2E2-2

分散 Web システム上での遷移履歴の分散構築

A Distributed Construction of Common Transition Histories on a Distributed Web System

楢崎修二 *
Shuji Narazaki

* 長崎大学大学院工学研究科
Graduate School of Engineering, Nagasaki University

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。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。しかし両者を組み合せるためには複数のサーバに分散する利用者ごとのページ遷移履歴を集約する必要がある。本研究では Bloom フィルタを用いて履歴集約時の転送データ量を減らす手法についてシミュレーションにより検討を行う。

はじめに

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集中が想定される Web システムのユーザビリティを改善するために P2P のアイデアを導入した Web システム提案し構築している [楢崎 07].

ユーザビリティの改善のための手法は以下の二つである．まず，サーバを分散化させ，ページの複製を作ることでアクセスを分散化し，利用者のアクセス時のレイテンシを削減させている．そして，アクセス情報の解析による共通性の高いページ間遷移列を抽出し，この情報を JavaScript で利用することでヒット率の高いページの先読み機構を実現している．

しかし，このようなサーバが分散化された環境ではセッションごとの履歴を構築し分析するために各 Web サーバの利用者ごとのページ訪問系列 (すなわちセッション) を記録したログ情報を転送し集約する必要がある．利用者やサーバ数が増えるにつれ，転送量は線形に増加することが想像される．一方，共通性が高い遷移情報を得るために各サーバが持つログ情報を完全に転送する必要は必ずしもない．そこで非可逆圧縮によって転送量を減らす手法について検討した [楢崎 11]．基本となるアイデアは Bloom フィルタの利用である．しかし，我々の従来手法では圧縮率は高いが十分な再現率 (妥当な遷移の抽出確率) を得られなかった．そこで，本論文では転送すべき情報を変更するで再現率を改善する新たな手法について提案する．

連絡先: 楢崎修二，長崎大学大学院工学研究科電気情報工学専攻情報システム工学コース，narazaki@cs.cis.nagasaki-u.ac.jp.

1. 対象システム

本論文で対象とする前述の Web システムは，分散ハッシュ表を用いて Chord[SMK+01] と同様のリング状の順序関係を構成する複数の Web サーバ群からなり，サーバサイドで P2P システムを構成している (図 1参照)．Web システムが提供する各コンテンツ (Ajax によって流通する Web ページであるため，以下では単にページと呼ぶ) はその URL をキーとして得られた分散ハッシュ値に基づき，置かれるサーバが静的に決定される．

そして，個々のサーバはその負荷に応じてコンテンツの複製をリング上で近隣のサーバ群に置くようにふるまう．これによりハッシュに基づくページの静的分散とサーバ負荷に基づくページの動的複製とを実現している．クライアントである Web ブラウザはユーザがリンクをクリックするたびにターゲットページの URL をハッシュし，また，対象サーバの負荷情報を考慮することで，特定のゲートサーバなどを経由することなく，適切と思われるサーバに直接リクエストを出す．

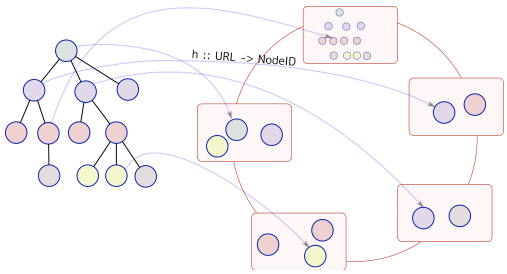

Figure 1: 分散 Web システム
ページによって木構造を構成しているコンテンツはそれぞれハッシュ関数によって複数あるノード (図では 5 つ) のいずれかにマッピングされる．ノードの 1 つは全てのコンテンツを持つプライマリサーバとして各種の管理を行なう．

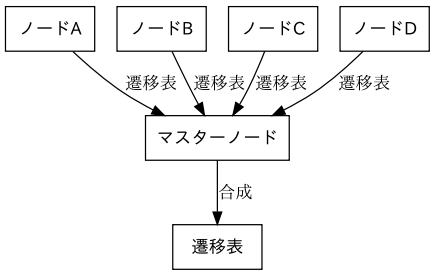

Figure 2: TT$_d$ 手法

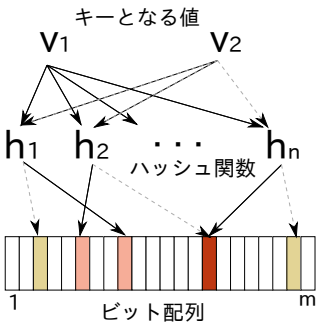

Figure 3: Bloom フィルタの構成

2. 遷移履歴の構築手法と問題点

このように対象 Web システムではページは複数のサーバに分散しているため，特定のユーザの完全なページ訪問履歴を取得するには，全サーバのログ情報を一箇所に転送し，アクセス時刻順にログを合成する必要がある．(この集約方法を LOG$_g$ 手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 が符号化されて残っているため，合成とは，同じセッション ID を持つアクセスを時刻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る処理となる．合成後に統計処理をすることで，頻出するページ間遷移系列列を抽出することができ，この情報をクライアントに送信することで，先読みが実現できる．

