キャッシュを使った頻出アイテムの抽出

吉田　健一[†a)]　勝野　聡[††]　藤田　昌克[†††]　鶴　正人[††††]
阿野　茂浩[††]　山崎　克之[††]

LRU is not better than Random!

Kenichi YOSHIDA[†a)], Satoshi KATSUNO[††], Yoshikatsu FUJITA[†††],
Masato TSURU[††††], Shigehiro ANO[††], and Katsuyuki YAMAZAKI[††]

あらまし　バスケット解析の基本手順の一部である頻出アイテムの抽出をオンラインで高速に行うための手法としてメモリキャッシュを使う方法を提案する．また，この手法の動作特性を解析し，頻出アイテム抽出のためのキャッシュ管理にはLRUよりもランダムベースのもので性能のよいものがあることと，データがZipfの法則に基づく場合ランダムベースのアルゴリズムが通常のヒット率においてもLRUより優れていることを示す．これは，ここ数十年標準的なキャッシュ管理の方法として用いられてきたLRUまたはLRUベースの手法に対して，randomベースの手法が有望なことを示唆している点で注意を要する．

キーワード　キャッシュ，LRU，頻出アイテム

1. まえがき

大量のオンラインデータからリアルタイムでデータマイニング[2]を行おうとする試み[7]の重要性が高まっている．我々は，この1方法としてキャッシュを使うアプローチ（CPM：Cache-based Pattern Mining）を検討している．本論文では，この過程で頻出アイテムの抽出をキャッシュを使って行う方法の性能を評価したので，結果を報告する．

オンラインデータからリアルタイムで頻出アイテムを抽出する方法としては，Coarse counting [6]やSticky sampling, Lossy Counting [11], Hashを使う方法[4],[8]，group testを使う方法[5]などが提案されている．筆者らが検討中のオンラインデータからリアルタイムで頻出アイテム（及び頻出する「アイテムの組合せ」）を抽出する方法は固定サイズのキャッシュを使うアプローチで，高速な処理の実現を目標としている．研究の過程で「頻出アイテムの抽出のためのキャッシュの管理にはLRU [10]よりもランダムベースのもので性能のよいものがあること」と「ランダムベースのアルゴリズムが通常のヒット率においてもデータの特性によってはLRUより優れていること」が分かったので，報告する．

最近の研究で，WWWやspam mailなどZipfの法則[1]に従うデータが多いことが分かってきた（例えば[12],[14]）．また，キャッシュのヒット率の研究を行う場合，stack distance [3]と呼ばれる方法で合成したデータ（以下stack distance dataと記す）を使うことが多い．我々の行った実験では，本論文で提案するキャッシュ管理の方法（random2と呼ぶ）はデータがZipfの法則に従う場合，ヒット率及び頻出アイテムの抽出性能ともに高い．stack distance dataにおいても頻出アイテムの抽出性能においてLRUをしのいでいる．

また，純粋なrandom replacementとLRUのキャッシュヒット率の差も，データがZipfの法則に従っていたり，stack distance dataであった場合，大きくな

† 筑波大学大学院ビジネス科学研究科，東京都
Graduate School of Business Science, University of Tsukuba, 3-29-1 Otsuka, Bunkyo-ku, Tokyo, 112-0012 Japan
†† KDDI研究所，上福岡市
KDDI R&D Laboratories Inc., 2-1-15 Ohara, Kamifukuoka-shi, 356-8502 Japan
††† 松下電器産業株式会社，東京都
Matsushita Electric Industrial Co., Ltd., 2-13-10 Kyobashi, Chuo-ku, Tokyo, 104-0031 Japan
†††† 九州工業大学情報工学部電子情報工学科，飯塚市
Department of Computer Science and Electronics, Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology, 680-4 Kawazu, Iizuka-shi, 820-8502 Japan
a) E-mail: yoshida@gssm.otsuka.tsukuba.ac.jp


```
Create empty heap;
while (input item) do
  i = index of item in heap;
  increment heap_cnt[i] by 1;
  if (heap_cnt[i]>thresh_hold)
    print message;
done
```

図 1　キャッシュを使った頻出アイテム抽出アルゴリズム
Fig. 1　Cache-based pattern mining algorithm.

い．これらは，ここ数十年標準的なキャッシュ管理の方法として用いられてきたLRUまたはLRUベースの手法（例えばLFU [10]，LRU-k [13] や 2Q [9]）に対して，randomベースの手法が有望なことを示唆している点で注意を要する．

2. CPM：Cache-based Pattern Mining

我々は大量のオンラインデータからリアルタイムで頻出するアイテムの組合せを抽出する方法として，固定サイズのキャッシュを使う方法を検討している．「頻出するアイテムの組合せ（itemset）」ではなく，「頻出するアイテム（item）」の抽出に限定すればアイデアは単純であり，固定サイズのキャッシュを使って処理対象のデータに含まれるアイテムの数を数え，設定値以上の回数出現したアイテムを頻出アイテムとして出力する（図 1 に擬似コードを示す）．

頻出するアイテムの組合せを検出する場合，キャッシュの構造が若干複雑になる（図 2）が，基本的には同じ考え方が使える．すなわち，初期は頻出アイテムの抽出を行っておき，頻出すると判断されるアイテムが見つかった後はその頻出するアイテムを含む二つのアイテムの組合せについても出現回数をチェックする．N 個のアイテムの組合せが頻出すると判断された後は，その N 個のアイテムの組合せを含む $N+1$ 個のアイテムの出現回数をチェックする．正確には処理速度をあげるための工夫などが必要となるが，本論文では以下アイテムの組合せ（itemset）ではなく，アイテム（item）の場合に限定して考える．

キャッシュの容量が十分大きい場合，アイテムの記憶位置を決める処理（図 1 の 3 行目）は単純である（過去に出現して記憶されているものは hash 関数などを用いて記憶位置を決定し，新しいアイテムであれば空きエリアを記憶位置とする）が，容量が全アイテムを記憶するだけ確保できない場合，何らかの指標に基づき既にあるアイテムのデータを削除して，キャッシュメモリを再利用することを考える．

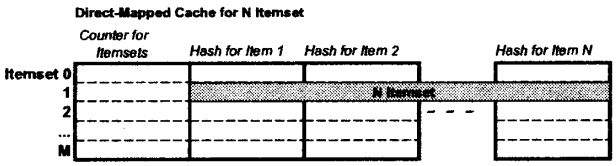


図 2　キャッシュを使った頻出アイテム抽出用データ構造
Fig. 2　Cache-based pattern mining data structure.

```
int i = random() % HEAP;
for (p=1; (heap_cnt[i]>p); p++)
   i = random() % HEAP;
```

図 3　Random 2 の C プログラムコード
Fig. 3　Random2.

