6B-5

Bloom フィルタを用いた分散 Web システムにおける共通アクセス履歴の効率のよい構築

楢崎修二 †
長崎大学工学部 ‡

narazaki@cs.cis.nagasaki-u.ac.jp

1 はじめに

Web システムのユーザビリティを改善する 1 つの方法としてアクセス履歴を用いたページ推薦やページの先読みがある．有効性の高い先読みを実現するには過去のアクセス履歴を元に遷移確率を計算する必要がある．そのために履歴情報を記録し利用することが必要になる．

またサーバを複数使うことで応答時間を改善する方法もある．この方法は規模によらず様々なシステムでサーバクライアントシステムとしての問題を解決するために使われている．しかし，サーバの分散の結果，ログ情報はシステムに散逸することになる．

従って，ユーザビリティを改善するこれらの手法は相性が悪い．そこで，本研究では Bloom フィルタを用いて分散されたログを効率よく収集し，アクセス履歴を構築する手法について検討を行なう．

2 対象とする分散 Web システム

我々の研究室では e-learning のような，小規模ではあるが短時間内に同じファイルへのアクセスが集中するようなアクセスが想定される Web システムのユーザビリティを改善するために Chord[3] と同様のリング状の順序関係を持つサーバ間での動的な複製を行う Web システムを提案し構築している [4].

このシステムではコンテンツ (Ajax によって流通する Web ページであるため以降では単にページと呼ぶ) は分散ハッシュ値に基づき置かれるサーバが決定されるが，一方でサーバは負荷に応じてコンテンツの複製をリング上で両隣りのサーバに置くようにふるまう [4]. これによりハッシュに基づくページの分散とサーバ負荷に基づくページの複製とを実現している．

さらに，より快適なユーザビリティを実現するため，過去のアクセス履歴から共通性の高いページ間の遷移確率を計算し，ページの先読みに反映させる機能も実装している．有効な先読みを行なうためには履歴情報から遷移確率を求めることが必要になる．図 1は実際のログから得られた遷移確率の高い遷移列の一例である．

しかし，コンテンツが複数のサーバに分散しているため，各ユーザの完全な履歴を構成するためには，現在のシステムでは各サーバのログファイルを一箇所 (マスターノード) に転送し，アクセス時刻順にログを合成する必要があった (ここでは，これを**大域ログ構成法**と呼ぶことにする)．ログには CGI を指す URL 中にセッション ID が残っているため，合成は，同じセッション ID を持つアクセスを時間順にならべてセッションごとの完全なアクセス履歴を作り，統計処理をすることで頻出する遷移部分列を返す手続きである．当然，この作業はテキスト情報であるサーバのログを全て転送することが必要となり通信路に負荷を掛けている．

3 Bloom フィルタによるログ圧縮

前項の問題を解決するために，転送量を削減する手法を二つ提案する．

分散遷移表合成法　大域ログ構成法において，合成によって構成されるデータは遷移元となるページからある遷移先ページへの遷移確率を表としたものである (これを遷移表と呼ぶことにする)．遷移表は遷移元ページから遷移先ページへの遷移回数を表現したものになるため，ログ取得時間に関わらず一定のデータ量となる．セッション ID は削除されるため，遷移表からは個々のアクセス履歴は再現できないが，記録に必要なデータ量は小さい．なお，遷移の偏りが期待できれば疎行列となるため，圧縮表現を使うことでさら必要なデータ量を減らすことができる．

そこでログテキストそのものを転送するのではなく，各 Web サーバ上でローカルに遷移表を構成し，転送したものをマスターノードで合成することが考えられる．これを**分散遷移表合成法**と呼ぶことにする．マスターノードでの合成は配列の各要素を合計する作業となる．

