

大規模 Web アーカイブのための 更新クローラの設計と実装

Design and Implementation of an Incremental Crawler for Large Scale Web Archives

田村 孝之[♥] 喜連川 優[♦]

Takayuki TAMURA Masaru KITSUREGAWA

筆者らは刻々と変化する Web 情報からの社会知の抽出を目指し、日本語 Web ページを中心とする大規模 Web アーカイブの構築を行っている。本プロジェクトは Web の大域的構造や時間変化の分析を主眼としており、Web アーカイブの網羅性と時間分解能に対する要求が高い。これらの要求に応えるため、取得した Web ページのリンクから新規 URL を発見して収集範囲を拡大しつつ、既知 Web ページに対してはその更新頻度を推定して適応的に再アクセスを行う更新クローラの設計と実装を行った。

For exploiting social knowledge from continuously changing Web information, we are building a large scale Web archive mainly containing Japanese Web pages. The project is unique in that its major concern is structural and temporal analyses of the Web. As a result, its demands for both comprehensiveness and temporal resolution are quite high. To fulfill the requirements, we have designed and implemented an incremental crawler which explores into the frontier of new URLs discovered from fetched Web pages while revisiting known URLs adaptively to their estimated change frequency.

1. はじめに

筆者らは刻々と変化する膨大な Web 情報を実社会の鏡像と見做し、その大域的な構造や時間変化の分析に基づく社会知の抽出に取り組んでいる [1]。Web 分析の基盤となるのは日本の Web 情報を網羅的に収集・蓄積した大規模な Web アーカイブである。

Web の大域的構造と時間変化の分析を主眼とする Web アーカイブに要求される網羅性と時間分解能の両立を図るには、Web ページを 1 回ずつ収集する一括クローラ (batch crawler) ではなく、連続的に動作し Web ページ毎に独立したタイミングで再アクセスを行う更新クローラ (incremental crawler) が不可欠である。更新クローラは過去のクローリング履歴をデータベース化し、Web ページ毎の更新頻度に適応したスケジューリングを行うことで、低更新頻度 Web ページのアクセスに割かれていたリソースを高更新頻度 Web ページのアクセスに振り向け、最短 1 日程度の高い時間分解能を

得ることを可能にする。

多くの国々で Web 上のデジタル文書の保全を目的とする Web アーカイビングプロジェクトが組織されているが、これらのプロジェクトにおいては網羅性よりもコンテンツの再現性 (スクリプトやプラグインの動作まで含む) が重視される傾向にある [2, 3]。また、米国の Internet Archive は古くから網羅的な収集に取り組んでいるが、1~2ヶ月毎に外部から提供されるデータが主体となっている [4]。Internet Archive 独自のオープンソースクローラの開発も進められているが、更新クローリングを実現するには至っていない [5, 6]。

また、大量の Web ページを扱うことの困難さを回避するために、Web サイトや Web ページ群を単位として更新の有無を推定する手法も提案されている [7]。これらの手法では 1ヶ月あるいは 1週間などの比較的長い間隔で一部の Web ページをサンプリングし、更新が検出された場合に残りの Web ページを一括してクローリングするという戦略を用いる。しかし、更新されていない Web ページの収集コスト (false positive) や更新された Web ページの収集漏れ (false negative) の問題が本質的に伴うため、大規模な Web ページ集合を高い時間分解能で収集するのは難しい。

一方、Web ページの更新頻度に適応してアクセススケジューリングを行っても、与えられたリソースの下ではその実施が不可能な場合もある。従来、クローラの動作を制約するリソースとしては、クローラ側のマシンやネットワークが想定されてきたが、相手 Web サーバに及ぼすアクセス負荷を考慮することも長期的にクローリングを継続する上では重要である。本稿で述べる更新クローラは、Web ページの更新頻度に基づく再アクセススケジューリングを行うと共に、その結果を Web サーバ毎に集約することで Web サーバの負荷状態を把握し、Web サーバに対するアクセス間隔の制御を実現する。