LRUは，このようなときに用いられる標準的な方法であるが，次章の実験ではLRUのほかにrandom（削除する古いデータの記憶位置を乱数で決める方法．より正確には乱数で発生した数をキャッシュサイズで割った余りを求め，データの記憶位置とする）とrandom2（図 3）等の性能を比較する．

random2は[14]にてSPAM filterを作るときのキャッシュ管理の方式として優れた性能をもつことが報告されている方法で，図 3 は次章の実験で実際に使ったCによるrandom2のプログラムコードそのものである．1 行目は random そのもの，2 行目と 3 行目は複数回出現したデータをなるべく捨てないようにしている処理を実現しているが，基本的にはこの修正をした後も，乱数で決めた数が新しいアイテムの記憶位置（削除するデータの記憶位置）となっている．

乱数で候補を選ぶという点で random は無作為抽出を計算機で実現した 1 手法である．一方 LRU も random2 も候補の選択にあたって「作為がある」ので，「作為の有無」の観点からは，random2 は random よりも LRU に近い．逆に実装を考えた場合，random2 は LRU よりも random に近い．すなわち，LRU や LFU などを実装する場合，エントリの挿入，削除などを効率的に実施するため双方向のリスト構造を実装するのが一般的であるが，実装にはそれなりの技術/工数を必要とする．3 行で済む random2 の単純性は「random に近い」ことからくる特徴で，実装を考えた場合の大きな優位点となっている．

3. 実験結果

本章ではrandom2, random, LRU, Simplified 2Q [9], Lossy Counting [11] の性能比較を行う．性能比較においては，キャッシュメモリの容量を変化させながら次の三つの指標を比較した．

（1） 通常のキャッシュヒット率
（2） 見つけた頻出アイテムの数
（3） 同じ頻出アイテムを別物と誤認した数

(2) は筆者らの本来の研究目標であるオンラインデータからリアルタイムで頻出アイテムを抽出する性能を測ったもので，大きい方がよい．(3) はキャッシュが小さいときに1種類の頻出アイテムを複数の頻出アイテムと誤認した数を数えたもので，小さい方がよい．

Simplified 2Q [9] は計算機の仮想記憶やデータベースのバッファを制御するために考案された方法で，頻出するページやデータを管理するためのLRUと1回のみ現れたページやデータを管理するためのFIFOの二つの記憶領域を利用する．全メモリのうち何%をLRUに何%をFIFOに割り当てるかはチューニングパラメータであるが，以降の実験では実験に利用するメモリ容量ごとにFIFOが使うメモリの容量を全メモリのうちの2%，4%，8%と変化させて比較した．

Lossy Counting [11] は頻出アイテムの抽出を目的に提案された手法で，キャッシュに記憶しているアイテム情報の削除をある基準で行うことで，指定した値 ϵ 以下の誤差内で頻出アイテム（やはり指定した値 s 以上出現するアイテム）の抽出を可能とする方法である．本来頻出アイテム検出用の手法であるが，キャッシュにアイテムを記憶しておくので，通常のキャッシュヒット率を計算することもできる．ただし，他の方法のように使用するキャッシュメモリの容量を事前に指定することができないため，ϵ の値を変える（大きな誤差を許せばメモリ使用量が減る）ことで比較を行った．

3.1 Zipfの法則に基づくデータ

Zipfの法則とは，f を英文中に現れる単語の使用度数，r を使用度数の大きい方から振った順位，C，k を定数としたときに，

$$fr^k = C$$

が成り立つという経験則で，図4に $k = -0.8$ の場合の分布を示す．最近の研究で，WWWやspam mailなどがZipfの法則に従うことが分かってきた（例えば[12], [14]）という点から，データマイニングのプログラムの性能を測る上で重要な性質をもったデータである．

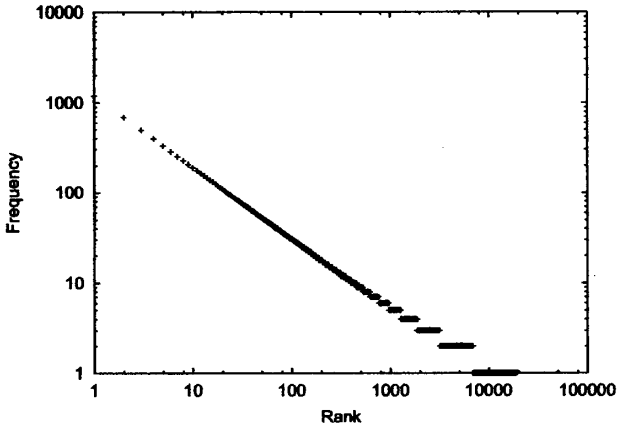


図4 Zipfの法則に従うデータの分布
Fig. 4 Distribution of Zipfian data.

図5，図6，図7に k を -0.8，-1.0，-1.2 として長さ（含まれるアイテムの数）が10,000,000のデータを人工合成し，更にキャッシュ容量を変化させて各値を計測した場合のキャッシュヒット率，見つけた頻出アイテムの数，同じ頻出アイテムを別物と誤認した数を示す．また，このとき，100回以上現れたアイテムを頻出アイテムとした．k の値によって若干結果のグラフは異なるが，データの分布がZipfの法則に従う場合，次のことがいえる．

- random2, LRU, randomの中では k の値によらずキャッシュヒット率（Cache Hit）はrandom2, LRU, randomの順によい（値が大きい）．

手法による性能の差はあまり明確でない．特にLRUとrandomの性能の差は大きくない．

- 見つけた頻出アイテムの数（Number of Frequent Items）もrandom2, LRU, randomの順によい（値が大きい）．

キャッシュヒット率に比較して手法による性能の差は大きい．

- 同じ頻出アイテムを別物と誤認した数（Number of Pseudo Items）もrandom2, LRU, randomの順によい（値が小さい）．キャッシュヒット率に比較して手法による性能の差は大きい．

