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

田村　孝之[†,††]　　喜連川　優[††]

† 三菱電機株式会社 情報技術総合研究所
〒 247–8501 鎌倉市大船 5–1–1
†† 東京大学生産技術研究所
〒 153–8505 東京都目黒区駒場 4–6–1
E-mail: ttamura@acm.org, kitsure@tkl.iis.u-tokyo.ac.jp

あらまし　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．Web の大域的構造や時間変化の分析を主眼とする Web アーカイビングプロジェクトは世界的にも稀であり，要求される高い網羅性と時間分解能を達成する手法については十分明らかにされていない．本稿では，大規模 Web アーカイブ構築を目的とする更新クローラの設計と実装について述べる．本クローラは取得した 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．Web ページ再アクセススケジューリングにおいては大規模化を可能にするため状態変数を削減し，物理的な制約を考慮したスケジューリング実施可能性判定のための指標を導入する．これらを具現化する更新クローラを PC クラスタ上に実装すると共に，実際の動作を通じて現実の Web サーバに関する性能指標を収集し，予備評価を行った．

キーワード　Web アーカイブ，更新クローリング，適応的スケジューリング，負荷制御，性能評価

Design and implementation of an incremental crawler for large scale web archives

Takayuki TAMURA[†,††] and Masaru KITSUREGAWA[††]

† Information Technology R&D Center, Mitsubishi Electric Corporation
5–1–1 Ofuna, Kamakura-shi, 247–8501 Japan
†† Institute of Industrial Science, The University of Tokyo
4–6–1 Komaba, Meguro-ku, Tokyo, 153-8505 Japan
E-mail: ttamura@acm.org, kitsure@tkl.iis.u-tokyo.ac.jp

Abstract　For exploiting social knowledge from continuously changing Web information, we are building a large scale Web archive mainly containing Japanese Web pages. The Web archiving project is quite unique in that its major concern is structural and temporal analyses of the Web. Methods to achieve both comprehensiveness and high time-resolution required for such analyses have not been well studied. This paper describes design and implementation of an incremental crawler for building large scale Web archives. The crawler explores into the frontier which is comprised of newly discovered URLs from fetched Web page, and simultaneously, it is capable of revisiting known URLs adaptively to their estimated change frequency. Our Web page revisit scheduling algorithm reduces necessary state variables per page, leading to good scalability. In addition, new metrics are introduced to help determination of feasibility of the schedule under a physical constraint. The incremental crawler which realizes these methods was implemented on a PC cluster. The crawler was run to gather performance figures of real world Web servers, which were to be evaluated preliminarily.

Key words　Web archiving, Incremental crawling, Adaptive scheduling, Load control, Performance evaluation

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], [2]．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．本稿では，大規模 Web アーカイブ構築のための更新クローラの設計と実装について述べる．

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．一括クローラによる大規模クローリングでは億ページ単位のデータをまとめて扱う必要があり，収集前後の処理まで考慮すると 1ヶ月未満の時間分解能を得ることは難しい．また，Web ページは更新されるだけでなく生成・消滅が頻繁であり，更新の頻度もページ毎に大きく異なる．収集対象の URL セットや Web ページ毎の更新有無に関するデータの管理は次第に煩雑となり，クローリング自体よりこうした派生データのメンテナンスがボトルネックとなってしまう．

更新クローラでは過去のクローリングに関する派生データをデータベースとして集約し，オフライン処理への依存を排することで連続的な動作を実現する．これにより 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い，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ることができ，最短 1 日程度の高い時間分解能を得ることが可能になる．

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，更新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などが想定されてきた．しかし，長期的にクローリングを継続する上では，Web サーバに及ぼすアクセス負荷を考慮してスケジューリングを行うことも重要である．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 サーバへのアクセス間隔を制御する．さらに，Web サーバの応答性能を考慮してネットワークリソースを割り当て，スケジュールの実施が不可能な Web サーバを検出してスケジュールの修正を施せるようにしている．

