが世界中にあることで，アクセスが一つに集中せずオリジンサーバの負荷の低減が可能なこともメリットである．さらに，HTTP/2 や HTTP/3 を利用するには配信サーバとユーザ端末の両方が対応する必要があるが，オブジェクトのオリジナルを提供するオリジンサーバが HTTP/2 や HTTP/3 に未対応であっても，CDN 事業者がキャッシュサーバを HTTP/2 や HTTP/3 に対応させることでキャッシュサーバとユーザ端末間で HTTP/2 や HTTP/3 を利用することができる．複数のキャッシュサーバを利用して一つのサーバへの負荷を低減させる点，Web 応答待ち時間の削減を目指すという点，オリジンサーバが HTTP/2 や HTTP/3 に未対応でも HTTP/2 や HTTP/3 を利用できる点の 3 つの点から CDN を本論文の共起によるキャッシュ制御の適応先として想定する．

2. 共　　起

本節では，提案手法で用いる共起について説明する．

2.1 共起とは

共起とは，任意の個数のオブジェクト組が複数の Web ページ内で出現することをさす．また，そこであるオブジェクトの組に対し，その組が出現する Web ページの数を共起度と定義する [11]．

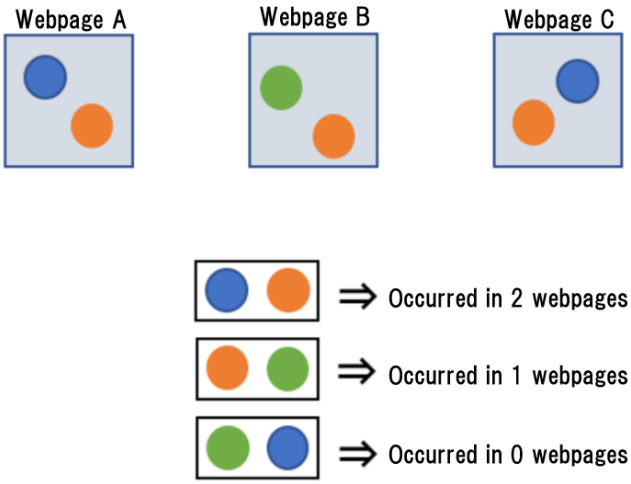

図 3　共起・共起度の例

図 3 に共起と共起度の例を示す．Webpage A には青と橙，Webpage B には緑と橙，Webpage C には青と橙のオブジェクトが出現している．この場合，Webpage A と Webpage C は同じオブジェクト組が出現しているため，青と橙のオブジェクト組は共起オブジェクト組である．また，青と橙のオブジェクト組は 2 つのページに存在するため共起度は 2，緑と橙のオブジェクト組は 1 つのページに存在するため共起度は 1，青と緑のオブジェクト組はいずれのページにも存在しないため共起度は 0 となる．Web 応答時間の低減効果を高めるには，少数の配信サーバから多数のオブジェクトを取得する必要があるため，共起度の高いオブジェクト組が優先的に同じキャッシュに残るよう，キャッシュへの挿入や置換を行うことが望ましい．

3.　共起を考慮した Web キャッシュ挿入・置換方式

複数のキャッシュサーバが存在するとき，共起を考慮したキャッシュ制御を行うことで，同じキャッシュサーバに単一の Web ページにアクセスした際に要求されるオブジェクトを可能な限り集約させる．キャッシュ制御は挿入時と置換時の 2 つがありが，挿入時に共起を考慮することで同じサーバに共起組のオブジェクト集約させ，置換時に共起を考慮することで共起組のオブジェクトが排出されにくくなる．後述する挿入ポリシィと置換ポリシィにおいて詳細に説明する．

