キャッシュ手法をもつ P2P 複製配置に対する性能比較

2004MT022　林裕之　2004MT092　澤木俊希

指導教員　河野浩之

1. はじめに

P2P の複製配置に関する研究として[3]では，FIFO (First-in First-out) ，LFU (Least Frequency Used) ，LRU (Least Recentry Used) の 3 種類のキャッシュ置換アルゴリズムを用いて最適複製配置を行った結果 LRU が最も良いという結論に至った．しかし Zipf 指数を変化させたところ LRU が使用したネットワーク帯域幅の値が最適値に最も近くなった．一方で，ファイルリクエストに対する複製ファイル数を比較すると LRU は最適値に最も近いキャッシュ置換アルゴリズムではなかった．

以上より，本研究では P2P ネットワークにおいて複製配置を行う際に各ピアがキャッシュ置換アルゴリズムによって，理想値に近付く比例する複製ファイル数を実現するキャッシュ置換アルゴリズムは何かを考察する．キャッシュ置換アルゴリズムは，先行研究[3]で用いていた FIFO, LFU, LRU の 3 種類に加え，LRU の派生系 HLRU (History LRU) ，LRU-SLFR (LRU-Small Latency First Replacement) を用いてシミュレーションし，評価を行う．

2. P2P の複製における関連研究[3]

Tewari らが提案したピュア型を使用しキャッシュ置換アルゴリズムを用いた複製ファイルの最適配置について紹介する．

[3]では，記憶容量の限界を考慮して FIFO, LFU, LRU の 3 種類のキャッシュ置換アルゴリズムを用いて最適配置を行っている．FIFO とは複製されたファイル順に削除する先入れ先出しアルゴリズムである．LFU とは使用頻度の少ない複製ファイル順に削除するアルゴリズムである．また，LRU とは過去最も長い期間使われていない複製ファイル順に削除するアルゴリズムである．

これらのキャッシュ置換アルゴリズムを用いてファイルリクエストレートに対する複製ファイル数と Zipf 指数の変化に対する使われたネットワーク帯域幅を求めた比較という 2 つのシミュレーションを行っている．この 2 つのシミュレーション結果から LRU が一番良いという結果が示されている．

3. LRU 派生キャッシュ置換アルゴリズム

3.1. HLRU アルゴリズム[1]

HLRU とはキャッシュ内のファイルへの要求の履歴を考えるLRUアルゴリズムの一種である．主な概念はキャッシュ内のファイルへの過去の参照数の記録をとることである．History LRU は HLRU (*h*) と表記され，*h* は過去からの履歴を表す．

・定義 (history function)

$r_1, r_2, \ldots, r_n$ は時間 $t_1, t_2, \ldots, t_n$ で記録され，キャッシュされたファイルの要求を表す．あるキャッシュされたファイルの履歴関数は (1) 式のように定義する．

$$hist(x,h) = t_i \quad \text{or} \quad 0 \qquad (1)$$

(1) 式において，もし時間 t_i から t_n の間で h-1 回の参照があれば t_i ，それ以外の場合は 0 である．

```
HLRU アルゴリズム
begin
  if (ファイル a を保持しているクライアントがアクセスを受信)
   if (ファイル a がキャッシュ内にある)
   then  /*history の更新*/
     for j:=n to 2 do
     begin
        hist (a, n)   = hist (a, n-1)
      end;
     hist (a, 1) =CurrentTime  /*先頭に現在の時間を代入*/
   /*現在と最近から n 番目の参照履歴の時間差を求める*/
  age[j]に CurrentTime-Hist (a, n) 0 を入れる
  qsort (age[j])    /*クイックソートで昇順にする*/
 if (キャッシュに空きが無い)   /*ファイルの置換*/
   age[j] = 0 の個数を数える
    if (age[j]の個数 > 設定値)
          LRU 置換
    else
      age[j]の最後尾のもの (最大のもの) と交換
 else
       空きに格納
end;
```

