

分散 WWW キャッシュのパフォーマンスに関する因果分析

内藤広志

大阪工業大学 情報科学部 情報システム科

本稿では、WWW アクセスの高速化のために学内ネットワーク上で分散 WWW キャッシュを運用した結果として得られたアクセスログを元に、分散キャッシュのパフォーマンスに関する因果分析を行った結果を述べる。因果分析の対象は、提案した分散キャッシュの評価モデル上の変数、アクセスログから得られる観測変数、および分散キャッシュのパラメータで、それらの間の線形または非線形関係を統計的な手法を用いて分析した。

Causal Analysis of Performance of Distributed WWW Cache

Hiroshi Naito

Faculty of Information Science, Osaka Institute of Technology

This paper discusses causal analysis of performance of distributed WWW cache based on access logs of WWW cache running on educational network.. Causal relationship between variables on proposed evaluation model for WWW cache, observed variables retrieved form access log, and parameters of WWW cache is analyzed using statistics methods.

1. はじめに

大阪工業大学の情報科学部では全学生にパスワードを与えており、自由にネットワークを利用できる環境がある。学内 LAN は、ATM(155Mbps)やFDDI(100Mbps)で構築されており高速であるが、インターネットとの接続は低速であり、両者の帯域に大きな差がある。このため、1996年よりネットワーク利用の中心がWWWに変化し、ネットワークトラフィックが顕著的に拡大すると、電子メールの利用にも支障をきたすようになった。この限られたネットワーク帯域を有効に活用するため、1996年11月より、WWW オブジェクトのキャッシュを行うキャッシュプロキシを複数個立ち上げ、分散キャッシュシステムの運用を開始した。分散キャッシュの構築はアクセスログなどを参考にして手作業で行っているが、分散キャッシュが大きくなると、それを自動化する必要がある。そのために、単なるヒューリスティックではなく、パフォーマンスに関する評価モデルが必要である。また、評価モデルは、分散キャッシュのボトルネックを分析するためにも必要である。

本稿では、分散キャッシュの評価モデルを提案するとともに、分散キャッシュのアクセスログを統計的手法を用いて分析することで、評価モデルの変数と分散キャッシュのパラメータ間の因果関係について考察する。

2. キャッシュプロキシについて

Proxy サーバは、クライアントとサーバ間で交わされる応用プロトコルを中継するソフトウェアである。Proxy サーバのうち、オリジナルサーバに格納されているWWW オブジェクトをキャッシングし、それをクライアントに提供するものをキャッシュプロキシ（以下キャッシュと略す）と呼ぶ。

キャッシュプロキシには、HTTP サーバを拡張したもの（例えば、CERN httpd）、漢字コード変換などの様々な処理を実現する汎用プロキシをベースにしたもの（例えば、DeleGate）、参照頻度の高いオブジェクトをプリフェッチするもの（例えば、Wool）があるが、本実験で使用したキャッシュプロキシの squid[1]は、WWW オブジェクトのキャッシュの有無をキャッシュ間で問い合わせるプロトコル、ICP(Internet Cache Protocol)を使用し、

分散して配置されたキャッシュ間で連携を行うものである。

2.1. Squid の機能

Squid は、Harvest プロジェクトで開発されたCache[2]をベースに拡張されたフリーウェアで、NLANR(National Laboratory for Applied Network Research)を中心にインターネット上で開発されている[3]。

Squid は次の機能を持つ。

- ① http, ftp, gopher, wais のオブジェクトをキャッシュ。
- ② ICP を使用した分散キャッシュのサポート。
- ③ ファイヤーウォールのサポート。
- ④ ドメイン名やURLに基づくアクセス制御。
- ⑤ 2 階層のキャッシュシステム（メモリおよびディスク(UFS))によるキャッシュの高速化。

2.2. ICP の特徴

squid で用いている ICP は、Harvest プロジェクトの Cache 用に最初に開発(1994 年)され[2]、現在は Internet-Draft となっている[4]。