以下，2. で関連研究について述べる．3. では Web ページアクセスの際の更新検出に基づく再アクセススケジューリングアルゴリズムについて説明し，4. で Web サーバの負荷やクローラリソース制約を考慮した Web サーバ毎のスケジューリング方法について述べる．5. ではこれらのスケジューリング手法を実現し，PC クラスタ上で動作する更新クローラの実装について説明する．6. では更新クローラを用いたクローリング実験から明らかになったことと課題について述べ，7. で全体のまとめを行う．

2. 関連研究

日本を含め各国で国立図書館を中心に Web アーカイビングプロジェクトが組織されている [3]~ [5]．これらのプロジェクトは文化的な背景から Web 上のデジタル文書を保全することを目的としており，規模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．米国の Internet Archive [6] 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．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，大規模化への取り組みは始まったばかりである [7]．また，更新クローリングのサポートが検討されたものの，断念されたようである [8], [9]．

Web ページ更新傾向の観測に基づく研究としては文献 [10], [11] があり，Web ページ更新は単純な Poisson 過程としてある程度モデル化できることが確認されている．Cho ら [12] は一定間隔で Web ページをアクセスした際に，検出された更新の有無から Poisson 過程のパラメータとして Web ページの更新頻度を推定する手法について述べている．本稿でも同様のモデルに基づいて更新頻度の推定を行うが，大規模クローリングへの適用においては更新頻度推定を目的としたアクセスを行うことは現実的ではなく，実際の Web ページ収集に伴う不規則なアクセス間隔を前提としている．また，各 Web ページの更新有無の履歴は全体として膨大なデータ量になるため，Web ページ毎の状態変数を数個に抑えるような近似を行った．

WebFountain クローラ [13] は特定の Web ページ更新モデルに依存せず，一定期間毎に非線形連立方程式を解いて収集対象 Web ページを決定する．クローラリソースの制約は考慮されているものの，リソース不足がクロール範囲や収集頻度に及ぼす長期的な影響を定量的に把握することはできず，リソース制約に適応してアクセス戦略を修正するのは困難である．また，Cho ら [11], [14] はクローラリソースが不足する場合に Web ページの更新頻度に基づいてアクセスの優先度を決定する方法を論じており，特に，更新頻度が非常に高い Web ページは取得した内容が最新である期間も短いため，他の Web ページの収集を優先した方がスナップショット全体の新鮮度が向上するという結論を得ている．しかし，この議論は最新情報の検索が目的であることを前提としたものであり，過去の任意の時点における分析を目的とするアーカイブ用途には必ずしも当てはまらない．User-centric クローリング [15] はリソース負荷を低減するために検索ニーズに応じて Web ページに重み付けするものであるが，収集対象に偏りを生じることになるためアーカイブ用途には適さない．

一方，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 [16], [17]．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ル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

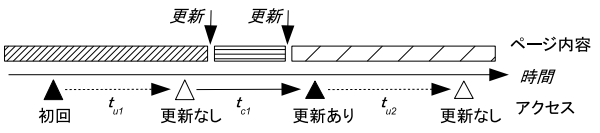

図 1　不規則アクセス間隔によるページ更新検出.
Fig. 1　Page change detection with irregular access intervals.

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．

Web サイト側が更新のあった Web ページを列挙する RSS 等のフィードを利用すると効率的に Web ページの更新を捉えることができる [18]．また，Web サイト側がクローラへのヒント情報として更新頻度や優先順位と共に Web ページ URL を列挙する sitemap プロトコルも提案されている [19]．これらの協調的な仕組みは全ての Web サイトや Web ページを網羅したものではなく，単独で大規模 Web アーカイブの構築に適用することはできないが，更新クローラの改善には有効と考えられる．

3. Web ページ再アクセススケジューリング

3.1 Poisson 過程に基づく Web ページ更新のモデル化

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．アクセスがこれよりも早いと前回と同じ内容を取得することになり，遅過ぎると別の更新が起こって未取得の内容が上書きされてしまうためである．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 [12] 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 [12] 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 i 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)．

