分散システム／インターネット運用技術 15－3
（1999．9．24）

プロキシサーバの故障を考慮したHRキャッシュクラスタの性能解析

大須賀 圭人　　河合 栄治　　知念 賢一　　山口 英
奈良先端科学技術大学院大学情報科学研究科

概要

本研究は、分散WWWキャッシュシステムの一手法であるHR(Hash Routing)方式のアルゴリズムに変更を加えることにより、故障に対して頑丈な仕組みのDHR(Duplication Hash Routing)方式を提案する。従来のHR方式は、特定のハッシュ関数を用いて、様々なWWWオブジェクトに対するキャッシュサーバを一意に決定する方法である。そのためキャッシュの使用効率は向上するが、プロキシサーバの故障に弱く、外界との通信ができなくなってしまうこともある。この欠点を改善するために、DHR方式ではハッシュ関数を2つ持たせ、わずかなオブジェクトの重複を許容するようにした。

本稿では、提案するDHR方式の設計およびシミュレーションによるその評価について述べる。

キーワード：**DHR方式、キャッシングプロキシサーバ、プロキシサーバの故障**

DHR: Design of a Robust Hash Routing with Cache Duplication

Kadohito Osuga, Eiji Kawai, Ken-ichi Chinen and Suguru Yamaguchi
Graduate School of Information Science,
Nara Institute of Science and Technology.

Abstract

We propose the DHR (Duplication Hash Routing), an algorithm derived from HR (Hash Routing), which has a greater robustness to proxy failures. Conventional HR uses a specific hash function to determine uniquely storage location of objects resulting in efficient use of the cache. However, it is not robust against proxy failure. To avoid this drawback DHR uses two hash functions to increase redundancy by storing objects in two proxies. When the primary proxy fails the secondary proxy is used to retrieve the required object.

In this paper, we describe design of DHR and its evaluation with simulations.

Keywords:DHR, Caching proxy server, Failure in proxy server

1　はじめに

いまやWWW(World Wide Web)は全世界的に使われるようになり、必要不可欠なサービスとなっている。ゆえに、サーバやネットワークの故障は最小限に食い止めなければならない。

特に分散WWWキャッシュシステムではキャッシュの効果を高めるためにプロキシサーバ間での協調が行われている。このような場合、あるプロキシが故障すると、クライアントはオブジェクトを取得できなくなることがある。

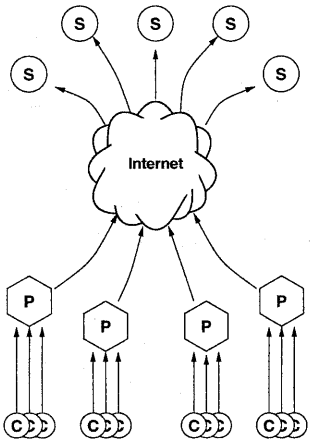

図 1: 一般的なネットワーク

さらに、プロキシサーバ間での協調を行わない場合と比較して、影響を受けるクライアントが多くなる。プロキシサーバの故障を回避する技術として PAC(Proxy Anto Configuration) や Layer4 switch 等が存在するが、分散 WWW キャッシュシステムには対応していない。

本稿では、故障発生時でもキャッシュを利用したオブジェクトの取得を可能にする方式を提案する。本方式は DHR(Duplication Hash Routing) と呼び、プロキシサーバの故障に弱い (理由は 2.1 で述べる) 分散 WWW キャッシュシステムの一手法である HR(Hash Routing) 方式の拡張である。

2 分散 WWW キャッシュシステムの方式とその故障

一般的なネットワークを図 1 に示す。各プロキシサーバは同じ LAN 環境に接続されたクライアントからリクエストを受け取り、処理を行い、オブジェクトを返す。分散 WWW キャッシュシステムでは、各プロキシサーバ間で協調を行う。

分散 WWW キャッシュシステムにおいて考えられる故障としては、クライアントの接続しているプロキシサーバ、リモートのプロキシおよびオリジンサーバ、そして、それぞれの間の 3 種類の中間経路の、合計 6 つの故障が考えられる (図 2)。