ただし，複数のサーバにまたがった個別のセッションの遷移履歴をマスターノードで合成された遷移表から再構成することは不可能であるという問題がある．もし各ページがそれぞれ分散ハッシュ値によって定まる唯一のサーバによってのみ提供されるなら，たまたま同一サーバ間におかれたコンテンツページ間での遷移のみが検出され，この手法の有効性はサーバ数に比例して減少することになるであろう．しかし 2 章で述べた対象システムではコンテンツの複製が起きる．簡単に言えば同時にアクセスがかかった人気のあるページほど複製されるため，先読みに値するページ間の遷移ほど同一サーバ内での遷移にも含まれるようになることが予想されるため，どの程度の検出率の低下が起きるかは実験での検証が必要である．

圧縮分散遷移表法　次に，Bloom フィルタを使うことでこの遷移表を非可逆圧縮し，さらに転送量を減らす手法を提

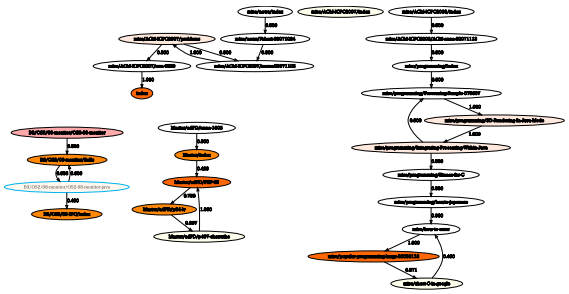

図 1: ログファイルによる訪問履歴に基づくページ遷移例

Effective Construction of Common Access History by Bloom filter in a Distributed Web System.
† Shuji Narazaki
‡ Nagasaki University, Faculty of Engineering

表 1: 検出率に関する実験結果

手法 (サイズ)	検出遷移数 0.25	0.33	0.4	0.5	遷移確率 0.5 0.25	0.33	0.4	0.5	遷移確率 0.25 0.25	0.33	0.4	0.5
大域 (469KB)	79	55	42	36	36 (.460)	36 (.65)	36 (.860)	36 (1.0)	79 (1.0)	53 (.964)	42 (1.0)	36 (1.0)
分散 (2.2KB)	93	85	73	71	6 (.065)	6 (.070)	6 (.082)	6 (.085)	14 (.151)	14 (.165)	14 (.192)	14 (.197)
圧縮 (160B)	87	76	71	60	6 (**.069**)	6 (**.079**)	6 (**.085**)	6 (.1)	15 (**.172**)	14 (**.184**)	14 (**.197**)	14 (**.233**)
圧縮 (320B)	90	79	71	60	6 (**.067**)	6 (**.076**)	6 (**.085**)	6 (.1)	14 (**.156**)	14 (**.177**)	13 (.183)	11 (.183)
圧縮 (640B)	89	79	68	61	6 (**.067**)	6 (**.076**)	6 (**.088**)	6 (**.099**)	15 (**.169**)	15 (**.190**)	14 (**.206**)	12 (.197)
圧縮 (2.5KB)	93	85	73	71	6 (.065)	6 (**.071**)	6 (.082)	6 (.085)	15 (**.161**)	14 (.165)	14 (.192)	14 (.197)

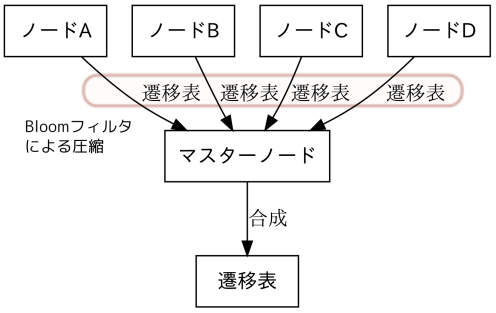

図 2: 分散遷移表合成法とその圧縮

案する (図 2参照). Bloom フィルタ [2] は複数のハッシュ表を使うことで 0 または 1 の二値を，偽陽性を許して記録するデータ構造である．遷移回数を記録するために Counting Filter[1] と呼ばれる，立っている bit 個数に応じて非負の値が記録できる実装を使うことにした．利用する bit 数を削減するために回数そのものではなく丸めた対数を記録する．分散遷移表を用いることで既にある程度の誤差が予想されるが，Bloom フィルタを用いることで (1) 存在しない遷移が検出されること，(2) 遷移回数が丸められたものになることにより，さらに誤差が増加すると考えらるため，この手法の妥当性も検証が必要となる．