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 $1 - \exp(-\lambda t_{ci})$，$t_{uj}$ の期間に更新が起こらない確率は $\exp(-\lambda t_{uj})$ 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式 (1) で与えられる．

$$\prod_{i=1}^{m}(1 - \exp(\lambda t_{ci})) \prod_{j=1}^{n-m} \exp(-\lambda t_{uj}) \tag{1}$$

λ の推定値は最尤法により，式 (1) を最大化する λ として得ることができる．式 (1) の対数を取ったものを λ で微分して 0 に等しいと置くことにより，λ が満たすべき条件は以下の式 (2) となる．

$$\sum_{i=1}^{m} \frac{t_{ci}}{\exp(\lambda t_{ci}) - 1} = \sum_{j=1}^{n-m} t_{uj} \tag{2}$$

Cho ら [12] は不規則アクセスに基づく更新間隔推定方法として式 (2) の直接解法を挙げるに留まっている．

3.2 不規則アクセスに対する更新間隔推定の近似

不規則アクセスに基づく更新間隔推定を表す式 (2) 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 (2) から λ の値を求める計算は単純ではない．そこで式 (2) 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 $t_{ci} = t_c$ $(1 \leqq i \leqq m)$ の場合は式 (2) を解くことができ，λ の推定値の逆数は式 (3) で与えられる．

$$\hat{\lambda}^{-1} = t_c / \log \frac{mt_c + T_{stable}}{T_{stable}} = t_c / \log \frac{T_{total}}{T_{stable}} \quad (0 < T_{stable} < T_{total}) \tag{3}$$

ただし，T_{stable} 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，$mt_c + T_{stable}$ は全アクセス間隔の総和に等しいので T_{total} とした．

全ての t_{ci} が等しくない場合については，式 (3) の t_c を $\{t_{ci}\}$ の代表値 $\bar{t_c}$ で置き換えた式 (4) により λ の逆数を近似する．

$$\hat{\lambda}^{-1} \approx \bar{t_c} / \log \frac{T_{total}}{T_{stable}} \quad (0 < T_{stable} < T_{total}) \tag{4}$$

$\bar{t_c}$ としては例えば t_{ci} の最小値 t_{cmin} や平均値 t_{cavg} 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 $\frac{T_{stable}}{T_{total}}$ 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 (4) 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_{total}，更新が検出されなかったアクセス間隔を累積した T_{stable}，更新を検出したアクセス間隔の最小値 t_{cmin} や平均値 t_{cavg} $(= (T_{total} - T_{stable})/m)$ 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 (4) 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 $(T_{stable} = T_{total})$

や全てのアクセスで更新が検出される場合 ($T_{stable} = 0$) 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，$T_{total} = 0$ 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をある範囲の一様乱数を生成して決定する．

4. Web サーバアクセススケジューリング

4.1 Web サーバ負荷の考慮

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を考慮する必要がある．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が重要となる．

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が多いが [4]，大規模 Web アーカイブ構築においては実現不可能である．筆者らは Internet Archive [6] と同様に明示的な拒否（管理者からの連絡や robots.txt などの Robot Exclusion Standard [20] に従った設定）がなければ収集を行う opt-out ポリシーを採用している．しかし，Web サイトによっては連絡なしにクローラの IP アドレスに基づく接続拒否設定が施されることもあり，Web サーバへの負荷を最小限に止める配慮は重要である．

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，

$$I_i \sum_j N_{ij} \leqq \tau \tag{5}$$

が成り立つ必要がある．$N_{ij} = \tau / T_{ij}$ （小数部分を無視）により，式 (5) から以下が得られる．

$$I_i \leqq \frac{\tau}{\sum_j N_{ij}} = \left(\sum_j \frac{1}{T_{ij}} \right)^{-1} \equiv {L_i}^{-1} \tag{6}$$