本研究では、故障の対象をその問題が最も深刻となるプロキシサーバのみとする。クライアントと同じ LAN に接続されたローカルのプロキシサーバの故障についてはクライアント側で PAC 技術を用いることにより回避し、リモートのプロキシサーバの故障については後述する DHR 方式を用いて回避する。

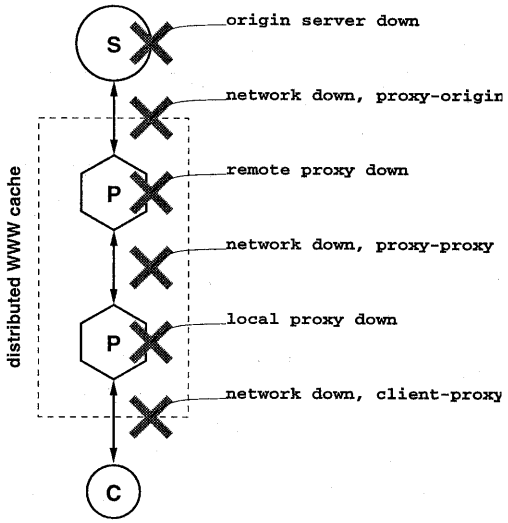

図 2: 考えられる故障箇所

2.1 HR 方式

HR 方式とは、オブジェクトの URL 等から特定のハッシュ関数を用いてハッシュ値を計算し、複数あるプロキシサーバの中から保存するプロキシサーバを一意に決定する方式である [1]。ここでは、プロキシサーバがハッシュ関数を計算すると想定する。

HR 方式を用いることで、キャッシュサーバクラスタ内に複数の同一オブジェクトを保存することがなくなる。よってキャッシュの容量を有効的に使うことができる。しかし、あるプロキシサーバが故障すると、そこに格納されているオブジェクトを取得することができなくなる (以後、これを故障ミスと呼び、その割合を故障ミス率と呼ぶ) という問題点がある。

2.2 PAC 技術

ローカルプロキシサーバの故障のため、外界と通信を行うことができなくなる場合がある。PAC はこのような故障があった場合、自動的

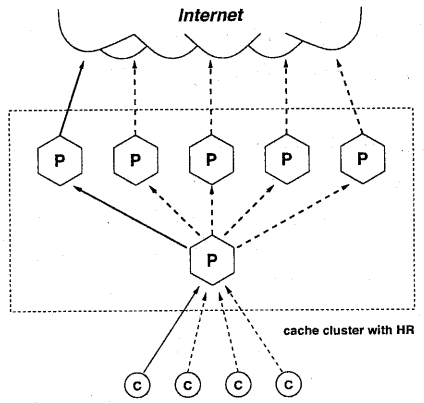

図 3: HR 方式の動作例

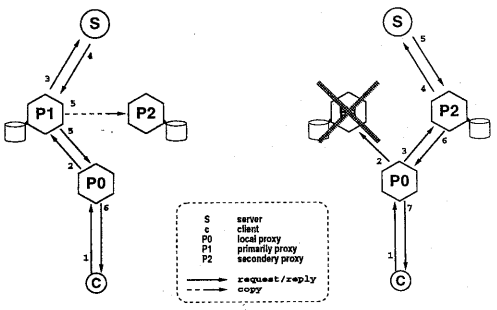

図 4: DHR 方式の動作例

に別のプロキシサーバに処理を依頼する技術である。

3 DHR 方式の提案

前章で述べた HR 方式の問題点を解決するため、オブジェクトの重複を許容することにより分散システム全体の稼働率の向上を目指す手法を提案する。一方、HR 方式の重複を避けるという方針に基づき、可能な限り重複を減らすことも考慮する。以下に提案する DHR 方式の処理の流れを述べる。

通常、クライアントはローカルのプロキシサーバに対しリクエストを送る。ローカルのプロキシサーバはリクエストに対して特定のハッシュ関数 (以後、第 1 ハッシュ関数と呼ぶ) を計算して、そのオブジェクトを保持しているプロキシサーバ (主プロキシと呼ぶ) を求める。そのリクエストが主プロキシでキャッシュヒットしなかった場合は、HR 方式と同様に主プロキシがオリジンサーバからオブジェクトを取得し、そのコピーを保存してから、ローカルプロキシ経由でクライアントに返す。リクエストがキャッシュヒットした場合には、主プロキシはそれをミス時と同様の手順でクライアントに返し、同時に別のハッシュ関数 (第 2 ハッシュ関数と呼ぶ) を計算し、割り当てられたキャッシュを管理するプロキシサーバ (副プロキシと呼ぶ) へオブジェクトをコピーする (図 4)。