ICP は次の特徴がある。

- ① キャッシュ間の通信を規定するプロトコルである。
- ② キャッシュオブジェクトの有無を問い合わせる。
- ③ 実際のオブジェクトの取得はHTTPを使用する。
- ④ HTTP の場合、GET 要求だけをキャッシュする。
- ⑤ UDP ベースの軽量のプロトコルである。

ICP の代表的な命令に、指定 URL のキャッシュオブジェクトの問合せをする ICP_OP_QUERY、キャッシュオブジェクトの存在の返答である ICP_OP_HIT、キャッシュオブジェクトの非存在の返答である ICP_OP_MISS がある。

2.3. squid のキャッシュ選択アルゴリズム

ICP を用いて、連携を取る複数のキャッシュ(neighbor と呼ぶ)から一つを選択するアルゴリズムは次のとおりである[1]。

- ① クライアントから HTTP 要求(ftp, wais, gopher も含む)がきたとき、キャッシュオブジェクトがなければすべての neighbor に対して ICP_QUERY を送る。
- ② すべての neighbor が ICP_HIT または ICP_MISS を返すまで待つ。2秒以内に返答がなければエラーとする。
- ③ 返答が ICP_HIT の場合、その返答を返した neighbor

に HTTP 要求を送る。複数の neighbor が ICP_HIT を返答したときは、RTT が最小の neighbor に要求する。

④ 返答がすべて ICP_MISS の場合は、次の neighbor のタイプによって処理が異なる。

- (a) sibling : 何もしない
- (b) parent : HTTP 要求を発行する。

24. squid の構成/パラメータ

squid.conf に書かれたパラメータを変更することで、squid の構成を変更できる。次にキャッシュ効率に関係する主要なパラメータを説明する。

(1) 分散キャッシュの構成を記述するパラメータ

- cache_host : neighbor として使用するキャッシュホストを定義する。
- cache_host_domain : キャッシュホスト毎に ICP 問合せを行うドメインを指定する。
- neighbor_type_domain : キャッシュホスト毎に neighbor のタイプとドメインを変更する。

(2) キャッシュの使用資源に関するパラメータ

- cache_mem : オンメモリキャッシュのサイズの指定。
- cache_swap : キャッシュ用ディスク容量の指定。

(3) キャッシュの使用資源に関するパラメータ

- maximum_object_size : キャッシュに格納するオブジェクトの最大サイズの指定。
- refresh_pattern : キャッシュオブジェクトが最新状態とする期間を指定。

3. 分散キャッシュの運用実験

3.1. 運用システムの構成

分散キャッシュの運用実験を行った大阪工業大学は、図1のように大宮と枚方のキャンパスに分かれており、両者は512Kbpsの専用線で接続されている。また、インターネットとは128Kbpsの専用回線を介して接続している。学内LANはFDDIやATMを用いているため高速であるが、それに対してインターネットとの接続が低速であるという問題点がある。そのため、大宮と枚方のキャンパスに複数のsquidを立ち上げ、WWWの利用はsquidを介してだけ可能としている。

大宮キャンパスにはルートキャッシュとなるPROXYAを立ち上げ、各学科用のキャッシュを立ち上げている。一方、枚方キャンパスには教員用キャッシュの

PROXYBと学生用キャッシュのPROXYCを立ち上げている。これ以外にも各学科用のキャッシュが実行されているが、運用実験の対象としたキャッシュはPROXYA、PROXYBおよびPROXYCである。それぞれのシステム構成は図2に書かれているとおりである。但し、PROXYAのキャッシュディスク容量は、1997年4月14日までは1Mbで、4月15日に2Mbに増強し、更に5月3日には3Mbに増強した。