特にrandom2は「同じ頻出アイテムを別物と誤認しない」結果となっており，頻出アイテムをminingするための手法として適している．

- Simplified 2Qはrandom2と同等の性能を示しているがパラメータが必要である（詳細は **3.4** 参照）．

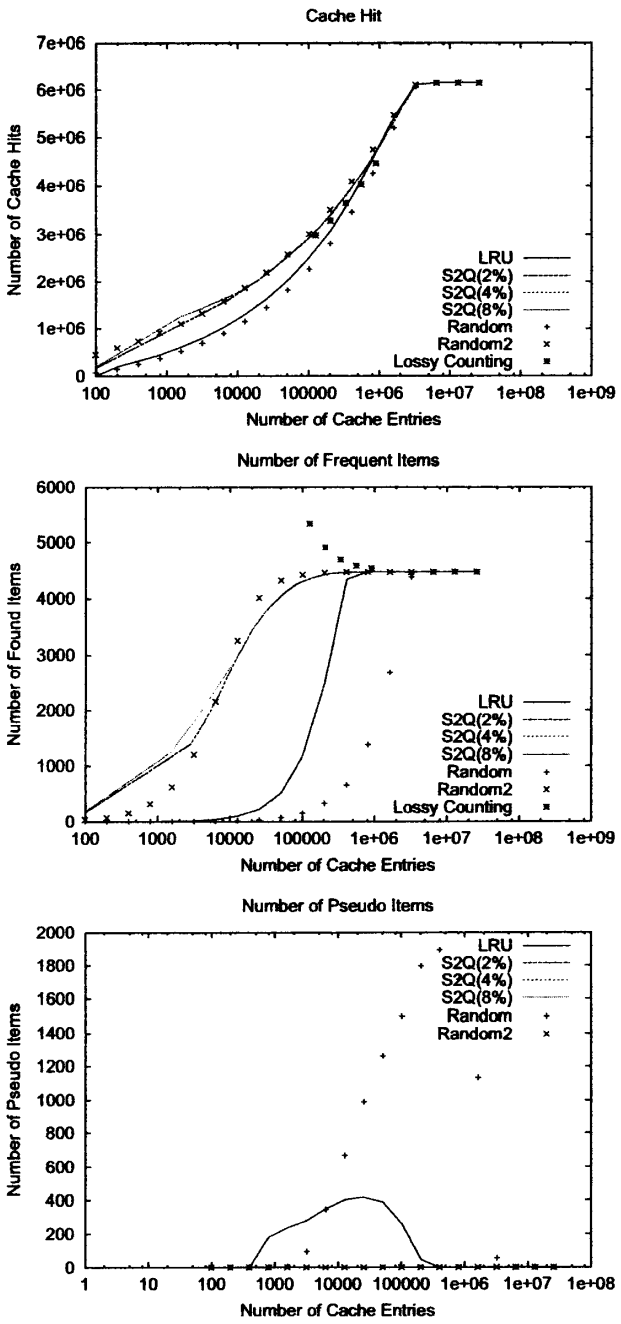


図 5　$k = -0.8$ の場合
Fig. 5　Results with $k = -0.8$.

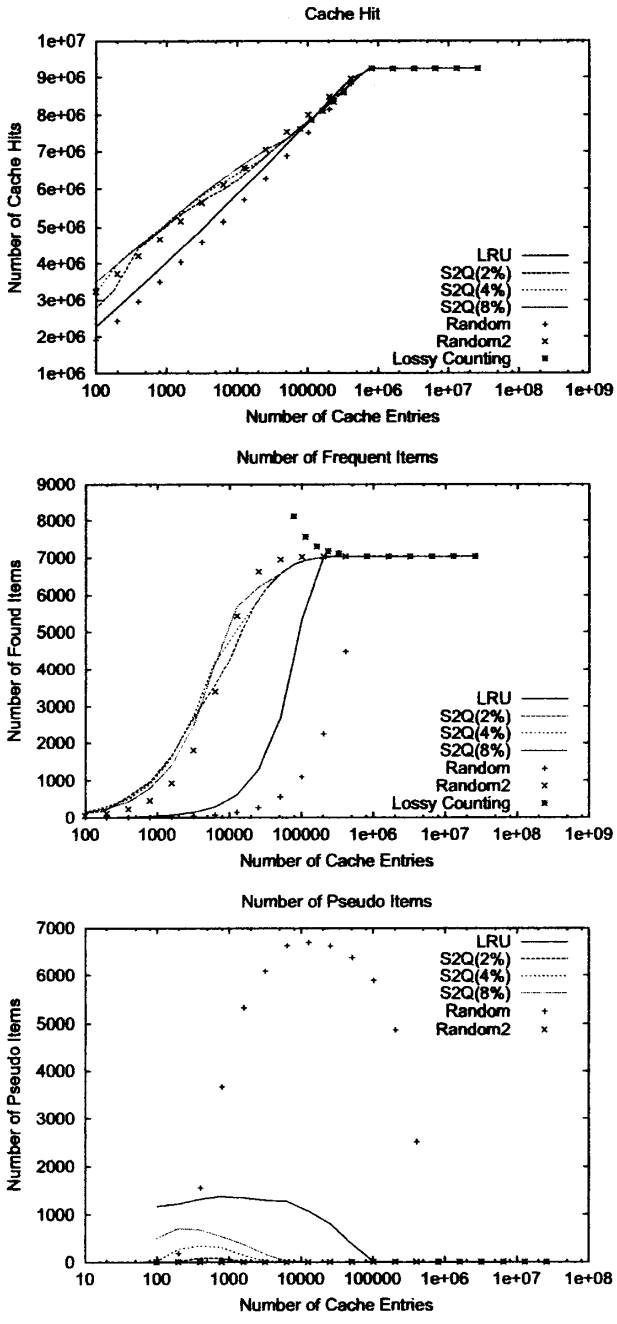


図 6　$k = -1.0$ の場合
Fig. 6　Results with $k = -1.0$.