2. アクセススケジューリングアルゴリズム

2.1 Poisson 過程に基づく Web ページ更新間隔推定

更新クローラにおいては、Web ページ毎に独立に再アクセスのタイミングを決定することにより、一括クローリングでは実現できない大規模かつ高頻度の収集を可能にする。あるページを再度アクセスするタイミングはそのページが次に更新された直後であることが望ましい。従って効率的な再アクセススケジューリングの実現には、Web ページの更新タイミングを正確に見積もることが重要となる。

Choら [8] は Web ページの更新が一定の更新頻度 λ (更新間隔の逆数) を持つ Poisson 過程に従うと仮定し、ある Web ページを一定期間毎にアクセスした際の更新検出結果から λ を推定する方法について包括的に議論している。しかし、大規模 Web アーカイブ構築においては大量の Web ページを扱うため、一定期間のアクセスを高頻度に繰り返して更新間隔を推定するのは現実的でなく、推定した更新間隔を直ちにアクセス間隔に反映させて不必要なアクセスを生じさせないようにしなければならない。

ここで Choら [8] と同様に不規則アクセスによる更新検出を表す変数を導入する。すなわち、あるページに対して n 回アクセスした際に m 回 ($m < n$) の更新を検出したものとし、第 j 番目の更新を検出した際のアクセス間隔を t_{ci} 、更新が検出されなかったアクセス間隔の第 j 番目のものを t_{uj} とする (図 1 参照)。

[♥] 正会員 三菱電機 (株) 情報技術総合研究所

ttamura@acm.org

[♦] 正会員 東京大学生産技術研究所

kitsure@tkl.iis.u-tokyo.ac.jp

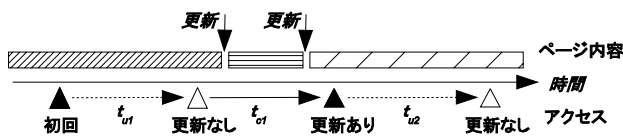


図1 不規則アクセス間隔によるページ更新検出
Fig.1 Page change detection with irregular access intervals

Poisson過程では t_{ci} の期間に1回以上の更新が起こる確率は $1 - \exp(-\lambda t_{ci})$, t_{ij} の期間に更新が起こらない確率は $\exp(-\lambda t_{ij})$ であるから、実際にある更新パターンが観測される確率はそれぞれの確率の積となり、最尤法により λ が満たすべき条件は以下の式(1)で与えられる。

$$\sum_{i=1}^m \frac{t_{ci}}{\exp(\lambda t_{ci}) - 1} = \sum_{j=1}^{n-m} t_{uj} \quad (1)$$

Cho ら [8] は不規則アクセスに基づく更新間隔推定方法として式(1)の直接解法を挙げるに留まっているが、この式は過去のアクセス間隔の履歴を全て含んでおり、Web ページ毎に保持すべき状態量が膨大になってしまう。また、保持する履歴の範囲を有限ウィンドウ内に限定したとしても、式(1)から λ の値を求める計算は単純ではない。そこで式(1)を近似し、より単純な推定方法を導いた。

まず、各 t_{ci} が全て等しい場合、すなわち $t_{ci} = t_c (1 \leq i \leq m)$ の場合は式(1)を解くことができ、 λ の推定値の逆数は式(2)で与えられる。

$$\hat{\lambda}^{-1} = t_c / \log \frac{mt_c + U}{U} = t_c / \log \frac{T}{U} \quad (0 < U < T) \quad (2)$$

ただし、 U は更新が検出されなかったアクセス間隔の総和、すなわち $\sum t_{uj}$ を表し、 $mt_c + U$ は全アクセス間隔の総和に等しいので T とした。

全ての t_{ci} が等しくない場合については、式(2)の t_c を $\{t_{ci}\}$ の代表値 \bar{t}_c で置き換えた式(3)により λ の逆数を近似する。

$$\hat{\lambda}^{-1} \approx \bar{t}_c / \log \frac{T}{U} \quad (0 < U < T) \quad (3)$$

\bar{t}_c としては例えば t_{ci} の最小値 $t_{c \min}$ や平均値 $t_{c \text{avg}}$ などを用いることができる。ただし、最小値、平均値とも極端な t_{ci} 値の影響を受け易いため、最小値と平均値の幾何平均を用いている。指数移動平均により有限の履歴を用いるようにしても良いだろう。また、比 U/T は Web ページの安定度を表すと考えられ、この値が大きいほど $\hat{\lambda}^{-1}$ を大きくする効果がある。