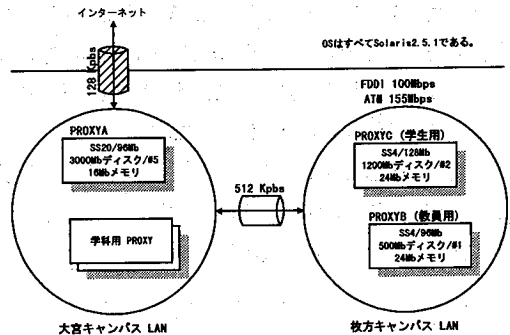


図1 大阪工業大学のキャンパスネットワーク概略図

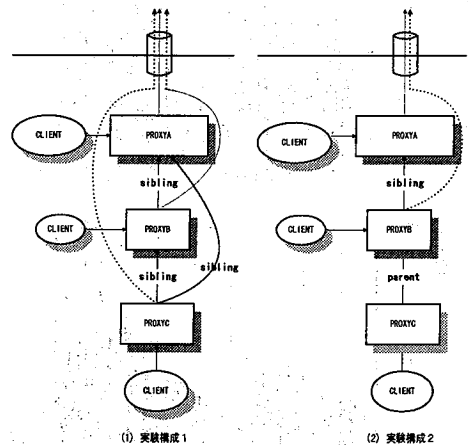


図2 分散キャッシュの構成

squidを使用すると、様々な構成の分散キャッシュが可能であるが、本研究に先だって実験を行った結果[7,8]、図2の実験構成(1)に示す分散キャッシュを採用した。その主要な理由は次のとおりである。

① イン트라ネット内で無駄なトラフィックを発生させないために、ネットワークトポロジーにそってparent/sibling関係を定義する。

② キャッシュディスクの容量の大小関係に従って、parent/sibling 関係を定義する。ICP 問合せに対するヒット率に影響する最大の要因はキャッシュディスクの大きさであり、子キャッシュより少ないディスク容量しか無い親キャッシュに ICP 問合せをしてもヒットしないからである。

運用実験は、1997年4月1日より開始したが、学内にファイアウォールを設けることとなり、インターネットにアクセスするキャッシュを PROXYA と PROXYB に限定することに変更した。そのため、図2の実験構成(2)にキャッシュの構造を変更し、あわせて PROXYB のキャッシュディスク容量を 1.5GB に増強した。

また、squid は頻りにアップデートされているが、実験の精度を高めるため、実験に使用するバージョンを 1.1.15 に統一し、期間中は更新しなかった。

3.2. 運用システムの利用状況

squid はクライアントからアクセスされるたびに、アクセス時間、転送バイト数、転送時間、キャッシュヒットの有無などのログをファイルに保存する。実験の期間中は、PROXYA、PROXYB、PROXYC のアクセスログを一日を単位として収集し、ファイルに保存した。このアクセスログを解析し、毎日の要約データを作成し、それを統計解析ソフトウェア SPASS によって分析した。