クライアントがリクエストを送る際にローカルプロキシが故障していた場合、PAC 技術を使って同じクラスタ内のプロキシサーバにリクエストを送ることにより、処理の代理を依頼する。

主プロキシが故障している場合には、ローカルプロキシは副プロキシを求め、副プロキシに処理の代理を依頼する。このように、故障ミスに対して副プロキシを介して通信することによって故障を回避する (これを故障回避と呼び、その数と副プロキシに処理を依頼した数の比を故障回避率と呼ぶ)。また、この後の動きは、キャッシュヒットが起こってもオブジェクトのコピーを行わないこと以外は、故障のないときの処理と同様である。

DHR 方式においては、ヒットしなかったオブジェクトの重複した保存は行わない。その理由は、ほとんどのオブジェクトは一回しか参照されないという事実 (Zipf の法則) [2] があり、これらのオブジェクトを複数のプロキシ間で共有して持つことは無駄だからである。逆に言えば、キャッシュヒットしたオブジェクトというのは、少なくとも 2 回はリクエストのあったオブジェクトなので、再利用の可能性が高いと考え、これらの重複は行う。

また、キャッシュヒットする度に副プロキシにオブジェクトを転送するのはトラヒックの無駄であるため、一度コピーを行った後はある条件の時だけコピーを行う。その方法としては、ヒットによりコピーを行ったオブジェクトがリクエスト列の t 番目のものであったとすると、その後は $t+T$(Tは再送周期) を越えた後にヒットしたときにのみ副プロキシにコピーすることで、トラヒックの増加を抑制する。

4 DHR 方式の性能予測

DHR 方式を導入したプロキシサーバの故障に対する性能を予測するために、シミュレーションを行った。また、リクエストは一日に 10 万件発生するものとし、その一年分のリクエスト列を用いてシミュレーションを行う。

4.1 実験手順

最初に参照頻度、オブジェクトサイズの 2 つのパラメータを持つオブジェクトに対するリクエスト列を生成した。それを 4.3節で述べるようなサーバクラスタに与えることにより、プロキシサーバの挙動を観測した。

4.2 オブジェクトのパラメータ

オブジェクトの参照頻度 fは Zipf の法則に従うことが知られている [2]。その式を以下に示す。ここで、rは参照回数のランキング、k, Cは定数である。

$$f = \frac{C}{r^k} \tag{1}$$

また、WWW で取得されるそれぞれのオブジェクトのサイズ x は対数正規分布に従う [3]。以下にその確率分布関数と確率密度関数を示す。

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma x} \exp\left[\frac{-(\log x - \mu)^2}{2\sigma^2}\right]$$

$$F(t) = \int_0^t \frac{1}{\sqrt{2\pi}\sigma x} \exp\left[\frac{-(\log x - \mu)^2}{2\sigma^2}\right] dx$$

本実験では、文献 [4] を参照し、$k = 0.66, \mu = 7.3, \sigma = 1.7$ を用いた。また、本実験では $C = 50,000$ を用いた。この値を用いると、総リクエスト数は 33,529,424 となり、この数は一年分のリクエスト数 (3650 万件) に近い。

4.3 プロキシサーバのパラメータ

本実験では、プロキシサーバはキャッシュ置換アルゴリズムには LRU(Least Recently Used) を用い、キャッシュ容量はすべて同一とした。キャッシュ容量は 512Mbyte および 1Gbyte、プロキシサーバの台数は 1,2,4,8,16 の場合をシミュレートした。

4.4 故障のパラメータ

故障の起きる間隔の平均時間 ($MTBF$:Meen Time Between Failure) および、故障から回復するのにかかる時間の平均 ($MTTR$:Mean Time To Repare) を決めることにより故障率 $MTTR/(MTTR + MTBF)$ が決定する。本実験では、故障が一年間に約 7 回起こり、一回の故障の回復に一日かかるような環境を想定した。このときの故障率は約 2%である。