- Lossy Counting は他の手法が小さなメモリ容量のときに指定回数出現したアイテムを頻出アイテムと認識できないのと異なり，使用するメモリ容量が小さい場合に指定回数出現しないデータも頻出アイテムと誤認する．使用するメモリ容量が小さくなっていった場合の誤認識率の上昇は著しく random2 や Simplified 2Q がすべての頻出アイテムをもれなく検出できる容量であっても 10%以上の誤認識を示すメモリ容量域がある．

この実験結果について，我々は次のような定性的な理解をしている．

- 大ざっぱにいって，Zipf の法則に従うデータは，非常に出現回数の多い少数のデータと，1 回しか出現しない非常に多数のデータに 2 分される．
- random2 は複数回出現したデータをなるべく残すような仕組みをもっているが，この仕組みが非常に出現回数の多いデータをもつデータ分布とうまくかみ合い random2 がよい結果を残している．

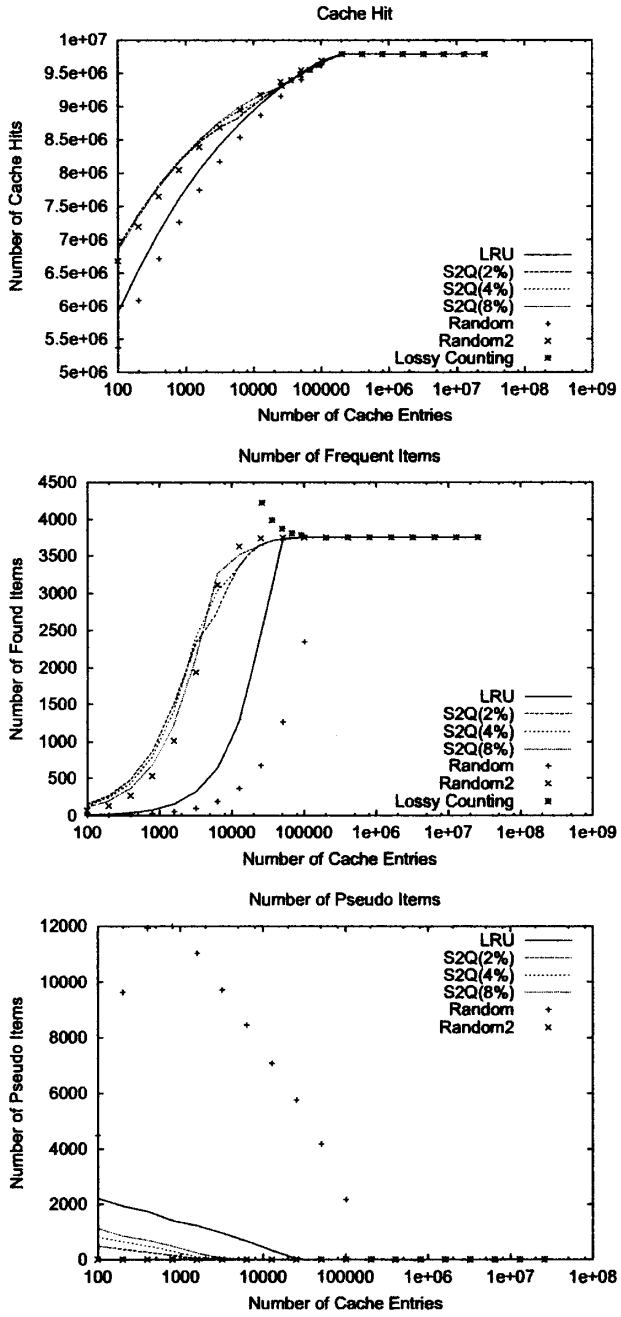


図 7 $k = -1.2$ の場合
Fig. 7 Results with $k = -1.2$.

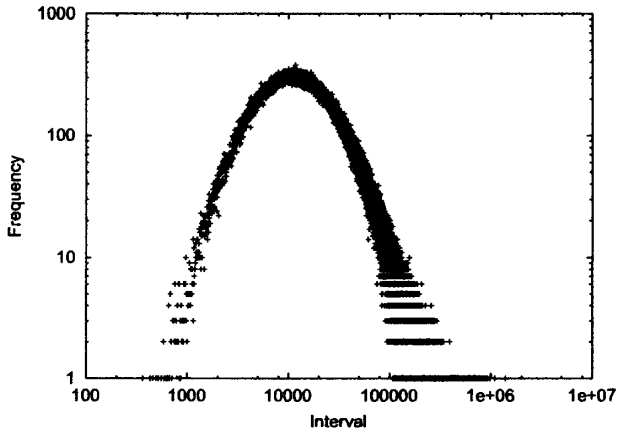


図 8 データ到着間隔の分布
Fig. 8 Interval of Stack Distance data.

• Simplified 2Q も複数回出現したデータをなるべく残すような仕組みをもっているが，パラメータが必要である．[9] ではパラメータの自動チューニング方法も提案されているが，別途チューニング用のメモリ領域を必要としている．したがって計算機の仮想記憶のように記憶しておくべきデータ（仮想記憶の場合，4 kByte 程度のページ単位）が管理領域より大きな場合でないと，オーバヘッドが大きくなり意味をなさない．random2 はこのようなオーバヘッドなしに同程度の性能をもつ点で優れている．

• LRU は極端にキャッシュ容量が小さい場合，たまたまある頻出アイテムの出現間隔が開いた場合にキャッシュからその頻出アイテムを削除してしまう．このため random2 が性能で LRU を上回る．

• 非常に出現回数の多い小数のデータがある場合，キャッシュヒット率の観点からは random でも LRU に近い性能を示すが，random は数を数えるという操作を伴う頻出アイテムの抽出には十分ではない．

• Lossy Counting は他の手法と異なり少ないメモリ容量でも頻出アイテムを見逃さないという点で，他の手法にはない特徴をもっているが，使用可能なメモリ容量が小さい場合の誤認率は高く利用は適さない．

3.2 Stack Distance に基づくデータ

キャッシュのヒット率の研究を行う場合，stack distance [3] と呼ばれる方法で合成したデータ（stack distance data）を使うことが多い．この方法はデータの到着間隔の分布が，特定の分布に従うと仮定するもので，同じ WWW ページにアクセスする間隔が対数正規分布に従うことなどが知られている [3]．図 8 に実験でデータ作成に用いたデータの到着間隔の分布を示す．この仮定で合成されたデータは Zipf の法則に従わず（図 9 に分布を示す），実際のデータとは異なった特徴ももっているが，キャッシュヒット率を考える上で標準的なデータであるので，実験結果を図 10 に示す．生成したデータの長さは同じく 10,000,000 で 100 回以上出現したアイテムを頻出アイテムと判断した結果：