3.1　提 案 方 式

本論文では共起を考慮したキャッシュ制御，つまり共起するオブジェクト組を構成するオブジェクトを単一のキャッシュサーバに可能な限り集約する手法を提案する．

図 4 は複数のキャッシュサーバを利用する際のオブジェクトの配置を示す．HTTP/2 を利用するうえでは 1 節で述べたように，図 4 の右側に示すような 1 つのキャッシュサーバからできるだけ多くのオブジェクトを取得できたほうが効率が良い．そこで共起を考慮したキャッシュ挿入制御と置換制御をそれぞれ用いることで Web ページリクエスト時に発生したオブジェクトをひとつのキャッシュに可能な限り集約させ，HTTP/2 や HTTP/3 の性能効果向上を図る．

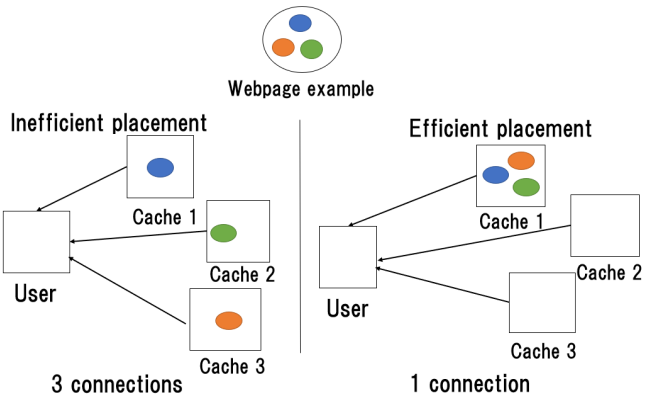

図 4　複数のキャッシュサーバに対するオブジェクト配置

3.2　挿入ポリシィ

複数のキャッシュサーバが存在するとき，オブジェクトを挿入するキャッシュサーバを選択する必要があるが，HTTP/2 や HTTP/3 の並列取得効果を得るためには，共起オブジェクト組は同一のキャッシュサーバに収納することが望ましい．そこで，オブジェクトをキャッシュサーバに新たにキャッシュするキャッシュサーバを選択するとき，既にキャッシュされているオブジェクトに，新たにキャッシュするオブジェクトが加わることによって，新たに生じる共起オブジェクト組の数をキャッシュサーバ毎に求め，その値が最も大きいキャッシュサーバを新たなオブジェクトをキャッシュするキャッシュサーバとして選択する．

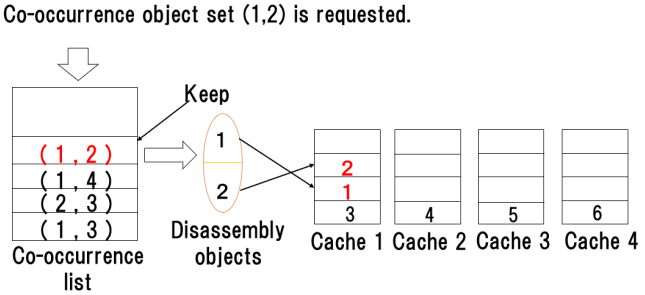

図 5　共起を考慮した挿入アルゴリズム

図 5 は共起を考慮した際のオブジェクトの挿入の流れを示す．ここでの共起リストとは共起オブジェクト組の情報を管理する場所である．オブジェクトの挿入はオブジェクトごとに行われるため，それぞれのオブジェクトが共起組のオブジェクトであったか単一のオブジェクトであったかを判断できない．そこで共起オブジェクト組がキャッシュに挿入される場合，共起リストに共起ペアを保存してからオブジェクト組を分解する．次に各オブジェクトを挿入するキャッシュサーバを決定するが，共起リストの情報を参考に，新しくキャッシュするオブジェクトを追加することによって，各キャッシュサーバで新たに作成可能な共起オブジェクト組の数を算出する．そして新たに作成可能な共起オブジェクト組が最も多く存在するキャッシュサーバを，そのオブジェクトを挿入するキャッシュサーバに決定する．新たに作成可能な共起オブジェクト数が同数の場合は，それらキャッシュサーバの中で最初にチェックを行ったものを選択する．