この故障パラメータは、プロキシサーバがハッシュ関数の計算結果の対象となるプロキシサーバにオブジェクトの取得を依頼する際に影響する。その対象となるシブリングが故障している場合、HR 方式ではクライアントにエラーを返すが、DHR 方式では副プロキシからオブジェクトを取得する。

4.5 ハッシュ関数

本実験では、第 1 ハッシュ関数として以下の式を用いた。

$$h_1(O) = O \bmod N$$

ただし、Oはリクエストされたオブジェクトの URL を 10 進数化したもので、Nはプロキシサーバの数であり、各プロキシサーバには 0 から $N-1$ の番号が順番につけられているものとする。

また、第 2 ハッシュ関数には以下の式を用いた。

$$H = O \bmod (N-1)$$

$$h_2(O) = H + f(h_1(O), H)$$

$$f(h_1(O), H) = \begin{cases} 0 & (H < h_1(O)) \\ 1 & (H \geq h_1(O)) \end{cases}$$

4.6 コピーの実行における再送周期

ここでは 3 章で述べた再送周期 Tの決め方について述べる。容量が A Bytes のキャッシュに、内容がすべて異なるリクエストが転送されることを考える。このとき、オブジェクトの平均サイズが 10 Kbytes だとすると、キャッシュがそのリクエスト列から放出されたもので埋め尽くされるまでのリクエスト通過数は $A/10K$ となる。言いかえれば、キャッシュに蓄えられ

たばかりのオブジェクトは、その後 $A/10K$ 個のオブジェクトが通過するまではキャッシュから取り除かれることはない。よって再送周期を以下のように決定した。

$$T = \frac{A}{10K} \tag{2}$$

4.7 PACのアルゴリズム

PACのアルゴリズムとしては、ローカルプロキシ k(4.5節で付けた番号) が故障していた場合は、プロキシ $k+1, k+2$ と、k以外のプロキシへと処理を依頼する。すべてのプロキシが故障している場合には、クライアントにエラーを返すように設定した。HR方式でのリモートプロキシ、またはDHR方式での副プロキシが故障している場合も同様に、クライアントへエラーを返す。

4.8 測定項目

1) DHR方式の利点である故障回避率の向上度を調べるために、HR方式、DHR方式の故障ミス率を測定する。

2) DHR方式の欠点であるプロキシ間のトラヒック量の増加率を調べるために、HR方式、DHR方式のプロキシ間のトラヒック量を測定する。

3) DHR導入によるヒット率の変化を調べるために、HR方式、DHR方式のヒット率を測定する。

5 実験結果と考察

4.8節の 1) ～ 3) の測定結果をそれぞれ図5 ～ 図7 に示す。キャッシュサイズに関しては512MBytes、1Gbyteのどちらの場合でも類似した曲線になることから、1Gbyteの場合のみを掲載した。

図5より、DHR採用の結果、故障ミスが無くなっているのが分かる。ただし、プロキシサーバが1つの場合には代わりとなるプロキシがないため、故障ミスが発生する。これより、オブジェクトを取得できない状況をなくすという、本研究における目的を達成できたことがわかる。DHR方式の性質から、故障ミスが起こるのは2つ以上のプロキシサーバが同時に故障し、かつそれらのうちの2つが主プロキシと副プロキシに相当する場合である。本実験では4.4節で述べたように故障率を2%としているので、プロキシサーバの台数が16台であっても、2つが同時に故障している確率は非常に小さい。よって、故障率をもっと高くし、プロキシサーバの数を増やせば、DHR方式でも故障ミスが起こるであろうが、HR方式の故障ミス率も同様に増加することが予測できる。