式(3)においては個々のアクセス間隔の履歴は不要であり、初回アクセスからの経過時間 T 、更新が検出されなかったアクセス間隔を累積した U 、更新を検出したアクセス間隔の最小値 $t_{c \min}$ や平均値 $t_{c \text{avg}} (= (T - U) / m)$ から次回アクセスのタイミングを決定することが可能になる。このように Web ページ毎に保持する状態変数を大幅に削減したことにより、大規模な Web ページ集合に対して再アクセススケジューリングを適用することが可能となった。

なお、式(3)には特異点があり、 $\hat{\lambda}^{-1}$ を求められない場合がある。すなわち、全く更新が検出されない場合 ($U = T$) や全てのアクセスで更新が検出される場合 ($U = 0$) である。これらの場合には直前のアクセス間隔を定数倍する exponential

back off により次のアクセス間隔を決定する。また、 $T = 0$ となる初回アクセス後には、2 回目のアクセスまでの間隔のある範囲の一樣乱数を生成して決定する。

2.2 Web サーバアクセス間隔の制御

Web ページ再アクセススケジューリングにより、各ページをアクセスするタイミングが決定されるが、実際のクローリングにおいては Web ページは独立な実体ではないことに注意が必要である。すなわち、Web ページは URL をキーとする Web サーバへの問合せの結果として得られるものであり、Web サーバに対するアクセスタイミングも考慮に入れなければならない。

一般にクローリングにおいては同一 Web サーバから同時に複数の Web ページを取得することはマナー違反とされる。また、Web サーバへのアクセスを逐次に行ったとしても、連続するアクセスの間隔が短いと Web サーバが高負荷となるので好ましくない。どの程度のアクセス間隔なら許容されるかは明確には規定されておらず、Web サーバ管理者の心証に依る所も大きい。少数 Web サイトを対象とする選択的アーカイビングでは事前に Web サーバ管理者に了解を得ることもできるが [3]、大規模 Web アーカイブ構築においては実施不可能であり、明示的な拒否 (管理者からの連絡や robots.txt などの Robot Exclusion Standard に従った設定) がない限り収集を行う opt-out ポリシーを採用せざるを得ない。従って、Web サーバへの負荷を最小限に止める配慮は極めて重要である。

ここで、Web サーバ i に属する全ての Web ページの集合を $\{P_{ij}\}$ とし、 P_{ij} の再アクセス間隔を T_{ij} 、十分長い期間 τ における P_{ij} のアクセス回数を N_{ij} とする。Web サーバ i へのアクセスを時間 I_i 毎に逐次的に行うという制約の下で各ページの取得が可能となるには、

$$I_i \sum_j N_{ij} \leq \tau \quad (4)$$

が成り立つ必要がある。 $N_{ij} \approx \tau / T_{ij}$ (小数部分を無視) により、式(4)から以下が得られる。

$$I_i \leq \frac{\tau}{\sum_j N_{ij}} = \left(\sum_j \frac{1}{T_{ij}} \right)^{-1} \equiv L_i^{-1} \quad (5)$$

L_i は Web サーバ i に属するページのアクセス頻度の総和となっており、Web サーバの負荷を表す指標と考えられる。負荷指標の逆数である右辺全体は、Web ページ再アクセススケジューリングに従ったアクセスを実現する Web サーバアクセス間隔の理論的な上限である。Web サーバの負荷指標が大きくなるに従って、より短い間隔で Web サーバにアクセスする必要が生じるが、 L_i^{-1} がマナー上許容できない値となった場合、Web サーバは過負荷状態となり、Web ページ再アクセススケジューリングに従ったアクセスが実施不可能となる。

このように、Web ページ再アクセススケジューリングの結果を Web サーバ毎にまとめて負荷指標という単純な値で表すことにより、必要十分な Web サーバアクセス間隔を設定したり、過負荷状態の検出に利用することが可能になる。

なお、Web サーバ負荷指標は要素である Web ページの追加・削除 (収集対象からの除外) や再アクセス間隔の変更に対し、インクリメンタルな更新が可能である。すなわち、現