図 1 HLRU アルゴリズムの擬似プログラム

また，HLRU アルゴリズムにおけるキャッシュ内のファイ

ルを持つ参照履歴の更新は，最後尾から先頭に向かって繰り返しを行い，1 つずつ参照履歴をずらす．その後，空いた先頭に現在の参照時間を代入する．履歴関数で求めたキャッシュ内の各ファイルの最も新しい参照履歴から i 番目の参照履歴を t_i とする．現在の時間と t_i の差を age[j]に格納する．その後，age[j]でクイックソートを行い昇順にする．また，最も新しい参照履歴から i 番目の参照履歴がない場合は履歴関数で求めた 0 を age[j]に代入する．

キャッシュ内のファイルの置換は，age[j]=0 の個数を求め，これが設定値より大きければ LRU 置換を行う．そうでない場合は，age[j]が最大（最後尾）のファイルを置換する．これらの処理を擬似プログラムにしたものを図 1 に示す．

また本研究では，過去からの履歴 h の値を 2 とし，ファイル置換時の設定値を 7 とする．キャッシュ内の複製ファイルで行う置換の際に現在の時間と最近から h 番目に新しい参照履歴の時間の差をクイックソートで昇順にする．しかし，本研究の場合はキャッシュの容量は 10 ファイル分としているため，クイックソートを行わなくてもシミュレーション時の負荷はないものと考えられる．なので，クイックソートは行わないものとする．

3.2. LRU-SLFR アルゴリズム[4]

LRU-SLFR は LRU アルゴリズムと遅延時間のパラメータを組み合わせたものである．[4]で Roland と Abrams が，ほとんどのネットワーク接続が規則的な遅延パターンを持っているので遅延が良いパラメータになることを示した．遅延時間およびアクセス回数の情報を持った LRU-SLFR アルゴリズムは，基本的な LRU アルゴリズムを改善したアルゴリズムである．

LRU-SLFR の特徴である遅延概算アルゴリズムについて説明する．遅延概算アルゴリズムのプロセスは，TCP の RTT (ラウンド・トリップ・タイム) 概算アルゴリズムによって求められる．これを求める変数は以下に示す．また遅延時間は以下の (2) 式～ (4) 式で表される．

また，(2) 式～ (4) 式で用いられている変数は，$C(k)$：サーバ k への接続を開始するための遅延，$W(k)$：サーバ k への接続（バイト/秒）の帯域幅，S_C：ファイル用の接続設立の遅延，S_W：ファイル用の接続設立の帯域幅，A：変数で，参考文献[4]において 1/8 をセットする，i ：ファイル i を表す，$ser(i)$：ファイル i が存在するサーバを表示，si：置換に選ぶファイル i の大きさ，di：ファイル i の最小ダウンロード時間の評価である．

$$C(k) = (1-A) * C(K) + A * S_C \quad (2)$$

$$W(k) = (1-A) * W(K) + A * S_W \quad (3)$$

$$di = C(k) * ser(i) + si / \{W(k) + ser(i)\} \quad (4)$$

キャッシュ内の複製ファイルが持つ情報の更新は，キャッシュ内のファイルにアクセスがあった場合にそのファイルが持つアクセス回数の情報を 1 つ増やす．その後，そのファイルをキャッシュ内の先頭に移動する．このときに遅延時間は変更しない．

キャッシュ内の複製ファイルの置換は，まずキャッシュ内のファイルを 1 つのリストとして考える．そして，リストの最後尾のアクセス回数と同じものをリストの後ろから見ていき同じ回数でない場合まで繰り返す．このときアクセス回数が同じものを SCGW (Same Conditional Group Window) というグループとする．このグループ内で遅延時間が最小のファイルと置換する．アクセス回数が同じものがなければ，リストの最後尾のファイルと置換する．これらの処理を擬似プログラムとしてまとめたものを図 2 に示す．

