周辺情報検索におけるプロキシシステムのためのキャッシュ置換アルゴリズム

田頭茂明[†] 吉岡浩路[††] 藤田聡[†]

本論文では，クライアント・サーバモデルを用いた周辺情報検索システムにおいて，プロキシ型のキャッシュシステムを導入することを考え，そのプロキシキャッシュ上での効果的なキャッシュ置換アルゴリズム（LFU-FF 法）を提案する．LFU-FF 法は，参照の局所性に加えて周辺情報のデータ構造を考慮して，周辺情報に適したキャッシュデータの置き換えを実現する．具体的には，本研究で対象とするシステムでは，周辺情報のデータ構造において地理的な隣接関係を記憶領域上での連続関係として保持することにより，検索エリアに該当する周辺情報への高速な連続アクセスを実現している．LFU-FF 法では，この構造を利用してキャッシュ内での周辺情報の連続性と参照回数を評価値として用い，キャッシュデータの追加または削除を行う．LFU-FF 法を採用したプロキシシステムのプロトタイプシステムを構築し評価を行った．結果として，LFU 法と比べて，キャッシュサイズが全体の 8%のときに，応答時間を約 21%向上することができた．

A Replacement Algorithm for a Proxy Cache in Location-dependent Information Retrieval System

Shigeaki Tagashira,[†] Kouji Yoshioka[††] and Satoshi Fujita[†]

In this paper, we propose a proxy caching algorithm, called LFU-FF, for location-dependent information retrieval system based on a client-server model. The main objective of LFU-FF is to improve replacement performance in the proxy cache by making use of the structure information of location-dependent data, other than only the locality information used in the well-known LFU algorithm. More concretely, the data structure used for our target system is designed to allow continuous access to location-dependent data in a specified area in order to provide fast retrieval. LFU-FF incorporates the continuity between location-dependent data in the cache into the LFU algorithm for high replacement performance. The result of experiments conducted on a prototype system implies that LFU-FF can improve the response time by 21% compared with the LFU algorithm.

1. はじめに

現在，PDA やノート型 PC などの移動型計算機を利用したモバイルインターネットが著しく普及し，いつでもどこでも様々な情報を得られるようになってきている．また，GPS 機器を容易に利用できるようになり，移動型計算機の現在の**位置情報**（緯度，経度）を取得できる環境が整備されてきている．このため位置情報を利用したサービス（位置情報サービス）の需要が増加しており，特に現在地周辺の地図情報，買い物や食事のための店舗情報や建物情報，イベント情報などに代表されるユーザの周辺の情報を移動先で獲得できる**周辺情報検索サービス**が注目を集めている．周辺情報検索を実現する代表的なシステムとしてはカーナビゲーションシステムがあげられる．カーナビゲーションシステムでは，移動型計算機（クライアント）上にある DVD や HDD といった大容量の記憶装置に周辺情報を蓄積しておき，ユーザが要求する周辺エリアの情報を提供する．一方で，モバイルインターネットを用いて周辺情報検索サービスを実現するシステムが近年注目を集めている[1),2)]．これらのシステムでは，サーバが周辺情報を保持し，ユーザの周辺エリアの情報をネットワークを通じてクライアントへ提供する．このようなシステムは，大容量の記憶装置を持たない小型のクライアントでも利用できる汎用性と，サーバの情報をつねに最新に維持することで鮮度の高い周辺情報を提供できる即応性といった利点を持つために，様々な分野での利用が期待されている．

† 広島大学大学院工学研究科
Graduate School of Engineering, Hiroshima University
†† 広島大学工学部第二類
Faculty of Engineering, Hiroshima University

本研究では，このクライアント・サーバモデルで実現される周辺情報検索システムにおいて，プロキシ型のキャッシュシステムを導入し，そのプロキシ上での効果的なキャッシュ置換アルゴリズム（LFU-FF 法：Least Frequently Used-Fragment Factor 法）を提案する．LFU-FF 法では，取り扱う周辺情報の構造として，Kiwi-W コンソーシアムで標準化がすすめられているデータ構造[3] に特化しており，参照の局所性に加えて，このデータ構造の性質を考慮してキャッシュデータの置換を行う．具体的には，本研究で対象とするシステムでは，周辺情報のデータ構造において地理的な隣接関係を記憶領域上での連続関係として保持することにより，検索エリアに該当する周辺情報への高速な連続アクセスを実現している．この構造を考慮して，LFU-FF 法ではキャッシュ内での周辺情報の隣接性と参照回数を評価値として用いて，キャッシュデータの追加と削除を行う．LFU-FF 法を実装したプロキシシステムのプロトタイプシステムを構築し評価を行った．結果として，LFU 法や LRU 法と比べて，キャッシュサイズが全体の 8%のときに，応答時間を約 21%向上することができた．

本論文の構成は以下のとおりである．まず最初に，2 章で本研究で対象とする周辺情報のデータ構造について述べ，その構造上でのプロキシキャッシュシステムについて議論する．3 章では，プロキシシステムにおけるキャッシュ置換アルゴリズムの提案を行う．4 章では，プロキシキャッシュシステムの試作を行い，提案した LFU-FF 法の性能評価を行う．5 章で関連研究を述べ，最後に 6 で，本システムについてのまとめと今後の課題について述べる．

2. 周辺情報のプロキシキャッシュ

2.1 周辺情報のデータ構造

クライアント・サーバ型の周辺情報検索システムでは，クライアントが欲しいエリア（検索エリア）の要求をサーバに送信し，サーバは該当する周辺情報をクライアントへ提供する．本研究が対象とするシステムでは，周辺情報を緯度・経度の 2 次元の空間上にマッピングし，緯経度方向により正規化されたメッシュ（**基準エリア**）単位で処理する．ただし，$2^m \times 2^m$ 個（m は正の整数）の基準エリアで，すべての周辺情報を構成できるように正規化を行う．本論文では，経度方向に x $(x = 1, 2, \ldots, 2^m)$ 番目，緯度方向に y $(y = 1, 2, \ldots, 2^m)$ 番目の基準エリアの情報を $i(x, y)$ で表記する．また，複数の基準エリアからなる正方エリアを，その正方エリアの対角線の始点に対応する基準エリア $i(x_1, y_1)$ と，サイズ s を用いて，$I_s(x_1, y_1)$ で表すことにする．ここで，サイズ s とは正方エリアの 1 辺の基準エリアの個数を表す．