図 5 ではオブジェクト 1 と 2 の組を挿入している．共起リストを確認すると，オブジェクト 1 の共起ペアとしてオブジェクト 3, 4 があげられる．キャッシュに挿入するオブジェクトのペアが存在するか探索すると，Cache 1 及び Cache 2 にそれぞれ一つずつ存在する．この場合，Cache 1 を先に探索したため，オブジェクト 1 は Cache 1 に挿入される．次にオブジェクト 2 の共起ペアとしてはオブジェクト 1 と 3 があげられる．同様に探索を行うと，Cache 1 に最も多く共起ペアが存在するため，オブジェクト 2 は Cache 1 に挿入される．

3.3 置換ポリシィ

キャッシュサーバのキャッシュがあふれたとき，キャッシュ済みオブジェクトのいくつかを排出する必要がある．提案手法では，オブジェクトを排出する際に，排出オブジェクトの共起ペアがキャッシュ内に存在しているかを確認する．存在していない場合は Least Recently Used (LRU) 方式に基づいて排出するが，存在している場合は排出予定だったオブジェクトを残し，LRU に基づき次の候補オブジェクトを排出する．また共起を考慮したキャッシュ置換制御では共起オブジェクト組がリクエストされた場合，共起オブジェクト組それぞれの単一のオブジェクトに加えて共起オブジェクト組単位でアクセス順のリストを管理する．リストに従って LRU 方式を用いて排出するオブジェクトを決定する．このとき，LRU のアクセス順リストを共起オブジェクト組で管理することで，LRU のアクセス順リスト内には，同じオブジェクトが異なるオブジェクト組を構成するオブジェクトとして複数回出現する．LRU リスト内に複数回出現するオブジェクトであったとしても，キャッシュとしてはオブジェクトひとつについては一つしかキャッシュを行わない．そこで，あるオブジェクトや共起オブジェクト組がリクエストされた際にオブジェクトごとに参照数をカウントする．キャッシュ内のオブジェクトが LRU リストから参照されている回数を用いてキャッシュにおけるオブジェクトの保持の要否を判断する．

あるオブジェクト，あるいは共起オブジェクト組のリクエストが発生したとき，他の共起オブジェクト組によってそれらのオブジェクトは既にキャッシュ内には存在しているものの，LRU のリストにはリクエストのあったオブジェクトあるいは共起オブジェクト組が存在しない状況が発生する．そのようなとき，LRU のリストにリクエストされたオブジェクトあるいは共起オブジェクト組を追加し，キャッシュされているオブジェクトの参照数をインクリメントする．排出するオブジェクトを決定するとき，LRU に従い最も使われていないオブジェクトまたは共起オブジェクト組を取り出し，そこに含まれるオブジェクトについて，キャッシュの参照数をデクリメントする．参照数が 0 になったオブジェクトは，キャッシュから削除する．これを，必要な数の空き領域が得られるまで繰り返す．

図 6 に，置換ポリシィによる排出制御と単純な LRU を用いた排出制御の比較を例示する．キャッシュサーバの中身をオブジェクト 1,2,3,4 があらかじめ格納された状態とし，オブジェクト 1 とオブジェクト 5 が順番に参照されるときの動作を示す．その時のオブジェクト情報を LRU リストに示す．LRU リストに存在する一番左のオブジェクトが最も過去の時間に参照

		Input	
	Initial	1	5
Cache	1	1	1
	2	2	2
	3	3	5
	4	4	4
LRU list	(1,2),1,2,3,4	2,3,4,(1,2),1	4,(1,2),1,5

⇒Removing object 3

Proposal method

		Input	
	Initial	1	5
Cache	1	1	1
	2	2	5
	3	3	3
	4	4	4
LRU list	1,2,3,4	2,3,4,1	3,4,1,5

⇒Removing object 2

LRU

図 6　置換ポリシィによるキャッシュ排出の比較

されたものを示し，(1,2) のような表記は共起オブジェクト組を示す．まずオブジェクト 1 が参照されるとき，単純な LRU はオブジェクト 1 の情報を更新するが，提案方式だとオブジェクト 1 の情報に加えて共起組である (1,2) の情報も更新する．次にオブジェクト 5 が参照されるとき，キャッシュサーバの中身があふれるため置換を行う必要がある．単純な LRU だと排出されるオブジェクトが 2 となるが，提案方式だとオブジェクト 2 が排出されても LRU リストに (1,2) が残っているためオブジェクト 2 を排出できない．そのため排出されるオブジェクトは 3 となる．