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 i 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．実際の Web サーバにおける負荷指標（Web サーバアクセス間隔理論値）の分布や過負荷状態への対処については 6. で述べる．

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．

4.2 クローラリソースの制約

大規模 Web アーカイブ構築においては同時に多数の Web サーバにアクセスするため，Web サーバの負荷とは独立にクローラのリソース制約も問題となる．ここでは，クローラのリソース制約が前節で述べた Web サーバ負荷を意識したスケジューリングにどのような影響を及ぼすか考察する．クローラが S 個の Web サーバに並行してアクセスする際に，C 個の独立な通信リソース（ソケットなど）を利用できるとする．期間 τ において Web サーバ i から K_i 個のページを取得するものとし，その第 k ページの取得に要した時間（通信リソースを占有した時間）を e_{ik}，第 k ページに対するアクセス開始から第 $k+1$ ページのアクセス開始までの時間を I_{ik} とすると，$I_{ik} \geqq e_{ik}$ であり，以下の 2 つの関係が成り立つ必要がある．

$$\sum_{i=1}^{S} \sum_{k=1}^{K_i} e_{ik} \leqq C\tau \tag{7}$$

$$\sum_{k=1}^{K_i} I_{ik} \leqq \tau \quad (1 \leqq i \leqq S) \tag{8}$$

ここで，$I_{ik} = r e_{ik}$ $(r \geq 1)$ となるように I_{ik} を設定することにすると，式 (8) が真ならば，

$$\frac{1}{r} \sum_{i=1}^{S} \sum_{k=1}^{K_i} I_{ik} = \sum_{i=1}^{S} \sum_{k=1}^{K_i} e_{ik} \leqq \frac{S}{r} \tau$$

が成り立つ．この時 $S/r \leqq C$ であれば式 (7) も満たされることになる（十分条件）．これは，与えられた S と C に対して

$$I_{ik} \geq \frac{S}{C} e_{ik} \tag{9}$$

となるように Web サーバアクセス間隔を設定していれば，全 Web サーバ間の調停を表す式 (7) を明示的に考慮しなくても通信リソースの配分が可能になるということを意味している．また，Web サーバ i からの取得ページ数 K_i は，このように与えられた I_{ik} に対して式 (8) が成立するように設定することになる．

なお，Najork ら [21] は，Web サーバ管理者からのクレームを避けるための経験則として，ページ取得時間の 10 倍の間隔を空けて Web サーバをアクセスするというルールを述べてい

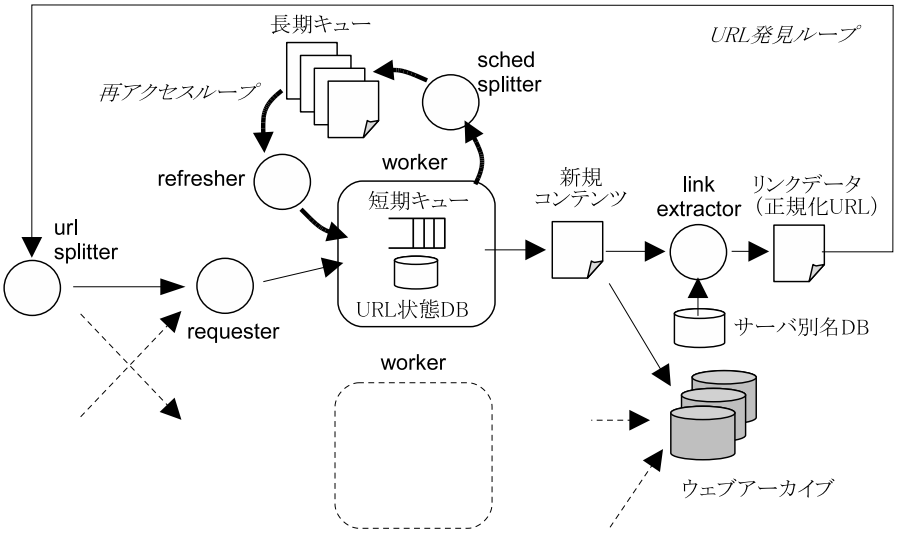

図 2　更新クローラの構成.