サーバは，$2^m \times 2^m$ 個の基準エリアを効果的に記憶領域上に配置することで，検索エリアに該当する基準エリアへの高速なアクセスを実現している．具体的には，基準エリアの地理的な隣接関係を，記憶領域上での連続関係として保持して，基準エリアを記憶領域上に配置する．基準エリアの配置順序を決定する具体的な手順について以下に示す．

(1) $2^n \times 2^n$ の入力エリアを $2^{n-1} \times 2^{n-1}$ のエリアへ 4 等分し，分割したエリア（分割エリア）を左下，左上，右下，右上のエリアの順（N 字順）に配置する．
(2) 分割エリア内でも配置順序を決定するために，4 つの分割エリアを入力エリアとして，さらに (1) の操作を繰り返す．
(3) 分割エリアのサイズが 1 になるまで（すなわち基準エリアになるまで），(1) と (2) の操作を繰り返すことで，すべての基準エリアの配置順序を決定することになる．

これを式で表すと，エリア $I_{2^m}(x_1, y_1)$ の順序集合を $N_{2^m}(x_1, y_1)$ で表すとき，$N_{2^m}(x_1, y_1)$ は，以下の式を用いて求めることができる．

$n = 0$ のとき

$$N_{2^n}(x_1, y_1) = i(x_1, y_1)$$

$n > 0$ のとき

$$\begin{aligned} N_{2^n}(x_1, y_1) = [&N_{2^{n-1}}(x_1, y_1), \\ &N_{2^{n-1}}(x_1, y_1 + 2^{n-1}), \\ &N_{2^{n-1}}(x_1 + 2^{n-1}, y1), \\ &N_{2^{n-1}}(x_1 + 2^{n-1}, y_1 + 2^{n-1})] \end{aligned}$$

周辺情報の具体的な配置例として，$N_4(1, 1)$ の周辺情報について考えると，**図 1** に示すように，$N_4(1,1) = [N_2(1,1),\ N_2(1,3),\ N_2(3,1),\ N_2(3,3)] = [\ [\ i(1,1),\ i(1,2),\ i(2,1),\ i(2,2)\ ],\ [\ i(1,3),\ i(1,4),\ i(2,3),\ i(2,4)\ ],\ [\ i(3,1),\ i(3,2),\ i(4,1),\ i(4,2)\ ],\ [\ i(3,3),\ i(3,4),\ i(4,3),\ i(4,4)\ ]\ ]$ の順に基準エリアを格納することになる．ここで，上記の操作でエリアを再帰的に基準エリアまで分割していく過程で，入力エリアとなるエリアの集合を **N 字エリア集合**と呼ぶことにする．すなわち，エリア $I_{2^m}(x_1, y_1)$ の N 字エリア集合は，$\bigcup_{k=0}^{m} \bigcup_{l_x=1}^{2^{m-k}} \bigcup_{l_y=1}^{2^{m-k}} I_{2^k}(x_1 + (l_x - 1)2^k, y_1 + (l_y - 1)2^k)$ と表すことができる．この配置方式を用いることで，N 字エリア集合に含まれるエリアの情報は，記憶領域上の連続領域に格納されることに注意されたい．

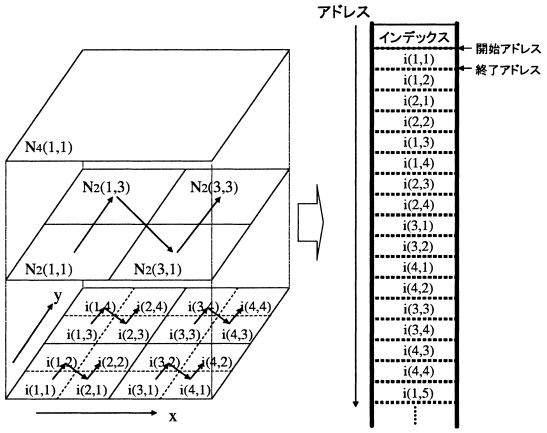


図 1 周辺情報のデータ構造
Fig. 1 Structure of location-dependent data.

次に，サーバの周辺情報へのアクセスについて説明する．本システムでは，クライアントが要求する検索エリアは N 字エリア集合に含まれるエリアに限定している．ユーザが N 字エリア集合に含まれない検索エリア（N 字エリア集合の境界をまたぐエリア）を要求することも想定できるが，そのような場合は本研究では，クライアントアプリケーションが，N 字エリア集合の同位または下位のエリアを組み合わせて検索エリアを自動的に構成することを考えている．

サーバは，周辺情報へのアクセスのために，記憶領域上にインデックス領域を確保しており，この領域には周辺情報を構成する各基準エリアの記憶領域上での開始アドレスと終了アドレスを格納する．検索エリア $I_s(x_1, y_1)$ へのアクセスは，インデックスから獲得した $i(x_1, y_1)$ の開始アドレスから，$i(x_1+s-1, y_1+s-1)$ の終了アドレスまでを連続してアクセスするだけでよい．このように周辺情報の連続構造を利用することで，データベースで用いられているような多くの緯度・経度の検索処理（照合処理）を必要とする方式と比較して，高速に（連続的に）周辺情報にアクセスできる．

たとえば，図 1 において，$I_2(1,1)$ が検索エリアであれば，$i(1,1)$ の開始アドレスから $i(2,2)$ の終了アドレスまでのデータを獲得すればよく，また，$I_4(1,1)$ が検索エリアであれば，$i(1,1)$ の開始アドレスから $i(4,4)$ の終了アドレスまでのデータを獲得すだけでよい．この配置方式は，Kiwi-W コンソーシアムで標準化がすすめられており[3]，カーナビゲーションシステムなどで周辺情報の配置方式として幅広く採用されている．本論文ではこの配置方式のことを **N 字アドレッシング** と呼ぶことにする．

N 字アドレッシングの効果を確かめるために，周辺情報にアクセスするベンチマークプログラムをサーバ上で実行し，N 字アドレッシングを用いる場合と N 字アドレッシングを用いない場合との周辺情報へのアクセス時間を比較した．N 字アドレッシングを用いない場合では，基準エリアを除く N 字エリア集合のエリアへのアクセスに関しては，ランダムアクセスになることに注意されたい．サーバは周辺情報とインデックスをすべてメモリ上に配置する．ベンチマークプログラムは，N 字エリア集合の中からランダムに決定したエリアへのアクセスを 5,000 回繰り返す．ベンチマークプログラムを複数のプロセスとして同時に動かし，各プロセスの平均終了時間を計測した．結果を**表 1** に示す．

表 1 周辺情報への平均アクセス時間
Table 1 Average access time to location dependent data.