TT$_d$ 手法　当然，この合成処理にはテキストであるログ情報を全て転送することが必要となり，通信路に負荷を掛けることになるため，データ構造の改良により転送量を減らす必要がある．そこで我々は (遷移元の URL，遷移先の URL) の組をキーとし，その遷移の発生回数を値とする 2 次元配列すなわち遷移表 (Transision Table) を構成して転送する手法 (以下では TT$_d$ 手法と呼ぶことにする) を提案した (図 2参照).

遷移表からは個別利用者の遷移情報は復元できないが，転送されるデータ量は訪問回数にはよらない定数となるため，利用者が多い状況では転送量の削減が期待できる．集約ノードでは送信された配列の和を取ることでシステム全体の遷移が復元できる．

TT$_{BF}$ 手法　次に，この遷移表は疎であることが期待できるため，Bloom フィルタを用いて遷移表を非可逆圧縮することで，さらに転送量を減らす手法を提案した．

Bloom フィルタ [Blo70] 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．Bloom フィルタ自身はビット配列 (長さを m で表わすことにする) と複数のハッシュ関数 (その個数を n で表わすことにする) からなる (図3参照)．キーは n 個のハッシュ関数によって得られた $I_n \in [1, m]$ の添字に対応付けられ，そのビットを 1 にすることで登録される．検索は対象となるキーに対応する n 個のビットが全て 1 であれば登録済みとみなされ，そうでなければ未登録とみなされる[1]．複数のハッシュ関数を使うことでハッシュ値の衝突による偽検出の確率を減らしている．

TT$_{BF}$ 手法では各ノードの遷移表がまず Bloom フィルタを用いた表現に変換され，転送される．集約ノードでは Bloom フィルタの本体である bit 列の or をとることで大域的な遷移表が復元される．非可逆圧縮をするため，分散遷移表を用いた場合以上に誤差が増加すると思われたが，実験で評価したところ，問題となるような再現率の悪化は分散遷移表を用いた時点ですでに発生し，Bloom フィルタによる圧縮段階での悪化はむしろ無視できる程度のものであった．

具体的なものとして．[楢崎 11] での実験結果をまとめたものを表 1に示す．表が示すように，Bloom フィルタを使うことで大きく転送量を減らすことができたが，再現率は 1 割以下となっている．ここで再現率とは LOG$_g$ 手法において検出された，確率 0.5 以上で遷移する (遷移元 URL, 遷移先 URL) の組[2]がそれぞれの手法で確率 0.25 以上で遷移すると判断された組の中に含まれる割合を意味する．先読みの精度を落した設定 (たとえば遷移確率 0.25 の遷移の検出) にすると再現率は 2 割程度にまでは増加するが，

[1]実際に使用したのは遷移回数を登録できるように，1 が立っているビットの数で正整数を表わすように拡張された Counting Filter[AB00] である．キーを (遷移元ページ, 遷移先ページ) の組，値を遷移回数とするものになる．

[2]従って 2 回に 1 回は先読みが成功し，リンクのクリック時に Web サーバへの通信をするなくそのページを表示できることになる．

Table 1: 既存手法のまとめ
(LOG$_g$ 手法では gzip 圧縮したものを転送)

手法	平均転送量	検出数	再現率
LOG$_g$ 手法	93KB	36	-
TT$_d$ 手法	440B	85	0.070
TT$_{BF}$ 手法	128B	79	0.076