LRU-SLFR アルゴリズム

```
begin
  /*カウントの更新*/
 if (リクエストファイルにアクセスがあった場合)
        SLFR_count[T][j]++  /*アクセス回数を+1 する*/
      if (SLFR[T][j]が先頭にない場合)
          SLFR[T][j]をキャッシュの先頭に移動させる
    /*ファイル置換*/
 if (ファイル要求をしたところのキャッシュに空きが無い)
      count = 0    /*カウントの初期化*/
      /*キャッシュの k-1 番目から先頭までファイル検索
        を行う*/
       for i : k-1 to 1
        /*キャッシュの最後尾 k 番目のファイルのアクセ
          ス回数と同じであるか*/
         if (SLFR_count[S][k] = = SLFR_count[S][i])
           count++
       /*最後尾のファイルのアクセス回数と同じものが
         なければ*/
         if (count=0)
           キャッシュの最後尾から k-count の間で遅延
           時間が最も小さいファイルを削除
       else
       最後尾のファイルを削除
       キャッシュの空きにリクエストファイルを格納
         遅延時間と SLFR_count[S][k]=1 を格納
end;
```

図 2 LRU-SLFR アルゴリズムの擬似プログラム

また，本研究では複製のファイルサイズはすべて同じにしている．さらに，従来のネットワークにおける帯域幅も同じにしている．そのため，上記の遅延時間を求めることができない．従ってピアに x,y 座標で位置情報を与えていることから，ファイルを要求したピアとファイルを供給するピアとの距離を遅延時間として考えることとする．

4. キャッシュ手法を用いたシミュレーション

本研究で行うシミュレーションの環境は，先行研究で用いられたキャッシュ置換アルゴリズムと本研究で適用するキャッシュ置換アルゴリズムの性能を比較するため，[2]，[3]と同じ環境にする．ただし，シミュレーション回数だけは 70,000 回とする．

非構造型トポロジーを用いたピュア型 P2P ネットワークにおいて，各ピアが自ピアや他のピアがオリジナルとして所持するファイルおよび複製にアクセスする環境を想定する．またシミュレーションで用いるピア数は，PLRG ネットワークトポロジーで構成した 10,254 ピアに x,y 座標を与え，その中からランダムに 1,000 ピア選択する．

表 1 シミュレーション環境[2][3]

パラメータ	設定値
ピア数	1,000 ピア
ピアのストレージ容量	10 ファイル
オリジナルファイルの種類	100 種類
初期配置ファイル数	100 ファイル
複製ファイル配置法	Owner Replication
ファイルリクエストレート設定	Zipf 分布
ネットワークトポロジー	Power-law Random Graph ネットワーク
ファイル検索手法	k-walker random walk
シミュレーション回数	70,000 回

5. キャッシュ置換アルゴリズムの性能比較

5.1. ファイルリクエストレートに対する複製数の比較

各アルゴリズムでシミュレーションを 70,000 回行った後にファイルリクエストレート別の複製ファイル数を求めるという作業を 10 回行い，10 回のデータの平均を求める．これを 21 回行い，1 回分は信頼区間を求めるために使い，残りの 20 回分のデータで信頼区間内に入っているかどうかの判断を行った．これらの 20 個のデータが信頼区間に入っているかどうかの例として図 3 を示す．これは FIFO の 20 個のデータとファイルリクエストレートが 0.02 の地点までの信頼区間を示したものである．FIFO のデータは全て×印で示し，信頼区間は信頼係数 95%で t 分布に従って求め，図 3 中に横棒で示している．

図 3 より，各ファイルリクエストレート別に信頼区間を取りデータの信頼性を図ったところ信頼性が低い区間が数箇所あり，そこでは 75%から 80%前後のデータしか信頼区間内に入らなかった．それ以外の区間では 95%前後のデータが入っていた．他の 4 つのアルゴリズムを比較してみても FIFO と同じような結果となった．しかし，数ヶ所だけ信頼区間内のデータが 20%～50%ぐらいのところも存在した．

この結果より本シミュレーションの結果は信頼出来るといえる．また，これらの信頼区間を求めるのに用いた 20 個のデータの平均をとり，グラフにしたものを図 4 に示す．これは各キャッシュ置換アルゴリズムにおけるファイルリクエストレート別の複製ファイル数の比較を行ったものである．