Fig. 2　Architecture of incremental crawler.

る．同様のルールがローカルリソース制約に対しても有効と言えるのは興味深い．

式 (9) **と** Web **サーバ負荷指標を組み合わせると**, Web **サーバアクセス間隔** I_{ik} **の満たすべき条件として次式を得る．**

$$L_i{}^{-1} \geqq I_{ik} \geqq \frac{S}{C} e_{ik} \quad (1 \leqq k \leqq K_i) \tag{10}$$

このような I_{ik} **が存在するためには，**

$$\frac{L_i{}^{-1}}{\max_k e_{ik}} = R_i \geqq \frac{S}{C} \tag{11}$$

が成立している必要があり，これがクローラリソース制約の下で Web **サーバ過負荷が起こらない条件となる．実際の** Web **サーバにおけるページ取得時間や比** R_i **の分布については** 6. **で述べる．**

5.　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実装した．図** 2 **はそのプロセス構成およびデータフローである．各プロセスはクラスタ構成の複数ノード上で並列に動作する．クラスタ全体を集中制御するプロセスは存在しない．以下，各プロセスの機能を述べる．**

a）　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．短期キューの容量は** Web **サーバアクセス間隔から決定され，**requester **からの新規** URL **と** refresher **からの既知** URL **で等分するようにしている．**

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（**3. **節の** T_{total}, T_{stable}, **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）．**

b）　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．**

c）　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．**

d）　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．**

e）　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**

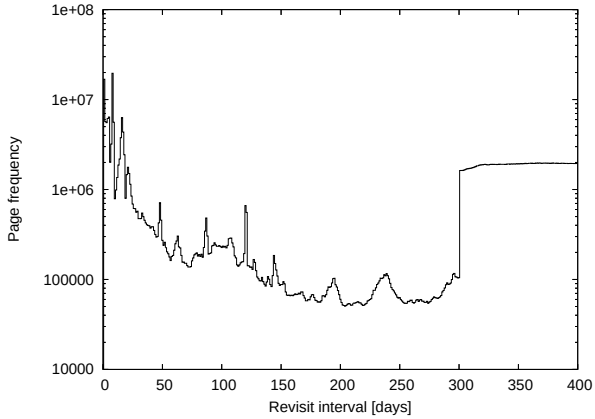

図 3 Web **ページ再アクセス間隔の頻度分布**.
Fig. 3 Histogram of web page revisit intervals.

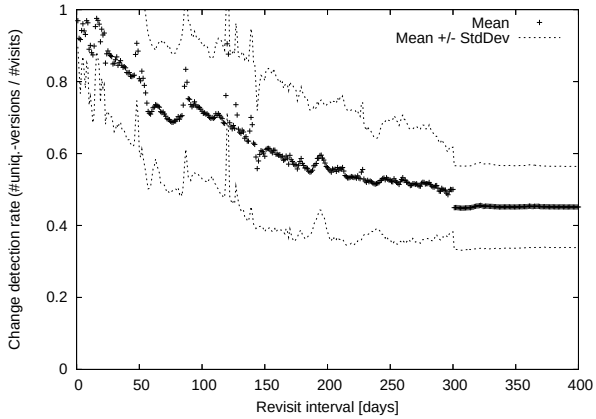

図 4 Web **ページ再アクセス間隔に対する更新検出率**.
Fig. 4 Change detection ratio vs. Web page revisit intervals.