プロセス数	1	2	4	8	16
N 字アドレッシングを用いた場合 (s)	3.8	5.9	11.8	23.5	46.4
N 字アドレッシングを用いない場合 (s)	8.9	10.7	21.4	42.7	81.7

結果から，N 字アドレッシングを用いることにより，用いない場合と比べて約 2 倍の性能の向上が確認できる．これは，インデックスの走査の回数の削減（すなわち，開始アドレスと終了アドレスを走査するだけでよい）と，メモリへのバーストアクセスによるものと考えられる．このように，N 字アドレッシングは，本来の目的である DVD などのシーク時間が遅いメディアだけでなく，ランダムアクセス可能なメモリなどのメディアにおいても有効であることが確認できる．

2.2 周辺情報のキャッシング

クライアントとサーバとの間にプロキシキャッシュシステムを導入し，キャッシュの効果によりシステムの性能の改善を試みる．本研究で想定するプロキシ環境としては，周辺情報を提供するサーバと，そのフロントエンドとして一定のエリアごと（たとえば都道府県ごと）にプロキシを設置することを考えている．クライアントは，あらかじめ準備されているプロキシのデータベースを用いて，現在の位置情報から利用するプロキシを決定し，そのプロキシを用いて周辺情報を検索する．クライアントがプロキシを介して検索することで，他のクライアントがすでに検索した周辺情報のキャッシュを利用できる．すなわち，多くのクライアントにより検索させる（人気がある）周辺情報をキャッシュを用いて効果的にクライアントへ提供できる．この結果，クライアントの応答時間，クライアント・サーバ間の通信量，サーバへの負荷（要求数）の

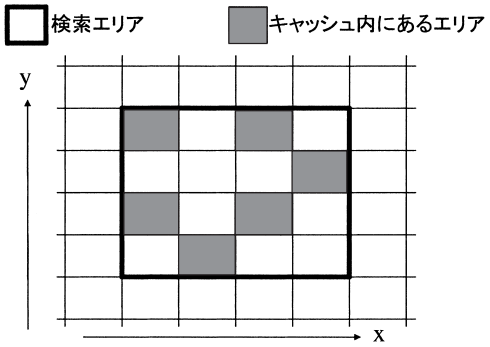


図 2 キャッシュ内の断片化の例
Fig. 2 Example of cache fragmentation.

削減が期待できる．

プロキシは基準エリア単位で周辺情報をキャッシュへ格納する．クライアントからの要求がキャッシュにヒットした場合はキャッシュデータをクライアントへ提供する．キャッシュにミスヒットした場合には，サーバからキャッシュにない部分を取得しクライアントへ提供することから，ヒット時に比べて余分なコストが生じることになる．このために，キャッシュ置換アルゴリズムを用いて，できるだけヒットするデータを有限なキャッシュへ格納するようにキャッシュを管理する必要がある．

キャッシュ置換アルゴリズムでよく利用される LFU 法や LRU 法では，キャッシュデータの参照回数または参照時間を用いて，キャッシュのヒット率を最大にするようにキャッシュを置き換える．しかし，N 字アドレッシングを採用するシステムにおいては，検索エリアの可変サイズや参照エリアの偏りにより，これらのアルゴリズムを用いると，**図 2** に示すようにキャッシュ内で連続構造を維持できなくなる（断片化する）可能性がある．断片化の発生により，プロキシは検索エリアと比べて下位の（小さな）エリアをサーバから断片的に取得することになる．キャッシュを用いることはシステムの性能の改善を意図していたが，断片化により N 字アドレッシングによる連続アクセスの効果がなくなり，キャッシュを用いることが逆にサーバの負荷と応答時間を増加させる原因になる可能性がある．

たとえば，キャッシュが図 2 のような場合は，プロキシは 4×4 の検索エリアをより細かくした 10 個の 1×1 の基準エリアをサーバへ要求する必要がある．ここで，本論文ではこの断片化の尺度として，**断片化数**を定義する．断片化数とは，プロキシが検索エリアを構成するために必要なサーバへの最小の要求数を指す．すなわち，検索エリア内でキャッシュに存在しないエリアを，N 字エリア集合内のできるだけ上位のエリアで構成するのに必要なエリア数のことである．検索エリアに必要な基準エリアが，キャッシュにまったく存在しない場合では断片化数は 1 であり（なぜなら，検索エリア全体を 1 度要求するだけでよいからである），また逆にすべてキャッシュされている場合では 0 である（要求する必要がない）．また，図 2 の場合では断片化数は 10 となる．

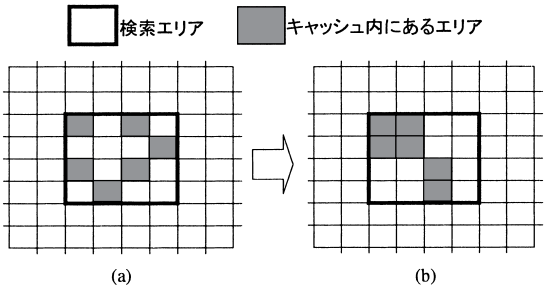


図 3 提案手法のキャッシュ置換の概要
Fig. 3 Overview of cache replacement in LFU-FF.

3. 提案キャッシュ置換アルゴリズム

我々は，参照の局所性に加えて N 字アドレッシングを考慮するキャッシュ置換アルゴリズム（LFU-FF 法）を提案する．LFU-FF 法は，キャッシュ内で最もアクセス頻度（スコア）の低いキャッシュデータをキャッシュから削除する LFU 法を拡張した手法である．提案手法の基本的なアプローチは，アクセス頻度情報だけでキャッシュ置換を行えば，**図 3** (a) に示すような断片化が発生するが，アクセス頻度が同程度ならば連続性を重視して図 3 (b) のようにキャッシュを置き換えて，連続構造を維持することである．これを実現するために LFU-FF 法では，キャッシュデータのスコアの算出時に，アクセス頻度情報に加えて Fragment Factor を考慮することで，キャッシュデータの断片化を抑えるようにキャッシュを置換する．

3.1 統計情報の管理

プロキシは，クライアントからの検索エリアに関して統計情報を管理する．この管理構造としては，**図 4** に示すような 4 分木の階層構造を用いる．節点に検索エリアを対応させ，そのエリアにアクセスした頻度情報を格納する．葉節点は，1×1 の基準エリアの要求に対応し，その親節点は 2×2 のエリアの要求に対応している．すなわち，N 字アドレッシングでは上位のエリアは 4 つの下位のエリアから構成されており，節点の親子関係でその構造を表現している．クライアントから要求があった場合は，検索エリアに対応する節点の値を 1 増加させる．

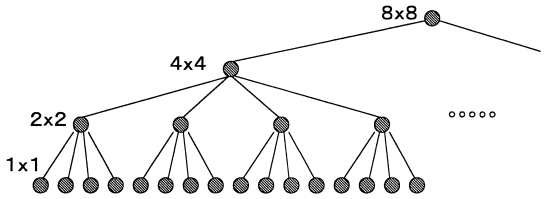


図 4 統計情報の管理構造
Fig. 4 Tree structure for managing statistical information in LFU-FF.