処理の流れは以下のようになる．

1. 各サーバがログ内で同一セッションでの遷移を抽出する
2. 得られた遷移を (遷移元 URL，遷移先 URL) をキー，その出現回数を値とする表にする．
3. (遷移元 URL, 遷移先 URL) をハッシュすることで表を Bloom フィルタによって圧縮された表にする
4. マスターノードに転送する
5. マスターノードは OR 結合により Bloom フィルタを合成し，逆変換によって表を復元する
6. 遷移確率が高い遷移列の集合を出力する

4 実験評価

提案する 2 手法の精度を検証するために，実際の運用中に得られた以下のデータを利用してシミュレーションを行なった．まず，92 ページのコンテンツに対し数日間のアクセスによって 1952 回のページ訪問，303 回のセッションが得られた．利用したサーバは 5 台であり，サーバ負荷の増加によって複製され 2 箇所に存在するページは 9 ページ，以下同様に 3 箇所は 5 ページ，4 箇所は 1 ページ，5 箇所 (全サーバ) に複製されたページはなかった．

実験ではこのデータに対し，前述の手法を模倣したプログラムを適用する．再現率に対するハッシュ値の衝突率の影響を調べるため，Bloom フィルタを構成するパラメータ (特にハッシュ空間の大きさ) を変えたものを 4 種類準備した．

結果を表 1 に示す．列の第 1 群は閾値以上の遷移で検出された個数，第 2 群はその中で閾値 0.5 の大域ログ手法で検出された遷移の検出個数 (括弧ないは比率)，第 3 群は閾値 0.25 に対する検出個数と比率である．なお，大域および分散手法でのデータ量はテキストとして記録されたものを gzip によって圧縮した後の数値である．

傾向として，ログの分散化を行なうことで検出率は 2 割程度にまで大きく下がり，分散手法と圧縮手法の間には大きな差はなく，むしろ圧縮することで改善されている場合も存在することがわかった．これは Bloom フィルタによって偽判定された遷移の出現が，分散遷移表で失なわれた，サーバをまたがる遷移群をランダムに再現しているからだと思われる．

5 おわりに

Bloom フィルタを用いることでサーバ間での転送報量を 2 〜 3 桁と大きく減しながら，分散 Web システム上でのアクセス履歴の構築が可能であることを示した．ただし，精度は 2 割程度に留まり，転送量を多少増やしても改善が必要な値と思われる．

再現率を改善する簡単な方法は，ページアクセス時にクライアントが参照元ページの情報を付加してサーバに送り，その情報をログに残せばよい．今回の結果を踏まえると，若干数のサーバだけが付加すれば十分と考えられる．あるいは，一部の定数個のサーバは大域遷移表合成法を用い，残りのサーバは圧縮分散遷移表法を用いる混合戦略を用いることで改善をすることができるものと思われる．さらに，遷移先 (または遷移元) がないアクセスに対するヒューリスティックを与えることでサーバ間の遷移の推測を行なう方法も考えられる．

これらの評価は今後の課題である．また，より大規模な分散システムでのデータを用いた評価についても検討したい．

参考文献

[1] J. Almeida and A.Z. Broder. Summary cache: a scalable wide-area Web cache sharing protocol. IEEE/ACM Transactions on Networking, 8(3):281–293, June 2000.

[2] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. Communications of the ACM, 13(7):422–426, July 1970.

[3] Ion Stoica, Robert Morris, David Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, pages 149–160. ACM, 2001.

[4] 櫓崎修二 and 田代純規. Web ブラウザ上で動的サーバ選択を行なう JavaScript エージェント. In 合同エージェントワークショップ & シンポジウム 2007(JAWS2007), 2007.