要約データを作成するためのツールは calamaris[5] を改良したもので、転送速度、エラー発生率、sibling 毎のヒット率などの要約情報を出力した。

実験期間中の各キャッシュの利用状況を示すために、図3に転送バイト数、要求クライアント数、HTTP の要求回数、バイトヒット率の時系列グラフを示す。

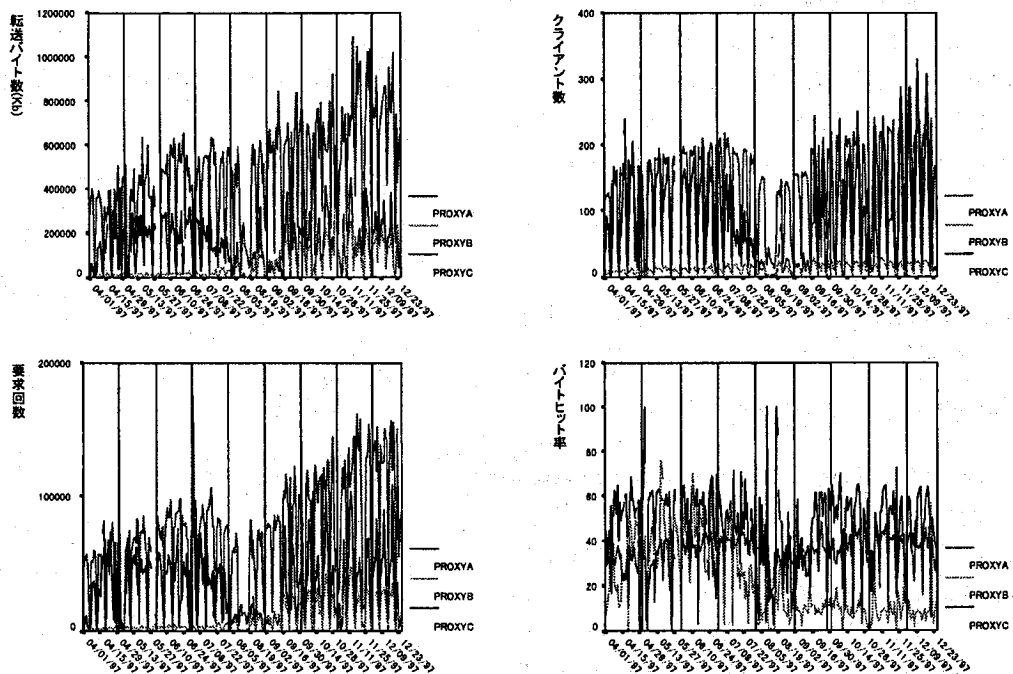


図3 キャッシュの利用状況

4. キャッシュの評価モデル

4.1. キャッシュ有効性の評価基準

従来、キャッシュの効果はヒット率だけで議論されることが多かったが、ヒット率が高くと、レスポンスタイ

ムが悪いと、ユーザがキャッシュを利用しなくなるので、レスポンスタイムも含めて評価する必要がある。また、ファイアウォールを構築している環境では、キャッシュがダウンすると全てのユーザがWWWにアクセス出来なくなるので、分散キャッシュの高信頼性も重要である。そのた

め、(1) ヒット率 (2) レスポンスタイム、(3) 信頼性の3つの観点から分散キャッシュを評価すべきである。

4.2 キャッシュの評価モデル

転送バイト数 (B), 転送時間 (T), 転送速度 (R) の間には次の関係がある。

$$R = \frac{B}{T}$$

キャッシュがクライアントに転送する総バイト数を B_{total} とし、キャッシュにヒットしたオブジェクトのバイト数を B_{cached} とし、イントラネット上の WWW サーバから取得したバイト数を B_{local} とし、インターネット上のサーバのオブジェクトを取得したバイト数を $B_{fetched}$ とすると、次の関係が成り立つ。但し、イントラネット上の WWW サーバのオブジェクトは、そのサーバから直接取得すると仮定している。

$$B_{total} = B_{cached} + B_{local} + B_{fetched}$$

更に、 $B_{fetched}$ は、neighbor のキャッシュにヒットして取得したバイト数を $B_{neighbor}$ とし、parent を介して取得したバイト数を B_{parent} とし、直接 WWW サーバから取得したバイト数を B_{direct} とすると、次の関係が成り立つ。

$$B_{fetched} = B_{neighbor} + B_{parent} + B_{direct}$$

分散キャッシュの構造によって、これらの値は大きく異なる。水平構造の場合は B_{parent} は 0 となり、階層構造のリーフキャッシュの場合は、 B_{direct} は 0 となる。

キャッシュ自身のヒット率 H_{cached} と、neighbor に問い合わせた時のヒット率 $H_{neighbor}$ は次のとおりである。

$$H_{cached} = \frac{B_{cached}}{B_{total}}$$

$$H_{neighbor} = \frac{B_{neighbor}}{B_{fetched}}$$

転送時間に関しても、どのような関係が成り立つ。但し、 T_{query} は neighbor に ICP 問合せを行うのにかった時間を表す。

$$T_{total} = T_{cached} + T_{local} + T_{fetched}$$

$$T_{fetched} = T_{query} + T_{neighbor} + T_{parent} + T_{direct}$$

前節で指摘したキャッシュの有効性の問題は、転送速度 R を最大にする条件を求める問題と考えられる。R を最大とするには、転送バイト数 B と転送時間 T の両方を考慮しなければならないが、本報告では B を中心に分析し、