在の負荷指標の値と更新前後のWebページの再アクセス間隔の値だけに基づいて（全Webページに関する値をスキャンすることなく）新たな負荷指標を設定することができる。

3. 更新クローラの実装

前節で述べたWebページおよびWebサーバに対するアクセススケジューリングを実現する更新クローラをPCクラスタ上で実装した。図2は単一ノード上のプロセス構成およびデータフローである。以下、各プロセスの機能を述べる。

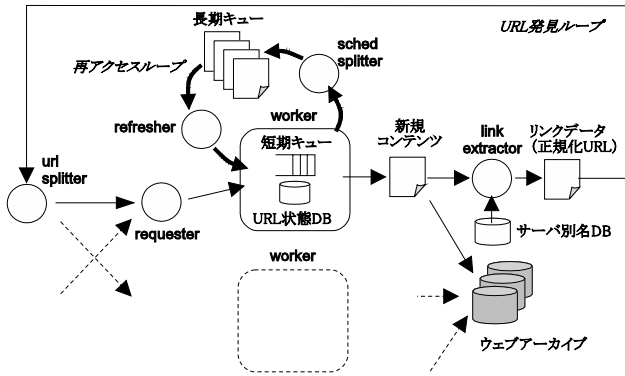


図2 更新クローラの構成
Fig.2 Architecture of incremental crawler

worker: クローラの本体であり、Webサーバと通信してWebページのダウンロードを行う。また、requesterプロセスおよびrefresherプロセスからダウンロード対象のURLを受け取って処理する。requesterが要求するのは新規URL候補であり、URL状態DBに既に存在する場合は棄却する。URL状態DBに存在しなかった場合は新たにエントリを追加し、URLを短期キューに投入する。refresherからは常に既知URLを受け取り、短期キューへの投入を行う。

短期キューはWebサーバ毎に独立したプライオリティキューと見做すことができ、Webサーバアクセススケジューリングに従ったURLの取り出しを可能にする。短期キューには1日以内に取得可能なURLのみを格納し、それを超えるURLは長期キューに書き出して3日後に再度受け付けるようにしている。

workerはWebページをダウンロードしてそのコンテンツをファイルに格納すると、Webサーバの応答コード(304: Not Modified)やコンテンツ(ハッシュ値)の変化に基づいて更新判定を行う。その結果をWebページ再アクセススケジューリングに適用し、再アクセス時期の決定を行うと共に、URL状態DBに含まれる状態変数(2.節の T, U , および \bar{t}_c など)やコンテンツハッシュ値を更新する。さらに、URLを短期キューから取り除いて長期キューに移動する(sched splitterに渡す)。

sched splitter: workerが出力する再アクセスURLをアクセス日付順にソートし、長期キューを構成する1日毎のファイルのうち、対応する日付のファイル末尾に追加する。

refresher: 一定時間毎に動作し、長期キューから再アクセス期限を過ぎたファイルを読み込んで、workerにURL再アクセス要求を発行する。

link extractor: workerが出力した新規コンテンツの内、リンクを含むHTMLデータを解析してリンク先URLを抽出し、ルールに従って選別や優先度設定を行った上でリンクデータファイルに出力する。リンク先URLは相対パス指定や表記

の曖昧さ(エスケープシーケンスなどによる)の解決と共に、サーバ別名DBを参照したサーバ名の正規化を施される。

url splitter: リンクデータファイルをURLのサーバ名部分にハッシュ関数を適用することにより、担当するクローラノード毎のファイルに分割する。分割後のファイルを各担当ノードに送付する。なお、クローラノード間の通信はこの部分でのみ発生し、残りのほとんどの処理は各ノードが独立して動作する。

requester: 各ノードのurl splitterから転送されたリンクデータファイルの断片を読み込み、workerに対して新規URLアクセス要求を送る。

link extractor, リンクデータファイル, url splitter, およびrequesterを含むループは新規URL発見のための機構であり、一括クローラにも見られるものである。一方, sched splitter, 長期キュー, およびrefresherからなるループはURL再アクセスに固有の機構である。更新クローラでは、これら2つのループが同時に動作することにより既知ページの更新と新規ページの出現の双方に対応することが可能となっている。