3.2 スコア値によるキャッシュ置換

キャッシュ置換は，プロキシが新たにサーバから基準エリアを取得したときに行われる．キャッシュ容量に十分な空き領域がある場合には，新しい基準エリアをキャッシュに追加する．キャッシュ容量に空きがない場合は，すでにキャッシュに存在する基準エリアと，新しく取得した基準エリアに対してスコア値を計算し，キャッシュ容量内に収まるまでスコア値の小さな基準エリアから順にキャッシュから削除する．

葉節点 v $(v = 1, 2, \ldots, 2^{2m})$ のスコア値は以下の式により算出する．

$$Score(v) = AF(v) + \sum_{q \in Pred(v)} \frac{AF(q)}{\alpha \times FM(q) + 1}$$

ただし，$Pred(v)$ は葉節点 v のすべての上位エリアに対応する節点の集合を表す（すなわち，図 4 では，根節点と，根節点から葉節点 v までのパス上の節点の集合である）．$AF(v)$ は，3.1 節で説明した構造で管理される節点 v のアクセス頻度，α は重みを，$FM(q)$ は，節点 q に対応するエリアの断片化数を表す．この式において，$\alpha = 0.0$ の場合では LFU 法のスコア算出式と等価になる．また，この式の右辺第 2 項の分母を Fragment Factor と呼ぶことにする．このスコアの算出法では，葉節点のスコア（すなわち基準エリアのスコア）は，その葉節点のアクセス頻度に加えて，その上位エリアのアクセス頻度を上位エリアの断片化に応じて累積するようにしている．Fragment Factor は，上位エリアからの累積率を断片化に応じて変化させるために導入している．すなわち，上位エリアがキャッシュ内で断片化しているならば，その累積率を低くすることで，葉節点でのスコアを下げるように働き，逆に，上位エリアがすべてキャッシュ内に存在しているならば（断片化していないならば），累積率を高くし，そのエリアに関する葉節点でのスコアを上げるように働く．

LFU-FF の実施例について説明する．**図 5** では例として 4×4 のエリアに関する LFU-FF の構造が示されている．この例では，節点 $v1$，$v2$，$v3$，$v4$，$v6$ がキャッシュ内に存在しており，これらの節点すべてをキャッシュ内に確保できないために，各節点のスコアを計算しキャッシュから削除する節点を決定している．$\alpha = 1.0$ の場合の $v1$ におけるスコアは，上述した計算法から $Score(v1) = 20 + \{\frac{10}{1.0 \times 0 + 1} + 0\} = 30$ となる．同様に，$Score(v2) = 15$，$Score(v3) = 50$，$Score(v4) = 40$ と計算できる．節点 $v6$ に関しては，$Score(v6) = 12 + \{\frac{4}{1.0 \times 3 + 1} + 0\} = 13$ となる．これから，$v6$ が最小スコアとなり，$v6$ がキャッシュから削除される．LFU では，単に上位エリアのアクセス頻度を累積するだけなので，$v6$ が削除されるのではなく，$v2$ が削除されることになる．一方で，LFU-FF では，$v2$ が削除されると 2×2 のエリアで断片化が発生するために，$v6$ が削除されることになる．

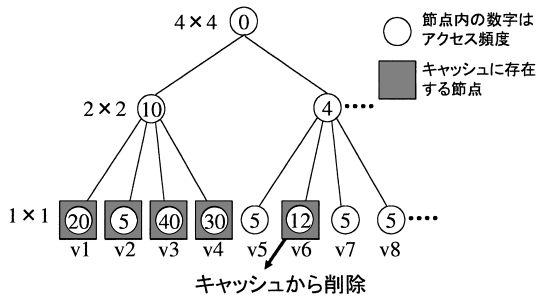


図 5 LFU-FF の実施例
Fig. 5 Example of cache replacement using LFU-FF.

4. 提案システムの評価

4.1 実験環境

3 章で提案した LFU-FF 法を実装したプロキシシステムを試作し，実験的に提案手法の性能評価を行う．LRU，LFU，LFU-FF の各手法についてクライアントにおける平均応答時間，プロキシ・サーバ間のリクエスト回数，プロキシ・サーバ間の通信量を計測し，これらを比較して性能の評価を行う．

本実験で用いる周辺情報として，**図 6** に示すような施設情報を用いた．施設情報のデータサイズの分布は，都市人口の分布や Web サイトの人気分布などの社会現象の分布として用いられる Power Law（ベキ法則）に従うことが一般に知られている．すなわち，少数のエリアに大半の情報が集まり，大多数のエリアには少数の情報しかないことを示している．これを実際に確かめるために，日本の 5 つの都市の施設情報を用いてエリアあたりのデータサイズの分布を調べてみた．**図 7** にその結果を示す．この調査では，施設情報を緯度・経度の 2 次元の空間上にマッピングした

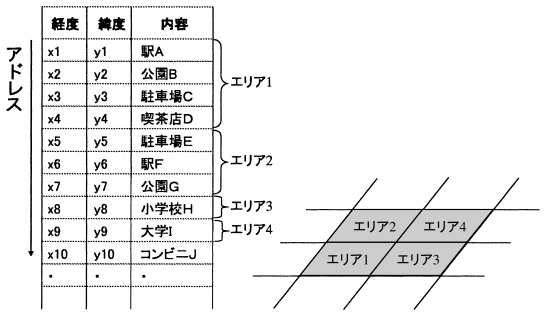


図 6 実験で用いた周辺情報の例
Fig. 6 Example of location dependent data using experiments.