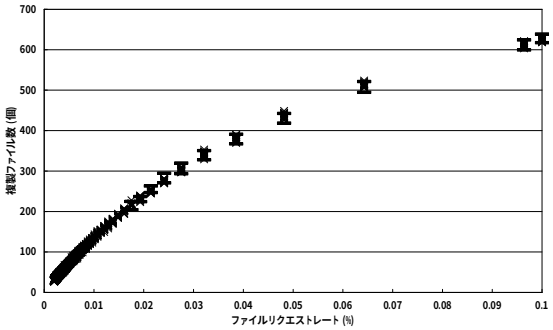

図 3 FIFO のシミュレーションデータと信頼区間

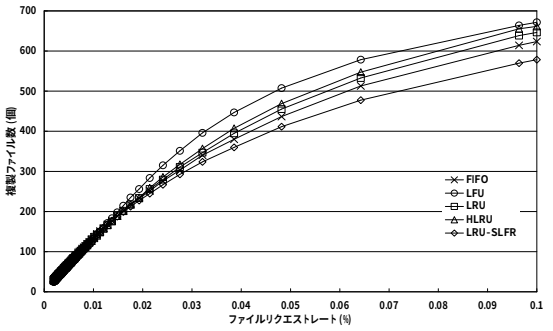

図 4 ファイルリクエストレートに対する複製ファイル数

図 4 に示されているように，先行研究[3]と同じく LFU が他のアルゴリズムと比較すると性能が良く，最適なアルゴリズムとなっていることが分かる．また，本研究で新たに適用した HLRU はファイルリクエストレート 0.1 のとき，LRU よりも 2.3%改善された．しかし，LFU よりかは 1.6%劣っている．さらに，LRU-SLFR においては FIFO よりも劣る性能であるということが分かる．

この結果より，LFU が LRU の派生系よりも良いアルゴリズムであるということがいえる．また，適用した HLRU は LRU よりも改善されたのに対し，LRU-SLFR は LRU の派生系ではあるが LRU よりも性能が劣っている．この理由としてキャッシュ内の複製ファイルの置換方法に関係があるといえる．

以上より，ファイルリクエストレートに対しての複製ファイル数では LFU が最適なアルゴリズムである．また，LRU-SLFR は性能的に適用することは困難である．さらに，キャッシュ数が多い環境では，HLRU は処理に時間がかかるため LRU を適用することも視野に入れておく必要がある．

5.2. Zipf 指数の変化に対する帯域幅の比較

先行研究に従って，各アルゴリズムにおける使用したネットワーク帯域幅に関して比較を行う．従来のネットワークにおける帯域幅とは異なるものである．

本シミュレーションでは定常状態にいたったシミュレーション回数 70,000 回目以降において, 使われたネットワーク帯域幅のデータを求める作業を 5 回行い, 5 回分のデータの平均を求めることにより, 使われたネットワーク帯域幅の数値を求めた. 図 5 は, シミュレーション回数 650,000 回目から 750,000 回目において 700,000 回目にファイルリクエストレートを求める際に用いるファイルリクエストレートの分散を決定する Zipf 指数を 1 から 0 に変化させたものである.

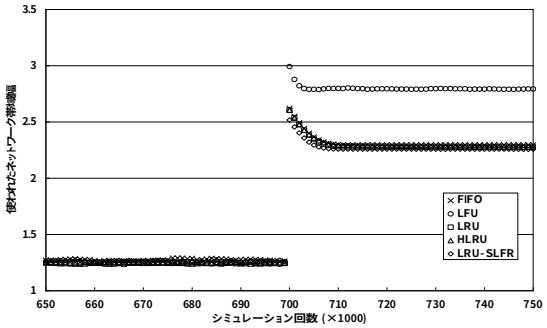

図 5 Zipf 指数 1 から 0 への変化時の帯域幅

図5より, Zipf指数を変化させる前のシミュレーション回数 650,000 回目から 700,000 回目の間では FIFO, LFU, LRU, HLRU, LRU-SLFR の各アルゴリズムは同じ状態であった. しかし, 700,000 回目に Zipf 指数 1 から 0 へ変化させたときの LFU の帯域幅は 2.9924 で, 先行研究[3]の示すように環境の変化に適応できていなかった. また, 最適値の帯域幅 2.9 に対して LRU の帯域幅は 2.6055 であった. 本研究で適用した HLRU は 2.6056 で, LRU-SLFR は 2.5182 であった.