Table 2: SL_{BF} 手法での検出率における配列長 m とハッシュ関数数 n の影響

n\m	13bit(1KB)	14bit(2KB)	15bit(4KB)	16bit(8KB)	17bit(16KB)	18bit(32KB)
1	0/13 (0.00)	0/24 (0.00)	0/36 (0.00)	0/8 (0.00)	4/23 (0.17)	15/44 (0.34)
2	0/21 (0.00)	0/30 (0.00)	0/28 (0.00)	16/47 (0.34)	37/69 (**0.54**)	51/94 (0.54)
3	0/26 (0.00)	0/9 (0.00)	13/42 (0.31)	58/109 (**0.53**)	64/99 (**0.65**)	64/103 (0.62)
4	0/12 (0.00)	0/3 (0.00)	16/47 (0.34)	57/105 (**0.54**)	64/104 (**0.62**)	64/103 (0.62)
5	0/16 (0.00)	0/10 (0.00)	5/35 (0.14)	43/91 (**0.47**)	61/105 (**0.58**)	64/103 (0.62)
6	0/16 (0.00)	0/10 (0.00)	5/35 (0.14)	43/91 (**0.47**)	61/105 (**0.58**)	64/103 (0.62)

それ以上の改善はどの圧縮手法でも実現できなかった．また，配列長 n を増やしノードあたりの転送量を 1024Byte などににしても有意な向上は見らねなかった．

以上より，転送量が多少増えても遷移表ではない別のデータ表現を基にした手法が必要となっていた．

3. 提案手法: SL_{BF} 手法

この問題の解決法として，本論文では，ページからページへの遷移回数ではなく，セッションごとに何回目の遷移でどのページにアクセスしたかを表わす表を Bloom フィルタで圧縮する手法を提案する (以下この手法を SL_{BF} 手法と呼ぶ)．すなわちキーは (セッション ID, セッション内遷移回数，URL) の三つ組であり，値はアクセスがあったかどうかの 2 値である．

単位時間内の訪問者数を 2^k，利用者のセッション当たりの平均訪問ページ数を 2^l，システム内に存在するページ総数が 2^m とするならば， 2^{k+l+m} のキーに対して 2 値を与えるハッシュ表が必要になる．この値は 2^{2m} のキーに対して 2^l の値を与える TT_{BF} 手法と比較して， 2^{k-m} 倍の大きさとなるが，いずれにせよ，ハッシュ表を用いることで，他の構造では実現が困難な，少ないデータ量で表現できる可能性がある．

4. 評価

実際の Web システムで得られたデータを基にしたシミュレーションによって，提案する SL_{BF} 手法と既存手法とを比較した．

設定は以下のとおりである，92 ページのコンテンツに対し 1952 回のページ訪問，303 回のセッションである (3 日間の e-learning 用コンテンツのログから得た)．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．サーバが 5 台であるため，個々のサーバでは，それぞれのセッションの遷移は 2 割程度しか追跡できない．この設定は表 1で用いたものと同じである．

結果を表 2に示す．各欄は，LOG_g 手法 で検出されたものに含まれる検出遷移数/ 検出遷移数 (再現率) として表わしている．太字は妥当な結果と考えられるものである．これより，比率が 0.5 前後を保持しながら転送量が 8KB に収まる 16bit のハッシュ空間で 3 または 4 種のハッシュ関数を用いるのがよいと思われる．この値は既存のどの圧縮手法よりも大きく再現率を改善できている．

Bloom フィルタでの衝突率は検出率と高い負の相関 (-0.93) を示したため，ページ数やセッション数に対して Bloom フィルタの大きさは調整する必要がある．ページ数やセッション数の増加に対して元となるログ情報も線形に増えるため，より規模が大きな Web システムであっても本手法は有効と考えられる．

5. まとめと今後の課題

提案手法は，ログ情報の単純な圧縮手法に比べ転送データ量を約 1 桁減らせ，5 割近い再現率を有する有効性の高い手法であることがわかった．特に，圧縮率は極めて高いが 1 割以下の再現性しか持たなかった，遷移表を圧縮する既存手法と比べて，妥当な範囲の転送量の増加によって十分に高い再現率を得ることができた．

さらに転送量を減らすためには，Bloom フィルタでの衝突率を減らすことが必要であることがわかった．そのため，現在は各サーバのログに記録された全てのセッション履歴を符号化しているが，頻出する系列や，遷移長が短かく有効性が低いセッションなどは前処理として削除してから符号化することで，衝突率を低く抑えたままでビット配列長を小さくできるのではないかと考えている．

今後の課題としては，より大規模なシステムでの評価が挙げられる．本論文では，我々が開発している Web システムを対象に提案手法の検討を行なったが，類似した分散環境での様々な時系列データの構築においても有用と考えられるため，さらなる検討を行なう予定である．

参考文献

[AB00] 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.

[Blo70] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, July 1970.

[SMK+01] 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.

[楢崎 11] 楢崎修二. 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. In **情報処理学会第 *73* 回全国大会**, ID 6B–5, 2011.

[楢崎 07] 楢崎修二 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.