• stack distance data は 1 回しか出現しない非常に多数のデータをもつが，非常に出現回数の多いデー

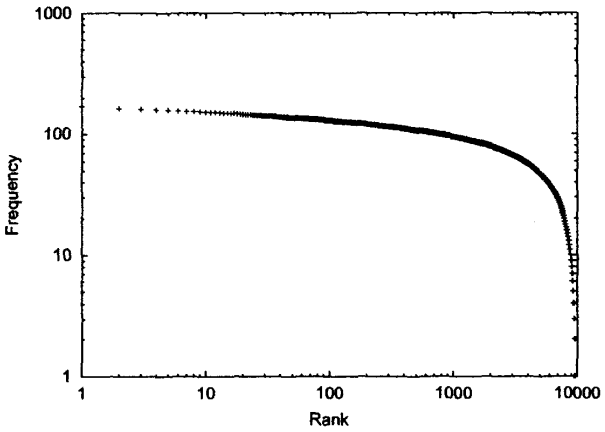


図 9　データの分布
Fig. 9　Distribution of Stack Distance data.

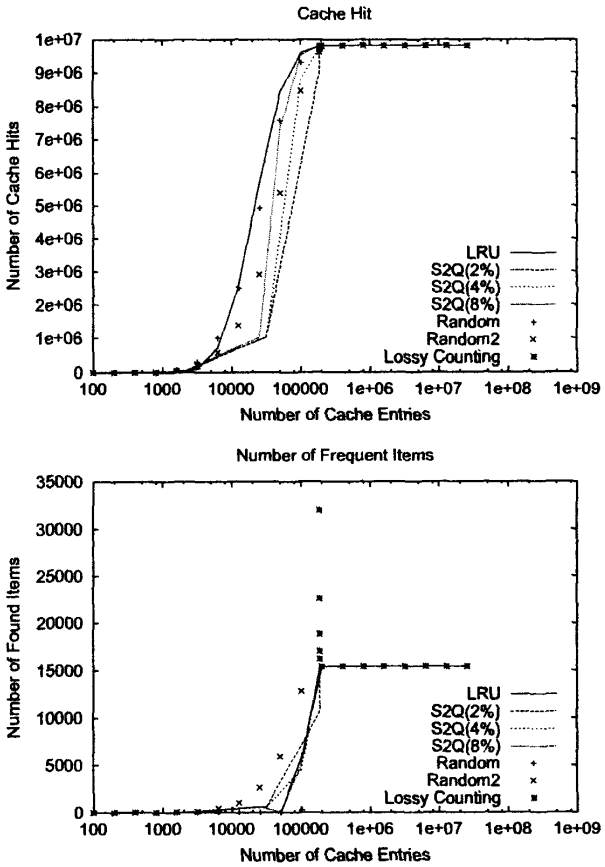


図 10　Stack Distance に基づくデータの結果
Fig. 10　Resutls on Stack Distance data.

タをもたない．

- このためヒット率については複数回出現したデータをなるべく残すような仕組みをもった random2 と Simplified 2Q が悪くなるが，頻出アイテムを見つける性能は random2 と Simplified 2Q がよい．
- いずれの方法も同じ頻出アイテムを別物と誤認はしなかった．

LRU の改良版として提案された Simplified 2Q のキャッシュヒット率が LRU より悪くなっているのは，2Q の研究の前提と今回の研究の前提が異なっているためと思われる．すなわち今回の実験では従来の計算機の仮想記憶の研究等と比べて極端にキャッシュメモリの容量が小さい場合まで性能の比較を行っている．これは研究の目的としている大量のオンラインデータからのデータマイニングでは，用意できる計算機の容量に比較し極端に大きなデータを扱う必要があることを考慮したためである．計算機の仮想記憶の場合，特にプログラムコードを対象とすれば，用意できるメモリ容量はプログラムコードの量に比較し今回実験したほどは異ならない．研究が前提としている計算機資源と対象データの大きさの差が，このような結果をもたらしたものと思われる．

3.3　実データの結果

図 11，図 12，図 13 は，それぞれ WWW ページのアクセスログ，mail のログ，インターネットバックボーンの header ログについて，random2，rondom，LRU，Simplified 2Q，Lossy Counting のキャッシュヒット率，見つけた頻出アイテムの数，同じ頻出アイテムを別物と誤認した数を計測（すべていったんハードディスクに記憶したものをオフラインで解析）した結果である．

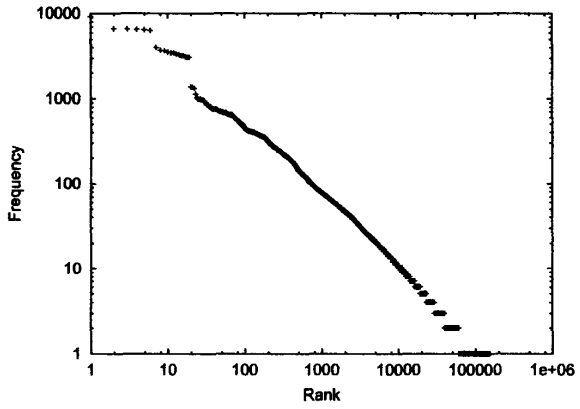
- WWW ページのアクセスログは WWW サーバに対するアクセスの記録であり，約 141 万アクセス分のものである．100 回以上アクセスされたページを頻出アイテムとして性能を評価した．

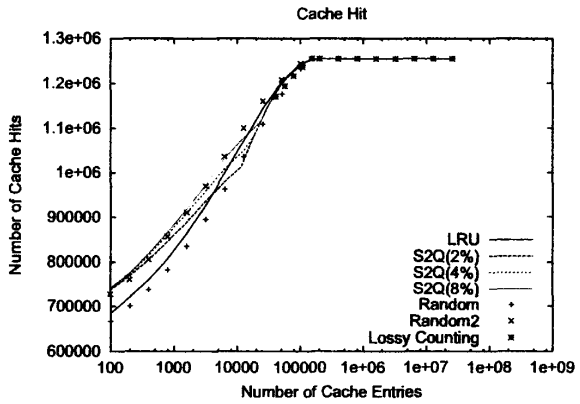
WWW ページのアクセスログから多数回アクセスされたページを発見することは，人気のあるページの mining に相当する．また，この情報は proxy cache の性能向上のために利用できることも報告されている [12].