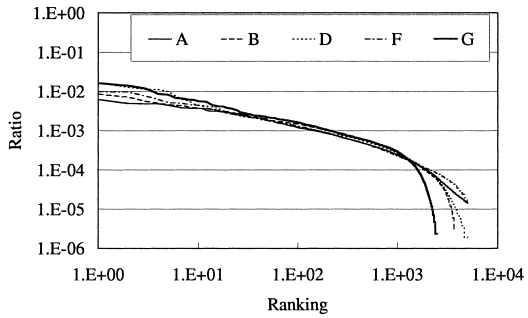


図 7 エリアあたりのサイズ分布
Fig. 7 Rank ordering of areas by data size.

後に，緯度・経度の情報から 32 秒単位（約 1 km）でメッシュに分割し，各エリアのデータサイズ（すなわち，施設数）を調べた．32 秒単位で分割するメッシュは，文献 3) で述べられている検索メッシュの定義であり，4 バイト表現で全世界をカバーすることができる．図 7 では，横軸にはサイズ順にエリアを並べ，縦軸には全体のサイズに対する比率を示している．ここで図中の軸は対数軸であることに注意されたい．

結果から，5 つの都市が対数軸において右肩下がりの分布を示しており，およそベキ法則に従っていることが分かる．このことから本研究では，ベキ法則に従った分布を持つ施設情報を用いて実験を行った．具体的には，図 7 の都市 A の情報を用いている．都市 A の情報は，128×128 のエリア数で全データサイズは約 15 MB である．サーバは，施設情報のデータ本体を N 字アドレッシングに従い記憶領域に格納し，加えて 2.1 節で述べたインデックスも同時に格納している．この分布では，16,384 エリア中の上位 5 位のエリアで全体のデータサイズの約 2.5%を占めており，上位 20 位のエリアでは約 8%，上位 100 位では約 22%を占めている．

クライアントからの検索要求は，最初に検索するエリアを決定し，次に検索エリアのサイズを決定する．ただし，クライアントが検索可能なエリアは，N 字エリア集合内のエリアのみとしている．クライアントが検索する総回数は，5,000 回とした．検索エリアのサイズに関しては，1×1，2×2，4×4，8×8 の 4 通りを設定し，その比率を変更して実験する．本実験では，検索するエリアに局所性を持たせている．文献 4) では，検索の局所性として繁華街などの情報が密集しているエリア（データサイズが大きいエリア）にアクセスが集中することが実験的に示されている．このことから，周辺情報のエリアあたりのデータサイズの分布がベキ法則に従うために，クライアントのアクセス分布についてもベキ法則に従うものとした．具体的には，エリアのデータサイズに応じて**図 8** のように，各エリアの検索確率を設定した．この図では，横軸は検索確率が高い順にエリアを並べたものであり，縦軸はそのエリアが検索される確率を示している．この分布では，上位 5 位のエリアが全体の約 20%検索されることになり，上位 20 位のエリアでは約 40%が，上位 100 位のエリアでは約 70%が検索されることになる．

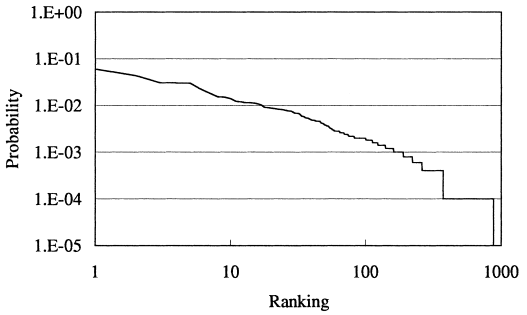


図 8 エリアあたりのアクセス分布
Fig. 8 Rank ordering of areas by access probability.

サーバプログラムは，Perl で書かれた CGI を用いて，周辺情報をクライアントへ提供する．プロキシシステムは，キャッシュ置換アルゴリズムを実装し，クライアントからの要求を受け付ける．要求を受付後，プロキシは以下のように動作する．

(1) クライアントからの検索エリア内で，キャッシュされていないエリアを特定する．
(2) キャッシュされていないエリアにおいて，N 字エリア集合内で最大のエリアをサーバにリクエストする．
(3) 要求したエリアをサーバから受け取り，検索エリアがすべて揃ったなら，クライアントへ情報を提供する．揃っていない場合は，(1) の操作に戻る．
(4) キャッシュの置き換えを行う．

このプロキシシステムでは，サーバへエリアをリクエストするごとに，TCP コネクションを 1 つ確立するように実装している．

実際に用いた計算機環境としては，サーバ：Xeon 1.7 GHz×2 1024 MB，プロキシ：Pentium4 2.53 GHz 1024 MB，クライアント：PentiumII 300 MHz 200 MB である．

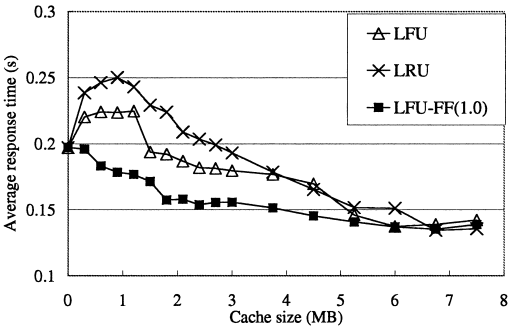
4.2 実験結果

4.2.1 基本評価

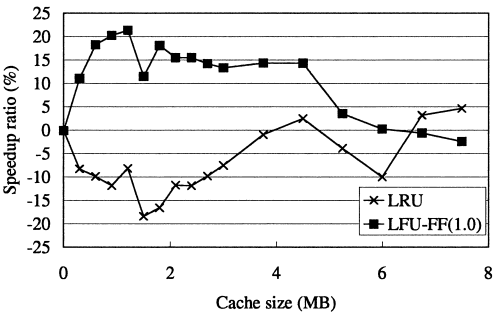
LRU，LFU，LFU-FF のクライアント平均応答時間，プロキシ・サーバ間のリクエスト回数，プロキシ・サーバ間の通信量の基本的な性能評価を行う．LFU-FF の重み α の値は 1.0 とし，クライアントの要求サイズの比率は $(1\times1):(2\times2):(4\times4):(8\times8)=1:1:1:1$ としている．なお，実験結果中で LFU-FF の括弧内の数字は，3.2 節で述べた重み α の値を示す．

まず，クライアントの周辺情報を獲得するまでの平均応答時間の結果を**図 9** (a) に示す．横軸はキャッシュサイズを表し，縦軸は平均応答時間を表す．また，LFU と比較した場合の LRU と LFU-FF(1.0) の改善率を図 9 (b) に示す．図 9 (a) から，キャッシュサイズが小さいときに LFU-FF が効果があり，大きくなると各手法の差がなくなることが分かる．LFU の結果では，キャッシュサイズが 1.5 MB 以上になるまではキャッシュサイズ 0 MB のときと比べて性能が悪化していることが分かる（キャッシュサイズ 0 MB は，プロキシキャッシュを利用しないときを示している）．この理由は，キャッシュを利用することにより，断片化が発生し逆に性能が悪化してしまったためと考えられる．