4. 提案評価

4.1 比較方式

提案方式と比較する従来の挿入方式としては，ラウンドロビン方式を採用する．あるオブジェクトや共起オブジェクトのリクエストが発生したとき，単一のオブジェクトに分解して複数のキャッシュサーバを順次に選択し，各キャッシュサーバに均等にオブジェクトを格納する．提案方式と比較する従来の置換方式としては，単一のオブジェクトに着目した単純な LRU 方式を採用する．排出するオブジェクトを決定するときは，キャッシュに格納されてから最も長い時間アクセスがなかったオブジェクトから順番に選択する．

4.2 評価条件

オブジェクト組が出現する Web ページの数を共起度と定義する．アクセス数の多い高人気の Web ページに含まれるオブジェクトを本研究の対象とするため，adult, art, business, computers, games, health, home, kids and teens, news, recreation, reference, regional, science, shopping, society, sports の 16 のカテゴリごとに,Web ページのアクセス数のランキングを公開している Alexa の Web ページから [1]，各カテゴリに対して上位 500 の Web ページの URL を測定対象として用意し，約 8,000 の Web ページ URL から取得したオブジェクトについて共起度の分析を行う [13]．各カテゴリの人気度順に基づき URL を並び変えることで，人気度の高い Web サイトへのアクセスの偏りが発生することを想定した要求頻度を再現し (Zipf 分布のパラメータ θ に従いリクエストを発生)，1 シミュレーションごとに 10,000 回のリクエストを発生させる．各キャッシュサーバのキャッシュサイズを全オブジェクトの 0.2%から 10% の間でシミュレーションを実施する．またキャッシュサーバ台数が 4 台と 8 台の場合で各々評価する．オブジェクトを探索する際にはすべてのキャッシュサーバから探索を行う．このと

き，キャッシュヒット率やオブジェクト集約度合いはキャッシュサーバ 1 つずつ結果を出力し，結果の合計の平均値を評価値とする．

4.3 評価指標

挿入にラウンドロビン方式を用い，置換に単純な LRU 方式を利用した場合 (LRU)，置換は単純な LRU 方式で挿入に共起を考慮した方式 (Insertion with co-occurrence)，挿入はラウンドロビン方式で置換に共起を考慮した方式 (Replacement with co-occurrence) の三つの方式で，平均オブジェクト集約度とキャッシュヒット率を比較する．ただし Web ページ w をリクエストしたときの集約度 C_w を $C_w = \sum_{s \in S} {O_{w,s}}^2 / \sum_{s \in S} O_{w,s}$ と定義する．ここで S は全キャッシュサーバ集合を，s は各キャッシュサーバを，$O_{w,s}$ はすでにキャッシュ s に存在する Web ページ w を構成するオブジェクトの数を表す．そしてキャッシュヒットした要求の集約度の平均値を，平均集約度と定義する．