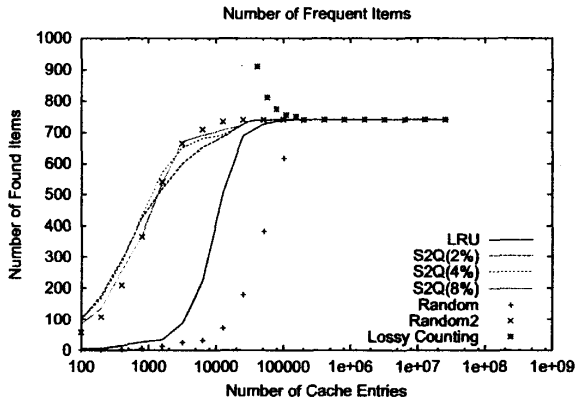
- mail のログは，実際の mail 本文の MD5 値を 1000 万通分計算したもので，これも 100 回以上同じ MD5 値のものが出現した場合を頻出アイテムとして性能を評価した．

mail のログから多数回出現する mail を発見することで性能のよい spam filter が作れることが報告されている [14].

- インターネットバックボーンの header ログは，バックボーンに流れるデータの header から src ip address，dst ip address，src port，dst port の情報を切り出して hash 化したもの約 1 分分 164,384,485 個







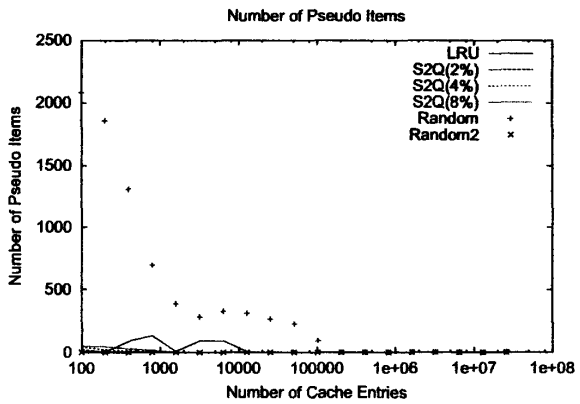
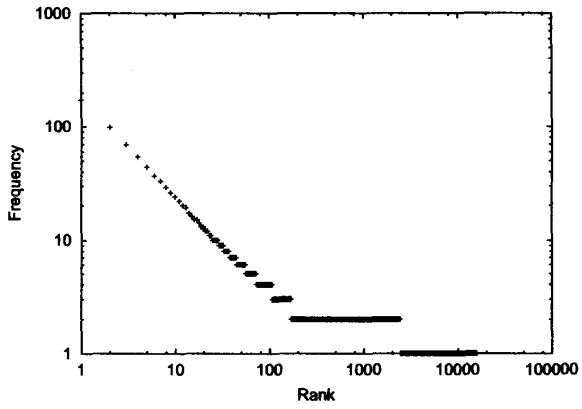
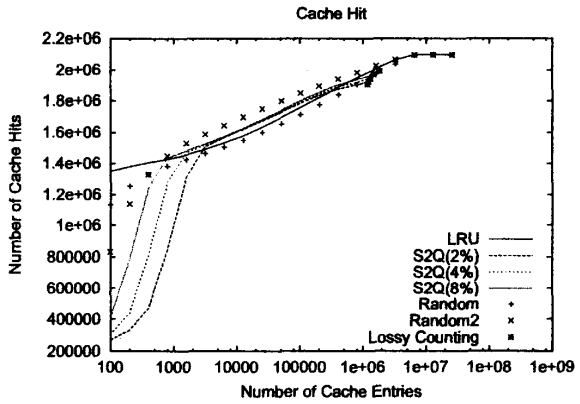
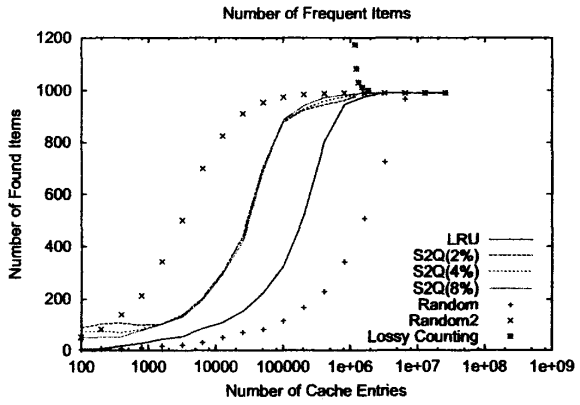


図 11　WWW のアクセスログ
Fig. 11　Resutls on WWW log.







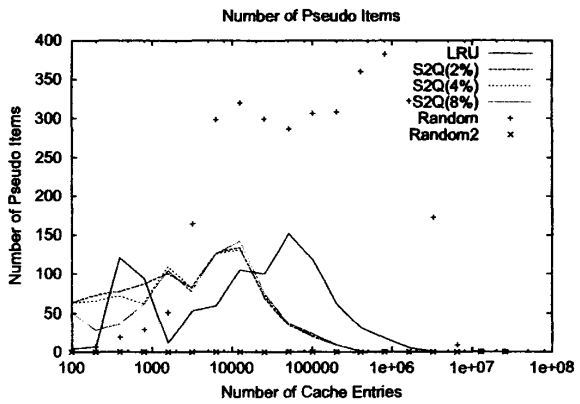


図 12　mail のログ
Fig. 12　Resutls on mail log.