一方で，LFU-FF ではすべてのキャッシュサイズにおいて，キャッシュサイズが 0 MB のときと比べて性能を改善できている．具体的には，図 9 (b) からキャッシュサイズが全体の 8%（1.2 MB）のときでは，LFU と比較すると，LFU-FF では約 21%の向上が見られた．図 7 と図 8 からキャッシュサイズが全体の 8%のときでは，上位 20 位のエリアしかキャッシュに入らないことになるが，LFU-FF では，このキャッシュサイズでもキャッシュを効果的に利用できている．このことからアクセス頻度だけでなく断片化を抑えることで，小さなキャッシュサイズでもキャッシュを効果的に利用できることが確認できる．逆にキャッシュサイズが大きい場合は，よく検索される情報に関してキャッシュできる十分な容量があるために，各手法の差がなくなったと考えられる（1 度しか検索されないエリアまで含めると，図 7 と図 8 から周辺情報の全体の約 60%のキャッシュサイズで，検索されるすべての情報をキャッシュすることができる）．



(a) LRU，LFU，LFU-FF(1.0) の応答時間

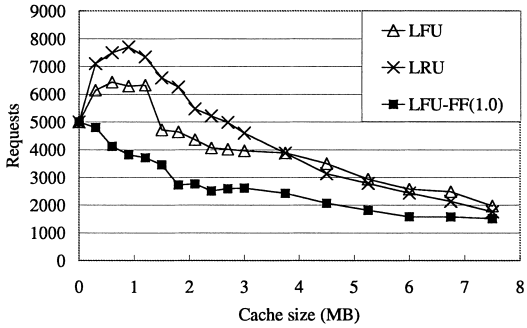


(b) LFU と比較した場合の LRU と LFU-FF(1.0) の応答時間の改善率

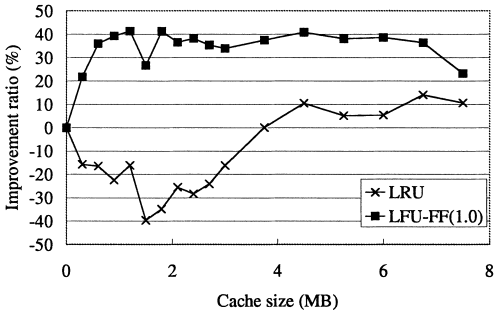
図 9 平均応答時間の結果
Fig. 9 Average response time.

次に，各手法におけるプロキシとサーバとの間のリクエスト回数を計測した．結果を**図 10** (a) に示す．横軸はキャッシュサイズ，縦軸はリクエスト回数を表す．また，LFU と比較した場合の LRU と LFU-FF(1.0) のリクエスト回数の改善率を図 10 (b) に示す．図 10 (a) から，LFU-FF は全体的にリクエストの数が少ないことが分かる．LFU-FF では，キャッシュサイズが 1.2 MB のとき最も効果があり，LFU と比較して約 40%の向上が見られた．LFU では，キャッシュサイズが 1.5 MB 以下のときリクエスト回数が 5,000 回以上になっている．これは，1 つのクライアントの要求に対して，複数のリクエストを送信していることが示されており，断片化の影響が出ていることが分かる．

次に，各手法におけるプロキシ・サーバ間の総通信量を計測した．結果を**図 11** に示す．横軸はキャッシュサイズ，縦軸は通信量を表す．図 11 から LFU-FF は，LFU と同程度の通信量で実現できていることが分かる．LFU-FF は断片化を抑えるために，局所性をある程度犠牲にしていることから，通信量が増加することが考えられる．しかし，連続構造を確保しつつアクセス頻度が同程度な基準エリアを選択するために，その増加は全体の通信量と比べて無視できるものとなって



(a) LRU，LFU，LFU-FF(1.0) のリクエスト回数



(b) LFU と比較した場合の LRU と LFU-FF(1.0) のリクエスト回数の改善率

図 10 プロキシ・サーバ間のリクエスト回数の結果
Fig. 10 The number of requests between the proxy and the server.

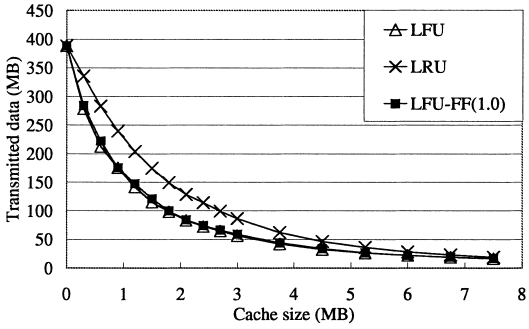


図 11 プロキシ・サーバ間の通信量
Fig. 11 The total size of the transmitted data between the proxy and the server.

いる．

次に，キャッシュ内における周辺情報の断片化の度合いについて計測を行った．クライアントの全要求に対するキャッシュのヒット率，フラグメント率，ミスヒット率の割合を計測した．この実験ではキャッシュサイズを 1.2 MB とした．ここで，ヒット率とは検索エリアがすべてキャッシュ内にあった割合，フラグメント率とは検索エリアの内の一部がキャッシュ内にあった割合，ミスヒット率とは検索エリアがすべてキャッシュになかった割合を示している．結果を**図 12** に示す．図 12 から，LFU-FF は，LFU や LRU と比べて，キャッシュ内で断片化を起こしている割合が減り，ヒット率とミスヒット率が上がっていることが分かる．

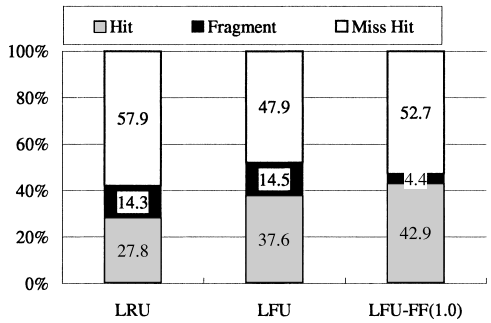


図 12 全要求に対するヒット率，フラグメント率，ミスヒット率の割合
Fig. 12 The percentage of hit ratio, fragment ratio, and misshit ratio to the total inquiries.