数を適用することで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．

f） 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．**

なお，URL **発見ループにおいては収集対象を日本語** Web **ページ中心とするため，**Web **ページの記述言語（文字コード）に基づく制御を行っているが，本稿では説明を省略する** [22]**．**

6. 大規模クローリング実験による予備評価

6.1 Web ページスケジューリングに関する統計

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．クローラが動作するクラスタは当初の** 10 **ノード構成から最終的には** 32 **ノードとした．**

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．**

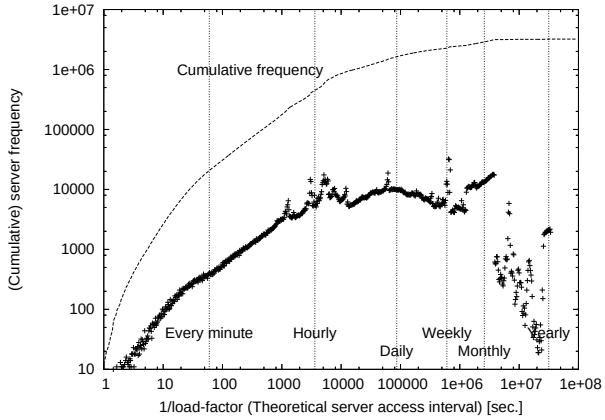

図 5 Web **サーバアクセス間隔理論値の累積頻度**.
Fig. 5 Cumulative frequency of theoretical web server access intervals.

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．更新クローラにおける** Web **ページアクセス間隔とその制御の妥当性については詳細な評価を必要とするため，稿を改めて述べることとしたい．**

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．ただし，**

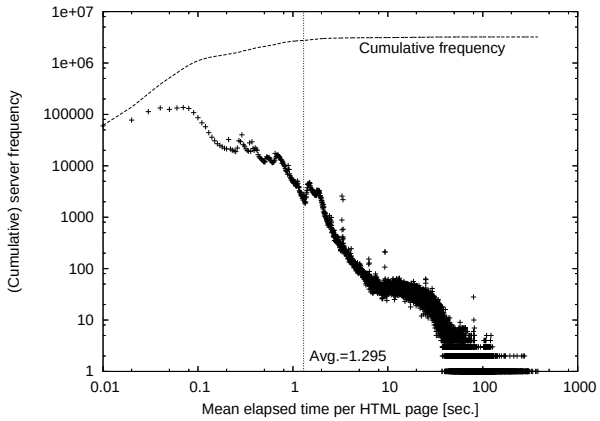

図 6 Web サーバ毎の平均ページ取得時間の累積頻度.

Fig. 6 Cumulative frequency of mean page-download time per server.

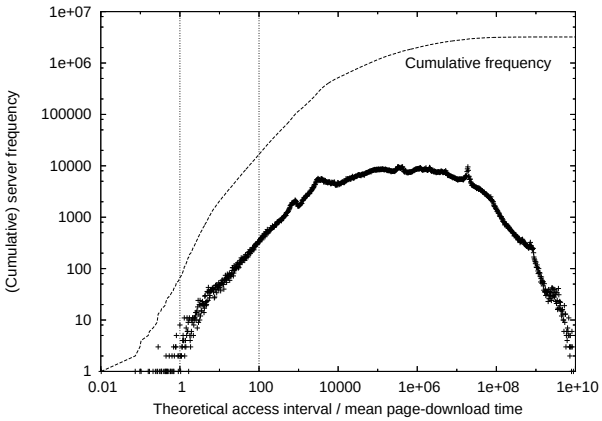

図 7 Web サーバアクセス間隔理論値と平均ページ取得時間の比の分布.

Fig. 7 Cumulative frequency of ratio of theoretical web server access interval to mean page-download time per server.

対象としたサーバは次節の分析と対応させるため，実験期間の最後の 12 日間にアクセスしたものとした．このようなサーバの数は 3,212,253，サーバ当たりの平均既知ページ数（12 日間にアクセスしていないものも含む）は 585.2 であった．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 分毎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．これらのサーバには 10 万単位のページが属しており，全てのページを定期的に再アクセスしようとすると容易に過負荷となってしまう．

実際のクローリングにおいては Web サーバアクセス間隔の下限は通常 1 分とし，1 万以上のページが属しているサーバについてのみ 5 秒とした．これは，多数のページを有する Web サーバは十分なリソースを持っているはずとの見込みに基づいている．