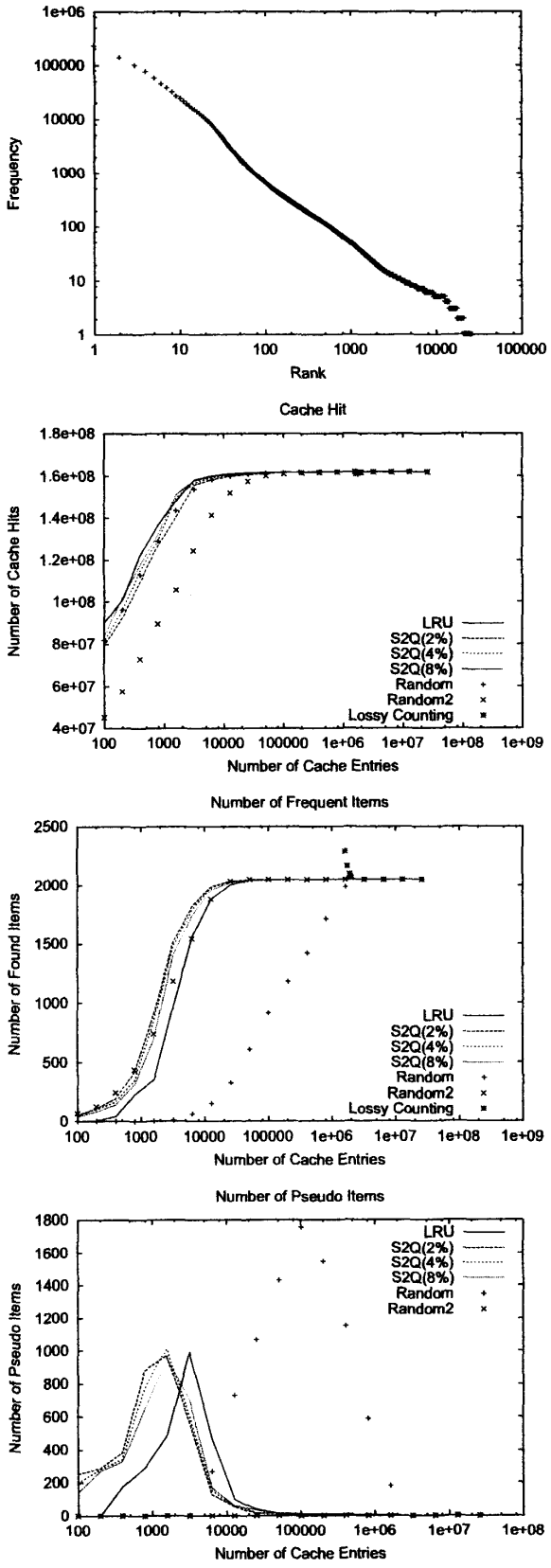


図 13　インターネットバックボーンの header ログ
Fig. 13　Resutls on header log.

で，10,000 回以上出現したものを頻出アイテムとして性能を評価した．
これは P2P 等の巨大フローを発見することに相当する．

以上の実データに対して，random2，LRU の性能を比較した結果，以下のことがいえる．

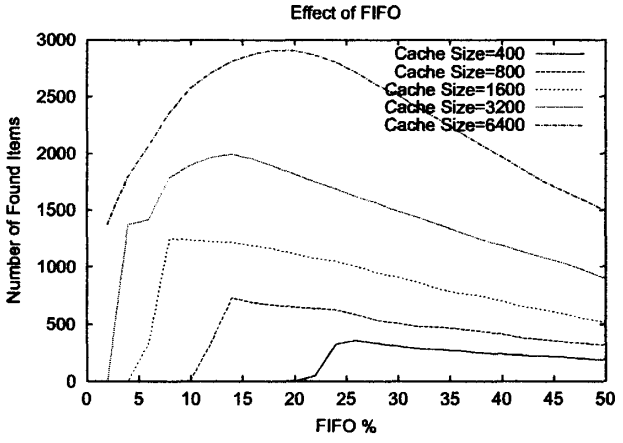
- WWW のアクセスログは Zipf の法則に基づくデータとして扱え，ヒット率，頻出アイテムの発見ともに random2 の性能がよい．LRU を使う理由は見当たらない．
- インターネットバックボーンの header ログは，stack distance data に近い結果をもつ．頻出アイテムの発見には random2 がよいが，通常のヒット率の観点からは LRU がよい．
- mail のログは両者の中間で，極端にメモリ容量が小さい場合のヒット率は LRU がよい．
- メモリが小さい場合に Lossy Counting の誤認識率が急速に悪くなる点，Simplified 2Q のパラメータ調整が難しい点，の 2 点は実データにおいても確認された．

ここで取り上げたデータはいずれもインターネット上のデータとして盛んに解析されている対象のデータであり，また頻出アイテムの発見は実用上の意味をもっていることから，ここ数十年標準的なキャッシュ管理の方法として用いられてきた LRU または LRU ベースの手法に対して，random2 など random ベースの手法が有望なことを強く示唆している点で注意を要する．

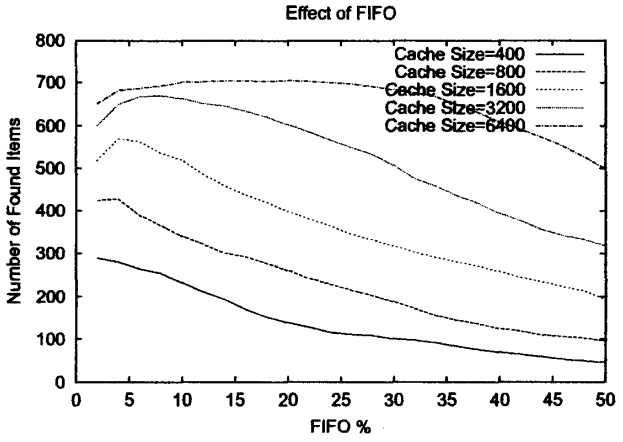
3.4 補足実験

Simplified 2Q は random2 と同等の性能を示しているがパラメータが必要である．このことを図 14 に示す．図 14 は，FIFO が使うメモリの容量を全メモリのうちの 2%から 50%まで 2%刻で変化させて頻出アイテムの検出性能を見たもので，対象は $k = 0.8$ で人工合成した Zipf の法則に従うデータと，WWW のアクセスログである．

図 14 から明らかなように，Simplified 2Q はよい結果を残すためにはパラメータ調整を必要とし，最適パラメータはデータの種類，キャッシュメモリの容量により，それぞれ大きく異なる．また，この最適パラメータ値は自明ではない．ここまで見てきたように random2 は Simplified 2Q とほぼ同等の性能をもつがパラメータ調整が不要であるという点で実用上優れた特徴をもっている．



Zipfian データの場合



WWW データの場合

図 14 Simplified 2Q のパラメータ調整
Fig. 14 Parameter tuning of Simplified 2Q.