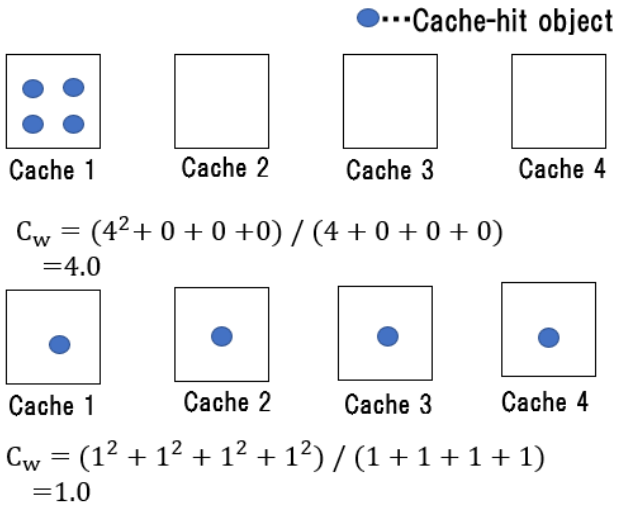

図 7　集約度の導出例

図 7 に，平均集約度の計算例を示す．図の上部はオブジェクトに偏りがある場合 (一つのキャッシュサーバにキャッシュヒットしたオブジェクトが集約されている場合) を示し，図の下部はオブジェクトに偏りがない場合 (すべてのキャッシュサーバにキャッシュヒットしたオブジェクトが分散されている場合) を示す．キャッシュオブジェクトに偏りがある場合は $C_w = 4.0$ となり，キャッシュオブジェクトに偏りがない場合は $C_w = 1.0$ となる．どちらの例もキャッシュヒットしたオブジェクトの総数は等しいが，一つのキャッシュサーバにキャッシュヒットしたオブジェクトが集約することにより，C_w の値は大きくなる．

4.4 評価結果

図 8 はキャッシュサーバ 4 台時の各方式の平均集約度を，各キャッシュの容量を総オブジェクト数で除して正規化した値に対してプロットする．また図 9 には同様に各方式の平均キャッシュヒット率を示す．平均集約度はどの方式ともにキャッシュサイズの増加に伴い増加した．また，平均集約度は挿入による提案方式がラウンドロビン方式の結果を大きく上回った．さらにキャッシュサイズが大きくなるにつれ提案方式とラウンドロビン方式との差が大きくなった．一方平均ヒット率はラウンドロビン方式が上回っているが大きな差はみられなかった．また，置換の提案方式も平均集約度は LRU 方式を上回るが，キャッシュサイズによっては LRU 方式のほうが上回る箇所が存在した．さらに置換による提案方式にキャッシュサイズの増加に伴い平均集約度が減少する場合も見られた．平均ヒット率はキャッシュサイズが小さいと LRU 方式が提案方式を大きく上回るが，キャッシュサイズが大きくなるにつれて差が小さくなった．

図 10 図 11 にキャッシュサーバが 8 台の時の結果を同様に示す．キャッシュサーバが増えることでどの方法も平均ヒット率が上昇する．またキャッシュサーバが増えることで平均集約度については全体的に低下傾向にあるが，挿入に共起を考慮したキャッシュ制御の優位性が大きく増加する．

以下，これらの結果について考察する．挿入に共起を考慮した場合は，キャッシュサイズの変化にかかわらずオブジェクト集約度合いが高くなっている．また，平均ヒット率に関しても比較方式と大きな差が表れなかった．この結果から，挿入に共起を考慮することは当初の目標である HTTP/2 や HTTP/3 の効果を高めることを期待できる．

しかし，置換に共起を考慮した際には 2 つの問題点が発生した．問題点一つ目として，平均集約度に着目するとキャッシュサイズが小さい際には提案方式が LRU 方式を下回るが，キャッシュヒット率に大きな差があるためだと考えられる．2 つ目の問題点として正規化キャッシュサイズ 0.018 時点からオブジェクト集約度が低下する点である．ここで評価指標としてオブジェクト集約度合い C_w の分子はキャッシュヒットしたオブジェクト総数の二乗であり，分母はキャッシュヒットしたオブジェクトの総数であるが，キャッシュサイズが増えるごとにオブジェクト集約度合いの分子の伸び率が分母の伸び率よりも小さいことが原因だと考察する．結果としては，キャッシュサイズが小さすぎると，平均ヒット率に大きな差が出るため、オブジェクト集約度の効果が小さくなる．またキャッシュサイズが大きすぎると、キャッシュにアクセスする URL のオブジェクトがほとんど入った状態となるため提案方式の効果が小さくなる．キャッシュサイズ比 0.01 から 0.03 付近だと，提案方式は平均ヒット率は単純 LRU と比較して下回っているが，共起を用いることで平均集約度が単一のオブジェクトに着目した単純な LRU を上回っている．このことから限定的ではあるが，HTTP/2 や HTTP/3 の並列転送効果を高めることを期待できる．