6.2 Web サーバアクセスに関する統計

ここでは実験期間の最後の 12 日分のクローリングログを用い，Web ページ取得に要した時間について調べた．単純に平均するとページ取得時間は 0.845 [s]，Web ページサイズ（HTML のみ）は 21806 [byte] であった．

図 6 は Web サーバ毎に求めた平均ページ取得時間の累積頻度を示す．平均ページ取得時間が 1 秒未満のサーバが大半を占めていることが分かる．

図 7 はサーバ毎のアクセス間隔理論値と平均ページ取得時間の比の分布を示したものである（平均と最大値の違いを除き，式 (11) の R_i に対応する）．逐次アクセスにおいては，アクセス間隔がページ取得時間を下回ることはできないため，比が 1 より小さい領域に属するサーバについてはリソースがどれだけあってもオーバフローが発生してしまうことを表している．このようなサーバは約 60 存在し，その内 20 サーバは同一ドメインに属していた．

また，32 のクローラノードそれぞれで 1000 コネクションまで同時に使用することにすると全体の通信リソース数 C は 32000 となり，サーバ数 S は約 320 万であるから S/C は約 100 となる．式 (11) から，サーバアクセス間隔理論値と平均ページ取得時間の比が 100 未満となっている約 1 万のサーバは，このリソース制約下では過負荷状態にあると言える．

6.3 過負荷状態解消に関する考察

Web ページ取得時間を考慮すると，全体の 1%未満ではあるものの過負荷状態となる Web サーバが存在することが分かった．ごく一部にはクローラにリソースを投入しても解消しないものもあった．これは，負荷指標の大きな Web サーバが大きなページ取得時間を要していることが原因と考えられる．

ページ取得時間の低減については，逐次アクセスの制限を解除し，複数 Web ページを並行して取得することが考えられるが，Web サーバの処理能力が限界に達している場合には効果が期待できない．ネットワーク経路の最適化（拠点の変更）なども考えられるがその効果は不明である．まずは負荷指標の低減が重要であろう．以下に Web サーバ負荷指標の低減による過負荷状態解消についていくつかのアプローチを考察する．

- Web ページ再アクセス間隔修正

過負荷状態ではスケジュール通りの Web ページアクセスができなくなり，得られる結果が予測不能となる．それよりは明示的に各ページの再アクセス間隔を理論値から修正し，全体のバランスを維持する方が望ましい．

具体的な修正方法としては，例えば各ページの再アクセス間隔を l 倍するというものが考えられる．これにより負荷指標は $1/l$ になる．効果としては各ページのサンプルを一様に間引くことに相当する．

または，各ページの再アクセス間隔に定数を加えても良い（d 日延長）．再アクセス間隔が小さく，負荷指標への影響が大きいページに集中的に作用し，その他のページへの影響を小さくできるという効果がある．

- URL のグループ化

各ページを一様に扱うのではなく，URL 文字列が類似するものをグループ化して処理する．非常に多くの URL が存在する場合，それらの一部がパラメータ化され，残りが共通になっていることが多い．パラメータとしては通販サイトにおける商

品 ID，ニュースの日付，掲示板等の書き込みの番号，等に加え，セッション ID のようなコンテンツ自体には影響しないものも考えられる．

URL 集合からテンプレートを抽出することで，代表元を残して残りを棄却したり，大きなアクセス間隔を付与したりできるようになる．

- Web ページ比較の高度化

Web ページ変更の検出条件を厳しくし，わずかな変更は変更と見做さないことにより，Web ページ再アクセス間隔の推定値が大きくなるようにする．HTML の構造に基づいてナビゲーション的な要素を無視したり，shingling [23] などの技法を用いる．

上記のアプローチはそれぞれ難易度が異なるが，実装を進め，実データに対する適用効果を評価して行きたい．

7. 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．更新クローラは連続的に動作し，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．32 ノードの PC クラスタ上に実装した更新クローラは数百万 Web サーバからの大規模 Web ページ収集を安定して実施することができた．