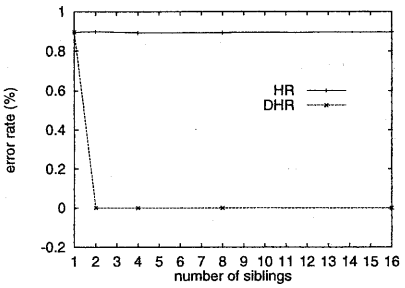

図 5: 故障ミス特性

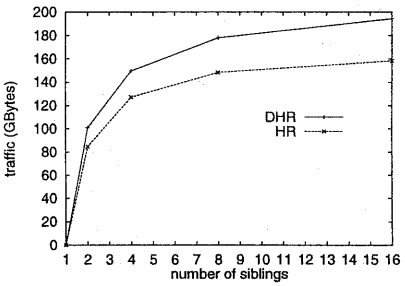

図 6: プロキシ間のトラヒック特性

図6はDHR方式でのプロキシ間のトラヒック量をHR方式のトラヒック量と比較したものであり、どの点を比較しても2割程度増加している。また、図7から、DHR導入によるヒット率の低下は1割以下であるのが分かった。すなわち、トラヒック2割の増加とヒット率1割の低下で故障ミス率を劇的に削減することを意味する。このトラヒックの増加は副プロキシへのコピーのため、またヒット率の低下はそのコピーの保存でキャッシュのディスク領域を使用

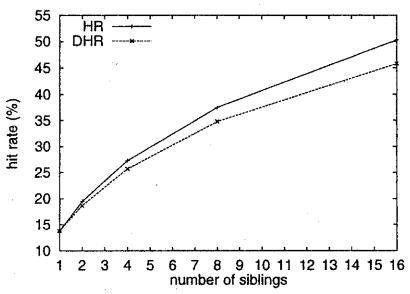

図 7: ヒット率特性

したために生じたものである。これらに対する解決策として、今回はキャッシュヒットしたものを無条件にコピーしていたのに対し、ある回数 (以後、コピー開始数と呼ぶ) 以上ヒットしたオブジェクトのみをコピーする、あるいは再送周期 Tの値を長くすることが考えられる。

本実験では、リクエスト数、ディスク容量、故障率に関するパラメータは固定して行った。これらのパラメータを変えると当然ヒット率特性やトラヒック特性の値も変わるが、曲線の形としては図5 ～ 図7 の形を保つ。したがって、これ以外の環境でも DHR 方式は HR 方式に対する優位性を保つ。

6 議論

今回のシミュレーションでは、プロキシサーバの故障に対するクライアント側の対処として PAC 技術を用いた。PAC 技術ではなく Layer4 switch を用いた場合のシミュレーションも必要であろう。また、実際の WWW では、異なるプロキシに所属するクライアント間でのオブジェクトの人気に片寄りがある。また、置換アルゴリズムは LRU でない、あるいは LRU を近似した実装であるため、本実験で用いた LRU とは異なっていることが考えられる。また、リロードボタンの使用やオブジェクトの追加、更新、削除により、実際のヒット率はもっと低い。本論文の結果を運用の参考にするためにはこれらの点を考慮する必要がある。

7 おわりに

本稿では、代表的な分散キャッシュの構築法である HR 方式のプロキシサーバの故障に対する弱さを克服するため、2 つのハッシュ関数を用意し、かつクラスタサーバ内でわずかなオブジェクトの重複を許す DHR 方式の提案を行った。そしてシミュレーションにより DHR 方式の性能を HR 方式と比較した。プロキシサーバ間でのトラヒック量やヒット率をわずかに犠牲にすることにより、故障回避という目的を達成できた。これらの犠牲をもっと小さくするために最適な再送周期、およびコピー開始数を決定するような研究を行うことが今後の課題である。

謝辞

本研究に対して助言をくださった、奈良先端科学技術大学院大学の情報ネットワーク講座および情報科学センターの皆様に感謝致します。また、色々な面においてご指導頂きました、九州工業大学情報工学部電子情報工学科の尾家祐二教授に深く感謝致します。

参考文献

[1] K. W. Ross, "Hash Routing for Collections of Shared WebCaches", *IEEE Network* November/December 1997

[2] L. Breslau, P. Cao, L. Fan, G. Phillips and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications", *IEEE Infocom*,New York, March 1999

[3] 名部正彦, 馬場健一, 村田正幸, 宮原秀夫, "WWW トラヒック特性を考慮したアクセスネットワークの設計に関する検討", 電子情報通信学会 進学会技報 SSE96-141,CQ96-51,pp.73-78,Dec1996

[4] 知念賢一, "Internet における大規模情報提供と情報取得の高速化に関する研究", 奈良先端科学技術大学院大学 博士論文 NAIST-DT9561026