4. 大規模クローリング実験

ここでは更新クローラの動作結果から得られた知見について述べる。クローリング開始に当たってはこれまで数ヶ月~1年置きに実施してきた一括クローリングの数年分の履歴を活用し、URL状態DBの初期値に反映した。その後、改良を加えながら延べ約1年に亘り更新クローラを動作させた。

図3は、実験終了時点でのWebページ再アクセス間隔の頻度分布である。ただし、1回しかアクセスしていないWebページは除いている。一週間以内に再アクセスするページが非常に多くなっている。一方、長期間更新がないと予測されるページも多い。再アクセス間隔の上限を400日とし、これを超えるページについては300~400日の間の一様乱数で代替したため、この区間のページ数が目立って多くなっている。これらのページは再度更新される見込みは極めて少ないと思われるが、Webサーバやドメインの廃止によりアクセスできなくなる可能性はあるため、強制的にアクセスするようにしている。

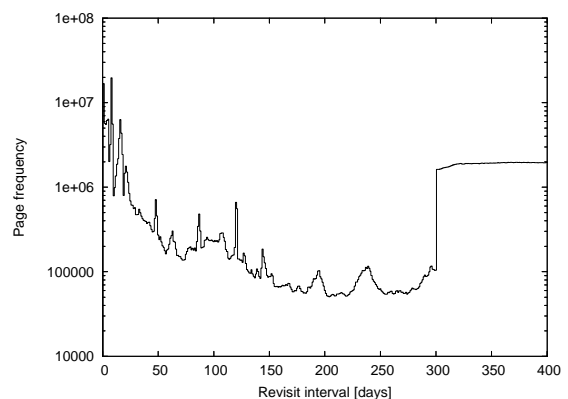


図3 Webページ再アクセス間隔の頻度分布

Fig.3 Histogram of web page revisit intervals

図4は同一再アクセス間隔を持つWebページについてページ更新検出率の平均および標準偏差をプロットしたものである。ページ更新検出率はページアクセス回数に対する一

意バージョン取得数の比であり、1に近い程無駄なアクセスがないことを示すが、逆にページ更新を取りこぼすリスクも増す。グラフでは再アクセス間隔の増加と共にページ更新検出率が徐々に減少している。再アクセス間隔が大きい領域ではさらにアクセス間隔を増加させてアクセス回数を減らす必要がある。また、再アクセス間隔が小さい領域ではページ更新検出率が1に近付いている。これらについては再アクセス間隔をさらに短縮し、バージョンの取りこぼしがないことを確認する必要がある。

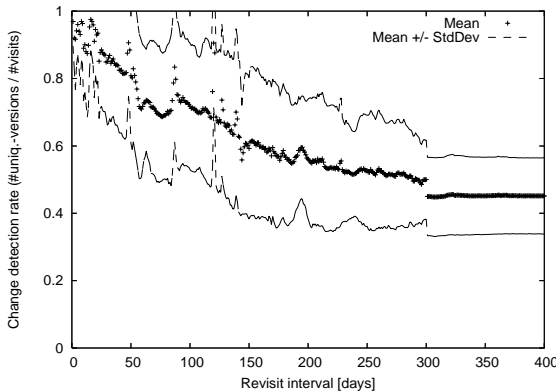


図4 Web ページ再アクセス間隔に対する更新検出率

Fig.4 Change detection ratio vs. Web page revisit intervals

図5はWebサーバ負荷指標（Webサーバに属するページの再アクセス間隔の逆数の総和）の逆数として与えられるWebサーバアクセス間隔理論値の分布を示したものである。