5. ま と め

本稿では，HTTP/2 や HTTP/3 対応キャッシュサーバアクセス時の Web ページの表示待ち時間削減を目的とした，複数のキャッシュサーバで共起を考慮したキャッシュ挿入・置換法を提案し，その集約度を評価した．共起を考慮しない LRU のほうが提案方式のキャッシュヒット率を上回るが，オブジェクト集約度に関しては比較方式よりも提案方式が上回る．挿入のみに共起を考慮すると，キャッシュサイズに関わらず提案方式のオブジェクト集約効果が確認されたが，置換のみに共起を考慮すると一部のキャッシュサイズにおいてのみ提案方式のオブ

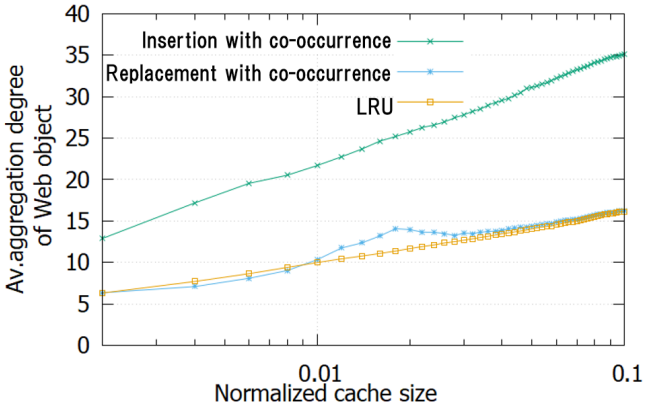

図 8　キャッシュサーバ 4 台時の平均集約度

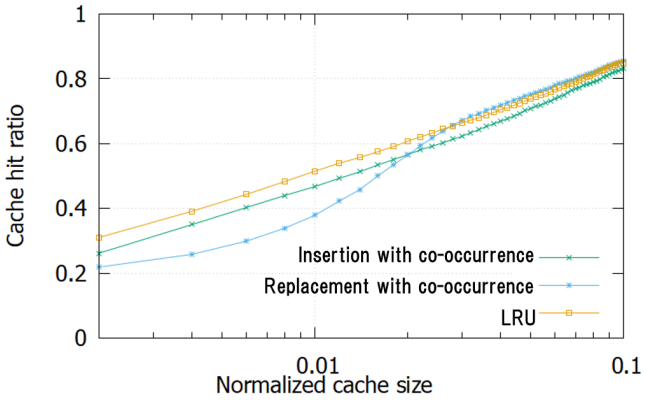

図 9　キャッシュサーバ 4 台時のキャッシュヒット率

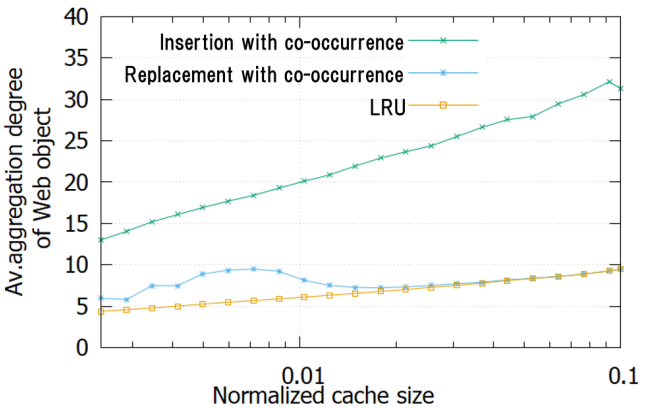

図 10　キャッシュサーバ 8 台時の平均集約度

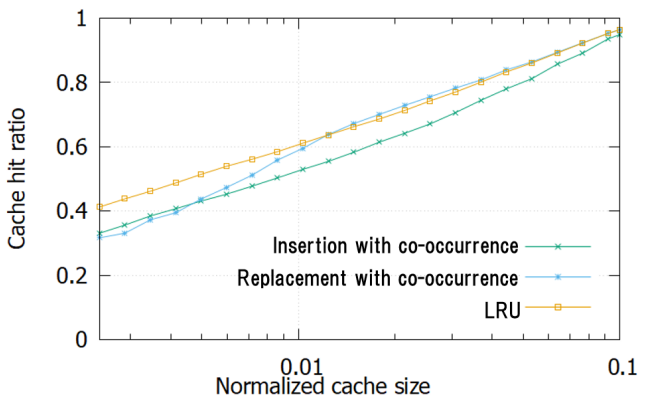

図 11　キャッシュサーバ 8 台時のキャッシュヒット率

ジェクト集約効果が確認された．

今後は，挿入と置換の両方に共起を考慮した際のオブジェクト集約度や，要求頻度の変化が集約度に与える影響を評価する．

謝辞　本研究成果は，KDDI 財団研究助成金 190051 の援助を受けたものである．ここに記して謝意を表す．

文　献

[1] Alexa, https://www.alexa.com/siteinfo
[2] M. Belshe, R. Peon, and M. Thomson, Hypertext Transfer ProtocolVersion 2 (HTTP/2), IETF RFC 7540, May 2015.
[3] M. Butkiewicz, H. V. Madhyastha, and V. Sekar, Understanding Website Complexity: Measurements, Metrics, and Implications, ACM IMC 2011.
[4] R. Fielding and J. Reschke, Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing, IETF RFC 7230, June 2014.
[5] P. Gill, M. Arlitt, N. Carlsson, and A. Mahanti, Characterizing Organizational Use of Web-based Services: Methodology, Challenges,Observations, and Insights, ACM Trans. The Web.
[6] GitHub, Network Monitoring with PhantomJS, http://phantomjs.org/network-monitoring.html
[7] Not as SPDY as You Thought, Guy’ s Pod, Thoughts and research on Web Performance & Secirity, Jun. 2012.