以降の節で B の大小に影響を与えるヒット率について考察する。

5. ヒット率の因果関係

前節で述べたようにヒット率とは、クライアントの要求に対するヒット率 H_{cached} と neighbor への ICP 問合せに対するヒット率 $H_{neighbor}$ がある。運用実験で得られたアクセスログデータを元、これらのヒット率に影響を与える要因を分析した結果を次に述べる。

5.1. キャッシュヒット率の因果関係

ヒット率に影響するものとして、キャッシュへの HTTP 要求に関する、転送バイト、クライアント数、要求回数およびバイトヒット率を考慮される。これらの観測変数の相関を示す例として、PROXYB におけるこれらの変数の行列散布図を図 4 に示す。土日祭日とそれ以外の日では観測変数の傾向に差があるので、両者を分けて散布図を示している。この図を見てもわかるように、転送バイト数、クライアント数および要求回数の間には線形関係がある。特に転送バイト数と要求回数との間には強い線形関係がある。つまり、要求回数が増えると転送バイト数が増すという関係である。このことは、SPSS を用いた回帰分析でも確認できる。同様な関係は PROXYA の回帰分析でも確認できる。

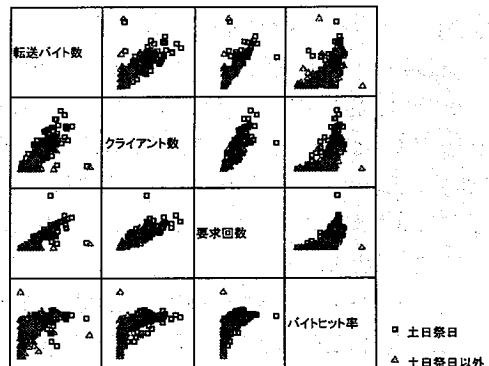


図4 主要な変数間の関係

一方、バイトヒット率と他の3つの観測変数との間には、明らかな線形関係は観測されない。SPSS の曲線推定を用いて、これらの観測変数とバイトヒット率との間を分析すると、バイトヒット率とクライアント数との間

の相関が一番強いことが半明した。PROXYC のデータに対して、両者の曲線推定した結果をグラフとして表したものを図 5 に示す。この図には、観測数と線形関数の他に、曲線推定の当てはまりの高い上位 3 つの関数を上げている。特に対数関数による近似が当てはまる。

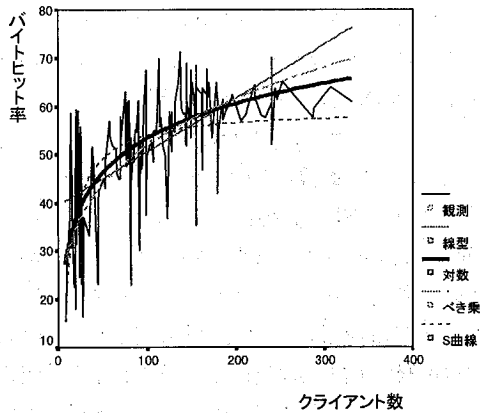


図 5 クライアント数とバイトヒット率の関係

一般に、キャッシュディスクの容量を増やすと、バイトヒット率は増加することが観測される。実験期間中、PROXYA のディスク容量を 2 回上げたが、その結果としてヒット率が上昇した。図 6 はその関係を示す箱ひげ図である。PROXYA のデータだけでなく、PROXYB および PROXYC のデータも参考として列挙した。

同じキャッシュの場合、ディスク容量を上げるとバイトヒット率が増加しているが、他のキャッシュも比べると、バイトヒット率とディスク容量の間に相関があるとは断言できない。クライアント毎の要求回数のばらつきなど、ディスク容量以外の要因がバイトヒット率に影響を与えていると考えられる。

これまで述べた関係をまとめたものが図 7 である。図中の長方形は観測変数を表し、角丸長方形は squid のパラメータを表す。図中の矢印を結ぶ線は、観測変数およびパラメータ間の因果関係を表す。線の上の文字は関係を表し、“+”はプラスの線形関係を、“-”はマイナスの線形関係を、“**”はプラスの非線形関係を、“%”はマイナスの非線形関係をそれぞれ表す。また、“++”は“+”より強い関係があることを意味する。