4. むすび

バスケット解析の基本手順の一部である頻出アイテムの抽出をオンラインで高速に行うためにメモリキャッシュを使う方法（CPM：Cache-based Pattern Mining）と，複数回出現したデータをなるべく捨てないようにしたメモリキャッシュ管理の方法 random2 を提案した．また，random2 の動作特性を解析し，以下のことを明らかにした．

- データの分布が Zipf の法則に従う場合と，キャッシュの性能を評価するときに使われる代表的な方法である stack distance と呼ばれる方法で合成したもの（stack distance data と記す）である場合，頻出アイテムの抽出のためのキャッシュの管理性能は LRU よりも random2 が優れている．
- データの分布が Zipf の法則に従う場合，通常のヒット率においても random2 は LRU より優れている．
- データが Zipf の法則に従っていたり，stack distance data であった場合，純粋な random replacement と LRU のキャッシュヒット率の差は大きくない．
- 複数回出現したデータをなるべく捨てないように LRU を改良した Simplified 2Q，頻出アイテム抽出のために提案されている Lossy Counting，と比較すると，random2 はパラメータ調整が不要な点やメモリが小さい場合の誤認識率の点で優れている．
- WWW ページのアクセスログ，mail のログ，インターネットバックボーンの header ログについて，実データによる実験を行い，
  - WWW のアクセスログは Zipf の法則に基づくデータとして扱え，ヒット率，頻出アイテムの発見ともに random2 の性能がよいこと．
  - インターネットバックボーンの header ログは，stack distance data に近い結果をもつ．頻出アイテムの発見には random2 がよいが，通常のヒット率の観点からは LRU がよいこと．
  - mail のログは両者の中間で，極端にメモリ容量が小さい場合のヒット率は LRU がよいこと．
  - いずれの実データにおいても頻出アイテムの発見は random2 の性能がよいこと．

などを明らかにした．

これらの結果は，ここ数十年標準的なキャッシュ管理の方法として用いられてきた LRU または LRU ベースの手法に対して，random ベースの手法が有望なことを示唆している点で注意を要する．

謝辞 実データの解析に御協力頂いた KDDI 株式会社赤木篤志氏，中島昭浩氏，松下電器産業株式会社吉田純氏に感謝します．

文　献

[1] 水谷静夫，数理言語学，培風館，1982.
[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," Proc. 20th Int. Conf. Very Large Data Bases, VLDB, pp.487–499, 1994.
[3] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira, "Characterizing reference locality in the WWW," Proc. IEEE Conference on Parallel and Distributed Information Systems (PDIS), Miami Beach, FL, 1996.
[4] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," Proc. 29th International Colloquium on Automata, Languages, and Programming, 2002.
[5] G. Cormode and S. Muthukrishnan, "What's hot and what's not: Tracking frequent items dynamically,"

Proc. Principles of Database Systems, pp.296–306, 2003.

[6] M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J.D. Ullman, "Computing iceberg queries efficiently," Proc. 24th Int. Conf. Very Large Data Bases, VLDB, pp.299–310, 1998.

[7] C. Hidber, "Online association rule mining," Proc. SIGMOD Conf., pp.145–156, 1999.

[8] C. Jin, W. Qian, C. Sha, J.X. Yu, and A. Zhou, "Dynamically maintaining frequent items over a data stream," Proc. 12th International Conference on Information and Knowledge Management, pp.287–294, 2003.

[9] T. Johnson and D. Shasha, "2Q: A low overhead high performance buffer management replacement algorithm," Proc. Twentieth International Conference on Very Large Databases, pp.439–450, Santiago, Chile, 1994.

[10] D.E. Knuth, The Art of Computer Programming, vol.3 – Sorting and Searching, Addison-Wesley, 1973.

[11] G. Manku and R. Motwani, "Approximate frequency counts over data streams," Proc. 28th International Conference on Very Large Data Bases, pp.346–357, Hong Kong, China, 2002.

[12] N. Nishikawa, T.Y. Mori, K. Yoshida, and H. Tsuji, "Memory-based architecture for distributed www caching proxy," Proc. World Wide Web Conference 98, pp.205–214, 1998.

[13] E.J. O'Neil, P.E. O'Neil, and G. Weikum, "The LRU-K page replacement algorithm for database disk buffering," Proc. ACM SIGMOD International Conference on Management of Data, pp.297–306, 1993.

[14] K. Yoshida, F. Adachi, T. Washio, H. Motoda, T. Homma, A. Nakashima, H. Fujikawa, and K. Yamazaki, "Density-based spam detector," KDD2004, pp.486–493, 2004.

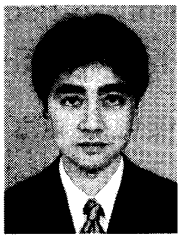
（平成17年1月11日受付，3月29日再受付）

吉田　健一　（正員）

1980東工大・理・情報科学卒，同年日立製作所入社．1992年9月博士（工学，大阪大学）．2002より筑波大学大学院ビジネス科学研究科教授．情報処理学会，人工知能学会各会員．

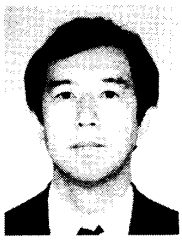
勝野　聡　（正員）

1989東大・工・電気卒．1991同大大学院修士課程了．同年国際電信電話（現，KDDI）（株）入社．2001通信・放送機構研究員．2004より（株）KDDI研究所主任研究員．



藤田　昌克　（学生員）

1984早大・理工・電気卒．同年松下通信工業（株）入社．現在，松下電器産業（株）eネット事業本部勤務．1994ジョージア工科大・計算機科学修士．1995筑波大学大学院・経営システム科学修士．現在，同大学博士後期課程在学中．



鶴　正人　（正員）

九州工業大学情報工学部助教授．1985京都大学大学院工学研究科了．沖電気工業(株)，長崎大学総合情報処理センター助手，日本テレコムインフォメーションサービス(株)，通信・放送機構研究員を経て，2003より現職．博士（情報工学）．

阿野　茂浩　（正員）

1989早稲田大学大学院理工学研究科了．同年，国際電信電話（株）入社．現在，(株)KDDI研究所IP品質制御システムグループリーダ．情報処理学会会員．

山崎　克之　（正員）

昭55電通大・通信卒．KDD（現KDDI）(株)においてISDN，SDH，ATM，IPの情報通信ネットワークの研究開発や国際標準化，などに従事．工博．現在，(株)KDDI研究所・研究戦略室長．