[8] S. Ihm and V. Pal, Towards Understanding Modern Web Traffic, ACM IMC 2011.
[9] M. Jiang, X. Luo, T. Miu, S. Hu, and W. Rao, Are HTTP/2 Servers Ready Yet?, IEEE ICDCS 2017.
[10] N. Kamiyama, Y. Nakano, and K. Shiomoto, Cache Replacement Based on Distance to Origin Servers, IEEE Transactions on Network and Service Management, Vol.13, Issue 4, pp. 848-859, Dec. 2016.
[11] N. Kamiyama, K. Sakurai, and A. Nakao, Measurement Analysis of Co-occurrence Degree of Web Objects, IEEE GI 2021.
[12] J. Manzoor, I. Drago, and R. Sadre, How HTTP/2 is changing Web traffic and how to detect it, TMA 2017.
[13] Software is hard, http://www.softwareishard.com/blog/harviewer/
[14] When seconds count. http://www.gomez.com/wp-content/downloads/GomezWebSpeedSurvey.pdf.
[15] E. Nygren, R. Sitaraman, and J. Sun, The Akamai Network: A Platform for High-Performance Internet Applications, ACM SIGOPS 2010.
[16] J. Ott, M. Sanchez, J. Rula, and F. Bustamante, Content Delivery and the Natural Evolution of DNS, ACM IMC 2012.
[17] A. Su, D. Choffnes, A. Kuzmanovic, and F. Bustamante, Drafting Behind Akamai: Inferring Network Conditions Based on CDN Redirections, ACM Trans. Networking, Vol. 17, No. 6, pp. 1752-1765, Dec. 2009.
[18] X. S. Wang, A. Balasubramanianm A. Krishnamurthy, and D.Wetherall, How speedy is SPDY?, NSDI 2013.
[19] T. Zimmermann, J. Ruth, B. Wolters, and O. Hohlfeld, How HTTP/2 Pushes the Web: An Empirical Study of HTTP/2 Server Push, IFIP Networking 2017.