図 7 には、maximum_object_size および

refresh_pattern という squid のパラメータも上げてあるが、これらとヒット率との間にも相関があると想像されるが、その関係は分析できなかった。

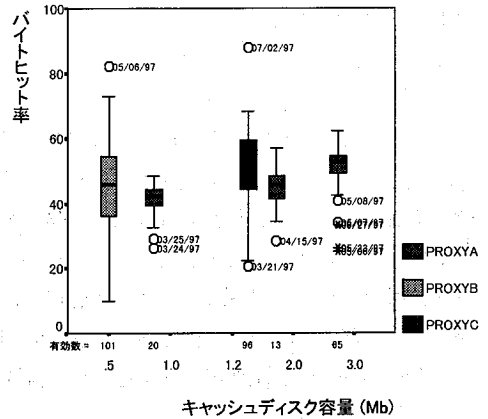


図 6 ディスク容量とバイトヒット率の関係

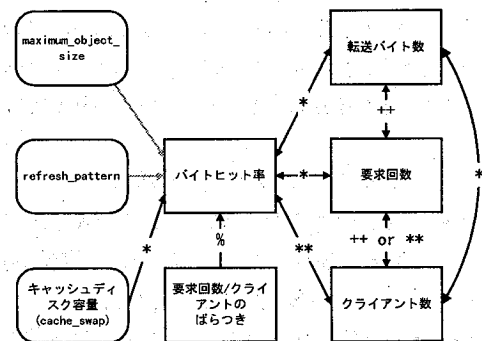


図 7 キャッシュヒット率と他の要因の因果関係

5.2 ICP ヒット率の因果関係

図 8 に、UDP 要求の回数、ヒット率および要求キャッシュの数の時系列グラフを示す。8 月 1 日より分散キャッシュの構成を図 2(b)に変更したため、その日より以降は PROXYC の UDP 要求回数が 0 となっている。また、同時期に PROXYB のキャッシュディスク容量を増加したこともあり、PROXYB の UDP ヒット率が大きくなった。

ICP に関する主要な観測変数の相関を示す例として、PROXYA の観測変数の行列散布図を図 9 に示す。土日祭日とそれ以外の日では、観測変数の傾向に差があるので、両者を分けて散布図に示している。また、この図でバイトヒット率の代わりに要求回数ヒット率を使用している。

それは、squidのパラメータを変更することによって、ヒットしたキャッシュオブジェクトをICPの返答パケットに載せてクライアントに送ることができるため、ICPの転送バイト数の分散が大きくなるためである。図9の転送バイト数と要求回数ヒット率の散布図を見ると、転送バイト数の分散が大きいきちがわかる。TCPの転送バイト数と要求回数の間にお互い線形関係があるので、この関係を利用すると、ICP問合せがヒットしたことでparentキャッシュから転送されたオブジェクトのバイト数 $B_{neighbor}$ のトータルをUDP回数ヒット率から推定できる。

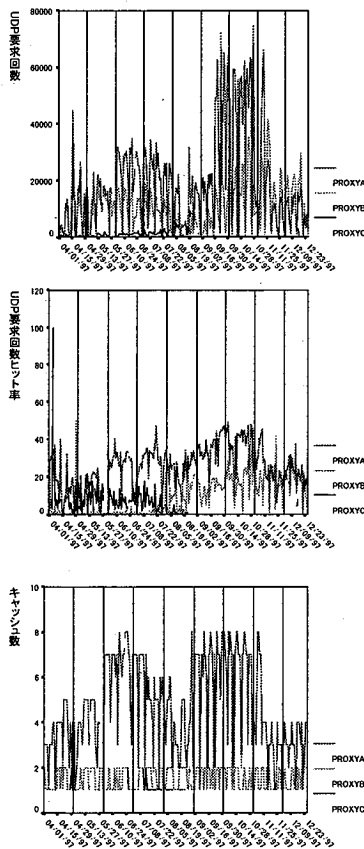


図8 ICP 要求の遷移

UDP 要求回数ヒット率はICPを利用した時だけに見られるものなので、UDP 要求回数ヒット率がICPを用いた効果を示していると言える。ただ、UDP 要求回数はTCP 要求回数に比べて小さいので、学内LANのトラフィック全体に占めるICPの効果は高くない。学科毎や教