さらに, 図 6 では Zipf 指数 1 から 0 へ変化させ Zipf 指数を保ち定常状態になったのちにシミュレーション回数 900,000 回目に再度 Zipf 指数を 0 から元の状態である 1 へ変化させたものである.

図 6 より, Zipf 指数を 0 から 1 へ変化させたときにも LFU の帯域幅は 1.2880 で, 環境の変化に適応できていないことが分かる. また, 最適値の帯域幅 2.0 に対して LRU の帯域幅は 1.8962 であった. 本研究で適用した HLRU は 1.8110 で, LRU-SLFR は 1.8838 であった. その後, シミュレーション回数が 910,000 回目からは 5 つのアルゴリズムすべての帯域幅が同じ値になった.

以上より, Zipf 指数の変化に対する使われたネットワーク帯域幅の比較では, 本研究で新たに適用した HLRU と LRU-SLFR は, 最適値と比較して先行研究[3]で最適なアルゴリズムとされた LRU とほぼ同じ性能であるといえる. さらに, HLRU と LRU-SLFR は環境の変化に適応できるアルゴリズムであることが分かる. その一方で, LFU は環境の変化についていけないアルゴリズムであるため, 適用することが困難であるということが分かる. よって, LRU の派生系である HLRU が Zipf 指数の変化に対する使われたネットワーク帯域幅の変化において最適なキャッシュ置換アルゴリズムであるといえる.

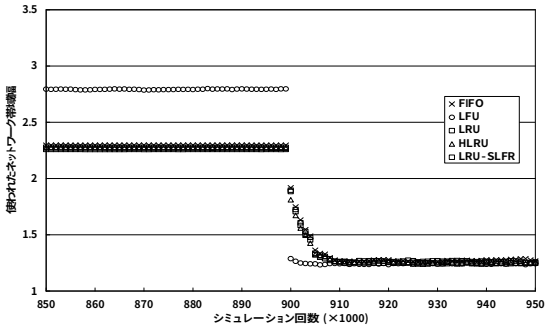

図 6 Zipf 指数 0 から 1 への変化時の帯域幅

この結果より, P2P ネットワークにおける最適なキャッシュ置換アルゴリズムは, HLRU である. しかし, HLRU はキャッシュ数が多い環境では, 処理に時間がかかるため LRU を適用することも視野に入れておく必要がある.

6. まとめ

我々は P2P ネットワークにおいて新たに HLRU と LRU-SLFR の二つのキャッシュ置換アルゴリズムを適用しシミュレーションを行い, 先行研究[3]のアルゴリズムと性能を比較した. シミュレーションをした結果, ファイルリクエストレートに対する複製ファイル数の比較では, 性能が良い順に LFU, HLRU, LRU, FIFO, LRU-SLFR という結果になった. また, Zipf 指数の変化に対する使われたネットワーク帯域幅の比較では, HLRU と LRU-SLFR は LRU とほぼ同じ性能であるといえた. 以上より HLRU が最も良いキャッシュ置換アルゴリズムである.

参考文献

[1] A. I. Vakali, "LRU-based algorithms for Web Cache Replacement," Lecture Notes In ComputerScience, Vol. 1875, pp. 409-418, 2000.

[2] S. Tewari, L. Kleinrock, "Analysis of Search and Replication in Unstructured Peer-to-Peer Networks," UCLA Computer Science Dept Technical Report UCLA-CSD-TR 050-006, 2005.

[3] S. Tewari, L. Kleinrock, "Proportional Replication in Peer-to-Peer Networks," Proc.IEEE INFOCOM 2006, pp. 1-12, 2006.

[4] S. W. Shin, K. Y. Kim, J. S. Jang, "LRU based small latency first replacement (SLFR) algorithm for the proxy cache," Proc. IEEE/WIC, pp. 497-502, 2003.