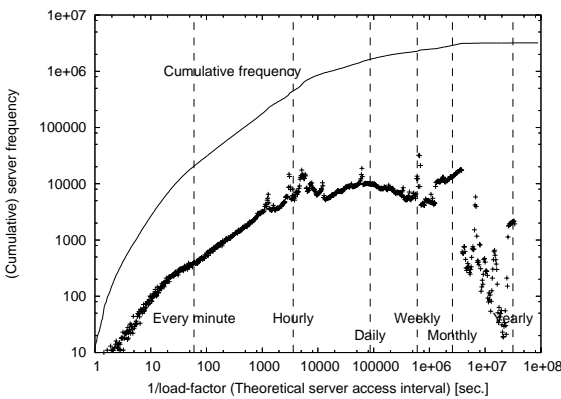


図5 Webサーバアクセス間隔理論値の分布

Fig.5 Distribution of theoretical web server access intervals

図5によると非常に緩やかなアクセスで十分なサーバが多いことが分かる。1分以内にアクセスする必要があるのは全体の1%に満たない。しかしながら、数秒毎にアクセスする必要があるサーバも100程度存在している。これらのサーバに対しては、理論値通りのアクセスを行うとマナー違反と見做されてしまう可能性が高い。しかしながら、単純にアクセス間隔の下限値を適用すると、収集対象Webページや再アクセス間隔が制御不能になってしまう。そのため、これらの過負荷サーバに対しては、Webページ再アクセス間隔の修正や一部URLの除外などの対処が必要になる。Webサーバ負荷指標を維持することで、過負荷状態を検知するだけでな

く、過負荷状態の解消を定量的に実施することも可能になる。

5. まとめ

本稿では、Web分析の基盤となる大規模Webアーカイブの構築を目的とした更新クローラについて述べた。更新クローラは収集したWebページから新たなリンク先URLを発見して収集対象に加えると共に、過去の更新傾向に基づいて各Webページの再アクセスをスケジューリングすることで高い網羅性と時間分解能の両立を図る。また、Webサーバに及ぼす負荷を考慮し、アクセスマナーの制約下でWebページ再アクセススケジューリングが実施可能かどうかを判定する指標を導入した。

実際に数百万Webサーバを対象とする大規模クローリングを行い、そこから得られたWebサーバの特性を評価した結果、一部のWebサーバが過負荷状態となっており、理想的なスケジューリングが実施できない条件下にあることが分かった。これらのWebサーバに対しては、Webページ再アクセススケジューリングを修正し、次善の状態を求めるようにして行くことが不可欠である。今後は各種スケジューリングアルゴリズムの詳細な評価と共に、過負荷状態の解消手法について検討し、更新クローラへの統合を進めていきたい。

【謝辞】

本研究の一部は文部科学省リーディングプロジェクト e-Society 基盤ソフトウェアの総合開発「先進的なストレージ技術およびWeb解析技術」による。

【文献】

- [1] 豊田, 喜連川: “日本におけるウェブコミュニティの発展過程”, 日本データベース学会 Letters, 2, 1, pp. 35-38 (2003).
- [2] International Internet Preservation Consortium: “netp-reserve.org”. <http://netpreserve.org/>.
- [3] 国立国会図書館: “インターネット資源選択の蓄積実験事業(WARP)”. <http://warp.ndl.go.jp/>.
- [4] Internet Archive: “About IA”. <http://www.archive.org/about/about.php>.
- [5] K. Sigurosson: “Incremental crawling with heritrix”, Proc. of IAWW (2005).
- [6] K. Sigurosson: “Managing duplicates across sequential crawls”, Proc. of IAWW (2006).
- [7] J. Cho and A. Ntoulas: “Effective change detection using sampling.”, Proc. of VLDB, pp. 514-525 (2002).
- [8] J. Cho and H. Garcia-Molina: “Estimating frequency of Change”, ACM TOIT, 3, 3, pp. 256-290 (2003).

田村 孝之 Takayuki TAMURA

三菱電機株式会社情報技術総合研究所専任および東京大学生産技術研究所協力研究員。1996 東京大学大学院工学系研究科博士課程修了, 博士(工学)。並列データベース, Webマイニングに関する研究に従事。情報処理学会, 電子情報通信学会各会員。日本データベース学会正会員。

喜連川 優 Masaru KITSUREGAWA

東京大学生産技術研究所戦略情報融合国際研究センター長, 教授。1983 東京大学大学院工学系研究科博士課程了。工博。データベース工学, 並列処理, Webマイニングに関する研究に従事。情報処理学会フェロー, 電子情報通信学会フェロー, SNIA-Japan 顧問。日本データベース学会理事。