室毎にキャッシュを設けることによって、ルートキャッシュの負荷を減らし、分散キャッシュ全体のパフォーマンスを上げることができる。

アクセスログの解析では、キャッシュの数が少ないため、明確な因果関係を得ることが出来なかった。本研究に先だって行った実験では、親と子のキャッシュにおけるキャッシュディスクの容量の大小関係が、ICP ヒット率に最も影響する要因であることがわかった。これなどを参考にすると、図10に示す因果関係がICP ヒット率に関連する要因間にあると予想される。特に「neighbor と自分との要求回数の差」や「neighbor と自分との転送バイトの差」が大きく影響すると考えられる。

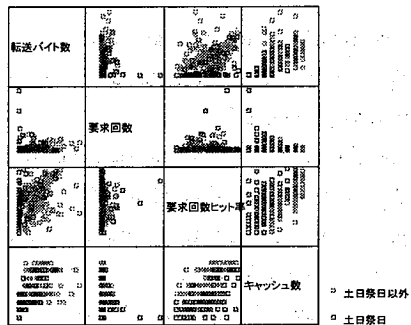


図9 ICP の解析量の相関

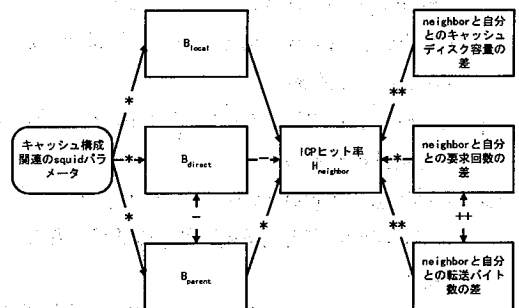


図10 ICP ヒット率の因果関係

6. おわりに

本稿では、バイトヒット率に関連する要因間の因果分析を行ったが、4章で述べたように分散キャッシュのパフォーマンスは転送速度 R によって評価しなければならぬ。そのためには、転送バイト数 B だけでなく、転送時

間T についても因果分析をする必要がある。また、本実験の対象となった大阪工業大学のキャンパスネットワークは、イントラネットの帯域と比べてインターネットへの接続大帯域が極めて低いという特殊な条件にある。本稿で得られた因果関係が一般的なものを検証するためには、他の条件での実験を行う必要がある。これらを今後の課題として研究を進めると共に、分散キャッシュの自動構成およびボトルネック解析に利用できるように評価モデルの検討を続ける予定である。

参考文献

- [1] Wessels, D., "ICP and the Squid Web Cache", <http://www.nlanr.net/~7ewessels/Papers/icp-squid.ps.gz> (Jan97).
 - [2] Chankhunthod, A., Schwartz, M., "A Hierarchical Internet Object Cache", USENIX '96, <ftp://ftp.cs.colorado.edu/pub/techreports/schwartz/HarvestCache.ps.Z> (Oct95).
 - [3] NLANR, "Squid Internet Object Cache", <http://squid.nlanr.net/Squid/> (Mar97).
 - [4] Wessels, D., Claffy, K., "RFC 2186: Internet Cache Protocol (ICP), version 2", IETF Internet-Draft, <http://ds.internic.net/internet-drafts/draft-wessels-icp-v2-03.txt> (Sep97).
 - [5] Beermann, C., "Tools for Squid", <http://www.fh-lippe.de/~cord/tools/>
 - [6] 内藤広志 "squid を用いた学内 proxy 網の構築", 第8回メディア統合技術研究会 画像電子学会 1997.
- 内藤広志, "squid による分散 WWW cache の有効利用法", '97 大規模ネットワーク管理シンポジウム, 日本 UNIX ユーザ会, 1997.