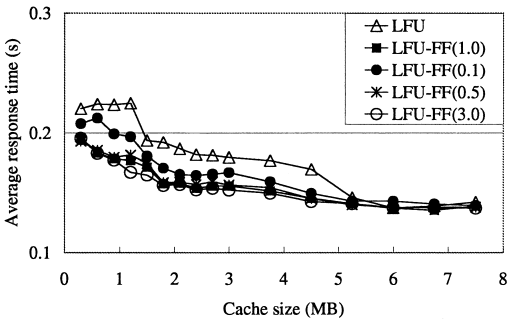


図 13 α の平均応答時間への影響
Fig. 13 The impact of parameter α to average response time.

以上の結果から LFU-FF の本来の目的であるフラグメント率の減少を確認でき，その効果として応答時間とリクエスト数を削減できたことを実験により確かめることができた．

4.2.2 α の影響

LFU-FF の重み α の影響について調べるために，α を変更し，応答時間と，キャッシュのヒット率，ミスヒット率，フラグメント率を計測した．応答時間の実験結果を**図 13** に示す．横軸はキャッシュサイズ，縦軸は平均応答時間を示す．$\alpha = 0.1$ の場合では，Fragment Factor の効果が少ないために，LFU と比べて LFU-FF は若干の性能の改善にとどまっている．$\alpha \geq 0.5$ の場合には，$\alpha = 0.1$ と比べてその改善率が向上していることが分かる．しかし，さらに α の値を増加させても（たとえば，1.0 と 3.0），その性能にほとんど変化がないことが分かる．このことをキャッシュのヒット率，ミスヒット率，フラグメント率の側面から確認

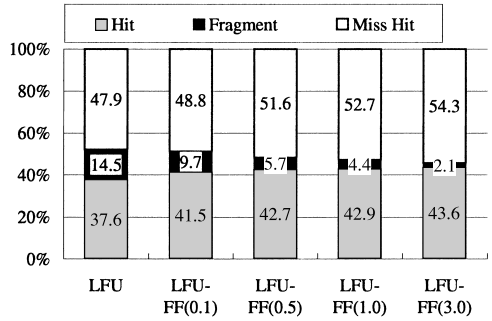


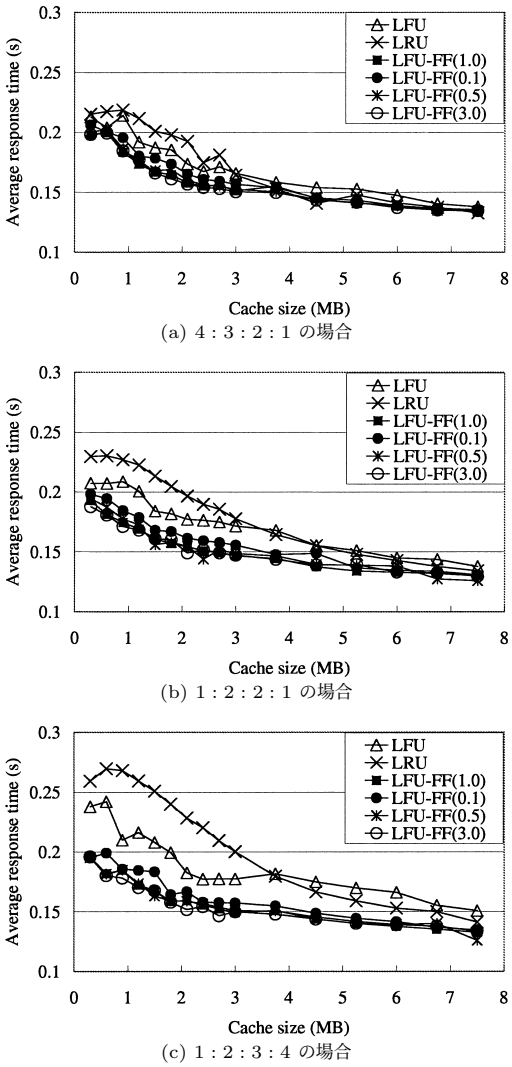
図 14 α のヒット率，フラグメント率，ミスヒット率への影響
Fig. 14 The impact of parameter α to hit ratio, fragment ratio, and miss hit ratio.

する．結果を**図 14** に示す．α の値を増加させることで，Fragment Factor の効果によりフラグメント率が削減していくことが確認できるが，$\alpha \geq 0.5$ では，フラグメント率の減少が収束していることが確認できる（Fragment Factor の効果を十分得ている状態）．これから，α の値は，0.5 から 3.0 程度が適切であることが分かる．

4.2.3 検索エリアサイズの影響

LFU-FF におけるクライアントの検索エリアサイズの比率の影響を調べてみた．具体的には，クライアントの検索エリアサイズの比率を，$(1 \times 1) : (2 \times 2) : (4 \times 4) : (8 \times 8)$= (a) 4 : 3 : 2 : 1，(b) 1 : 2 : 2 : 1，(c) 1 : 2 : 3 : 4 と変化させて，その応答時間を計測した．(a) は，狭いエリアへの要求が多い環境を想定したモデルであり，(c) は，広いエリアへの要求が多い環境を想定したモデルである．(b) は，(a) と (c) の中間のエリアに要求が集中する環境を想定している．これらのモデルにおける LFU-FF の効果について評価する．

結果を**図 15** に示す．図 15 (a) では，$\alpha \geq 0.5$ 時の LFU-FF が良い性能を示していることが分かる．この環境では，狭いエリアへのアクセスが多いために検索エリア内の断片化があまり発生しないため各手法の差はあまり確認することができない．図 15 (b) や (c) では断片化がより多く発生する環境であるために断片化を考慮していない手法（LRU や LFU）の性能は LFU-FF と比べて悪くなっている．この環境においても，(a) の場合と同様に $\alpha \geq 0.5$ 時の LFU-FF は良い性能を示している．



(a) 4 : 3 : 2 : 1 の場合

(b) 1 : 2 : 2 : 1 の場合

(c) 1 : 2 : 3 : 4 の場合

図 15 検索エリアサイズの応答時間への影響
Fig. 15 The impact of the distribution of the size for retrieval area to average elapsed time.