Web ページ再アクセススケジューリング方法は Poisson 過程による更新間隔推定に基づき，わずかな状態変数で導出できるよう簡略化を行った．これにより，大量の URL を扱うことが可能となった．また，Web ページに対する再アクセススケジューリングの実施可能性について考慮し，Web ページが属する Web サーバの負荷と，Web サーバの応答時間のそれぞれに対応する指標を導いた．これらの指標は単純であり，クローラが容易に維持することができる．

実クローリング動作から得られた Web サーバの特性を用いて評価した結果，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．この場合，Web ページ再アクセススケジューリングにフィードバックし，次善の状態を求める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行きたい．

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] M. Toyoda and M. Kitsuregawa: “What’s really new on the web?: identifying new pages from a series of unstable web snapshots”, Proc. of WWW ’06, pp. 233–241 (2006).

[3] International Internet Preservation Consortium: “netpreserve.org”. http://netpreserve.org/.

[4] 国立国会図書館：“インターネット資源選択的蓄積実験事業 (WARP)”. http://warp.ndl.go.jp/.

[5] 廣瀬：“国立国会図書館におけるウェブ・アーカイビングの実践と課題”, 情報処理学会研究報告, **2003**, 51, pp. 95–111 (2003).

[6] Internet Archive: “About IA”. http://www.archive.org/about/about.php.

[7] G. Mohr, M. Kimpton, M. Stack and I. Ranitovic: “Introduction to Heritrix, an archival quality web crawler”, Proc. of IWAW (International Web Archiving Workshop) (2004).

[8] K. Sigurosson: “Incremental crawling with heritrix”, Proc. of IWAW (2005).

[9] K. Sigurosson: “Managing duplicates across sequential crawls”, Proc. of IWAW (2006).

[10] B. E. Brewington and G. Cybenko: “How dynamic is the Web?”, Proc. of WWW9, pp. 257–276 (2000).

[11] J. Cho and H. Garcia-Molina: “The evolution of the web and implications for an incremental crawler”, Proc. of VLDB, pp. 200–209 (2000).

[12] J. Cho and H. Garcia-Molina: “Estimating frequency of change”, ACM TOIT, **3**, 3, pp. 256–290 (2003).

[13] J. Edwards, K. S. McCurley and J. A. Tomlin: “An adaptive model for optimizing performance of an incremental web crawler.”, Proc. of WWW ’01, pp. 106–113 (2001).

[14] J. Cho and H. Garcia-Molina: “Effective page refresh policies for web crawlers.”, ACM TODS, **28**, 4, pp. 390–426 (2003).

[15] S. Pandey and C. Olston: “User-centric Web crawling”, Proc. of WWW ’05, pp. 401–411 (2005).

[16] J. Cho and A. Ntoulas: “Effective change detection using sampling.”, Proc. of VLDB, pp. 514–525 (2002).

[17] 熊谷, 山名：“リンク構造を利用した Web ページの更新判別手法”, DEWS2004 論文集 (2004).

[18] K. C. Sia and J. Cho: “Efficient monitoring algorithm for fast news alert”, Technical report, UCLA (2005).

[19] sitemaps.org: “Sitemaps XML format”. http://www.sitemaps.org/protocol.html.

[20] M. Koster: “Robots exclusion”. http://www.robotstxt.org/wc/exclusion.html.

[21] M. Najork and J. L. Wiener: “Breadth-first crawling yields high-quality pages.”, Proc. of WWW ’01, pp. 114–118 (2001).

[22] 田村, ソンブーンワィワット, 喜連川：“特定言語で記述された web ページの選択的収集手法とその評価”, 電子情報通信学会論文誌 D, **J89-D**, 2, pp. 199–209 (2006).

[23] A. Z. Broder, S. C. Glassman, M. S. Manasse and G. Zweig: “Syntactic clustering of the web.”, Computer Networks, **29**, 8–13, pp. 1157–1166 (1997).