5. 関 連 研 究

有限なキャッシュ領域においてキャッシュデータの格納または削除を行うキャッシュ置換アルゴリズムとしては，WWW システムを対象とした研究が数多く行われている[5)~11)]．文献 5) では，ページサイズを均一と考え，ページを獲得するためのコストを考慮してキャッシュを置き換える Greedy-Dual 法が提案されている．Greedy-Dual 法をベースにして，不均一なページサイズを考慮した Greedy-Dual-Size（GDS）法[6)] や，ページの参照頻度を考慮した Greedy-Dual-Size-frequecy 法[8)] なども提案されている．また，LFU 法をベースにした置換アルゴリズムも提案されており，LFU 法に時間の概念を導入した LFU-AF 法[8)] や，キャッシュのスコア計算時に経験的な要因を取り入れた LRV 法[10)] などがある．これらの手法では，WWW 上のコンテンツを対象にしており，周辺情報をプロキシ上でキャッ

シュすることに関しては考慮されていない．このために，断片化が発生するという点では LRU 法や LFU 法と同様である．

周辺情報（位置依存情報）を扱うキャッシュについての研究も行われている[12),13)]．文献 12) は，クライアント側にキャッシュを置くシステムであり，クライアントの移動性に着目しキャッシュを置き換えることで，キャッシュによる性能の向上を達成している．また，文献 13) は，カーナビゲーションシステムを対象としたキャッシュシステムであり，移動計画と呼ばれるクライアントの行動予測と，周辺情報の有効範囲（スコープ）を用いてキャッシュの置き換えを行っている．これらのシステムでは主にクライアント側のキャッシュシステムに注目しており，プロキシ上でのキャッシュについては考慮されていない．

6. おわりに

本研究では，周辺情報獲得システムのためのプロキシキャッシュシステムの試作を行った．また，キャッシュデータの断片化を抑えるために，アクセス頻度に加えて Fragment Factor を導入しスコアを算出するキャッシュ置き換えアルゴリズム（LFU-FF 法）を提案し，試作したプロキシシステムへの実装と提案手法の性能評価を行った．実験の結果，提案した LFU-FF 法ではキャッシュサイズが全体の 8%のときに，LFU と比較して応答時間が約 21% 向上し，最も効果的であるという結果を得た．また，断片化が発生しにくいクライアントの要求や，発生しやすい要求に対しても LFU-FF は良い性能を得ることを確認した．本実験では周辺情報として施設情報を用いたが，地図情報，イベント情報などのその他の位置情報を持つ情報に関しては今回の実験と同様に利用することができる．また，本研究では都市 A の情報を用いたが，ベキ法則に従うその他の都市についても，分布が同じ傾向であるために，提案方式を適用できると考えられる．

今後の課題としては，本研究では，N 字エリアの境界をまたぐような検索の場合は，同位または下位の N 字エリアを組み合わせて，検索エリアを構成することを考えている．このため，余分なエリアを取得することによる通信量の増加と，複数のエリアを要求する必要があるために通信効率が悪化する可能性がある．このために，N 字エリアの境界をまたぐような検索が本システムに与える影響の評価（N 字アドレッシングを採用しない場合との比較）と，N 字エリアの境界をまたぐような検索に対して有効な手法の提案を行っていく予定である．また，多くのクライアントを用いて本システムのスケーラビリティの評価，木構造のメンテナンスにかかる計算量を削減するアルゴリズムの提案，サーバキャッシュへの適用，実際の環境での運用などを考えている．

謝辞 本研究の一部は，文部科学省科学技術研究費若手研究（B）課題番号 15700061 の助成による．

参考文献

1) MapFan Web. http://www.mapfan.com
2) VISE. http://www.vise.jp/viseILS/vise.jsp
3) Kiwi-W Consortium: Input for ISO Physical Storage Format, 28 March 2000. http://kiwi-w.mapmaster.co.jp/
4) 平松 薫：地図を用いた web ページ検索システムのログ解析，情報処理学会研究報告，2002(115), pp.217–224 (2002).
5) Young, N.: On-line caching as cache size varies, *Proc. 2nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pp.241–250 (1991).
6) Cao, P. and Irani, S.: Cost-Aware WWW Proxy Caching Algorithms, *Proc. USENIX Symposium on Internet Technologies and Systems*, pp.193–206 (1997).
7) Jin, S. and Bestavros, A.: Popularity-Aware Greedy Dual-Size Web Proxy Caching Algorithms, *Proc. Int'l Conf. on Distributed Computing Systems* (*ICDCS 2000*), pp.254–261 (2000).
8) Dilley, J. and Arlitt, M.: Improving Proxy Cache Performance — Analyzing Three Cache Replacement Policies, HP Labs Technical Report HPL-1999-142 (1999).
9) Jiang, S. and Zhang, X.: LIRS: An Efficient Low Inter-reference Recency Set Replacement Policy to Improve Buffer Cache Performance, *Proc. Int'l Conf. on SIGMETRICS*, pp.31–42 (2002).
10) Rizzo, L. and Vicisano, L.: Replacement policies for a proxy cache, *IEEE/ACM Trans. Networking*, Vol.8, No.2, pp.158–170 (2000).
11) Dilley, J., Arlitt, M. and Perret, S.: Enhancement and validation of squid's cache replacement policy, HP Laboratories, Palo Alto, CA, May 1999, pages HPL–1999–69 (May 1999).
12) Ren, Q. and Dunham, M.H.: Using Semantic Caching to Manage Location Dependent Data in Mobile Computing, *Proc. Int'l Conf. on Mobile Computing and Networking* (*MobiCom'00*), pp.210–221 (2000).
13) Sato, K., Saisho, K. and Fukuda, A.: A Cache System of Location Dependent Data for a Mobile Computer with Mobility Specification,

Proc. Int'l Conf. on Parallel and Distributed Processing Techniques and Application, Vol.2, pp.977–983 (1999).

(平成 15 年 9 月 16 日受付)
(平成 16 年 9 月 3 日採録)

田頭 茂明（正会員）

1996 年龍谷大学理工学部卒業．1998 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．2000 年同大学院情報科学研究科博士後期課程修了．博士（工学）．現在，広島大学大学院工学研究科助手．システムソフトウェアの研究に従事．1999 年第 14 回電気通信普及財団賞入賞．IEEE Computer Society 会員．

吉岡 浩路

2003 年広島大学工学部第二類卒業．同年パナソニック MSE 株式会社入社．現在に至る．ネットワークシステムに興味を持つ．

藤田 聡（正会員）

1985 年広島大学工学部第二類（電気系）卒業．1990 年同大学院博士課程修了．工学博士．現在，広島大学大学院工学研究科助教授．この間，カナダサイモンフレーザー大学客員研究員（1995），パリ南大学客員研究員（1996）．並列・分散アルゴリズム，組合せ最適化問題に興味を持つ．電子情報通信学会，日本応用数理学会，IEEE CS，SIAM 各会員．