Vol.2012-DPS-150 No.44
Vol.2012-CSEC-56 No.44
2012/3/1

Web アーカイブを用いた Web ページの存在保証システムの提案

小澤 龍之介[†], 上原 稔[†]

今日、インターネットは巨大なデータベースとして利用されている。しかし、インターネットで公開されたページは閲覧者と無関係に公開者によって改変および削除される。その結果、ページを公的な文書で引用しても内容を証明することができない。そこで、本論文では、Web ページの存在保証を行うシステムを提案する。本システムでは、Web アーカイブを用いて Web ページの存在保証を行う。本システムは、Web アーカイブほど網羅的にかつ事前に収集する必要がないという点で、Web アーカイブと異なる。また、Web ページを事前に収集するのではなく、利用者が明示的にアクセスしたページのみを保証する。本システムでは、キャッシュサーバのログを監視し、キャッシュからアーカイブする。また、ある時間を指定して、その当時の内容を検索することができる。

Guarantee of Web Page Existence using Web Archive

Ryunosuke Ozawa[†], Minoru Uehara[†]

Today, Internet is usually used as large databases. However, Web pages published by an author is often modified and deleted by the author but not by viewers. Therefore, it is difficult for viewers to guarantee the contents of Web pages referred by a official documents. So, in this paper, we propose a system that guarantees Web page existence. This system checks whether the contents of a page accessed at a time is the same or not. It does not need to collect all pages and can guarantee only pages accessed at once. In this system, the log of cache server is always monitored and cached pages are copied from cache to archive. Furthermore, a user can search pages by a query specified with a time and gets the contents created/modified at that time.

[†]東洋大学 工学部 情報工学科
Dept. of Information and Computer Sciences, Toyo University.

1. はじめに

今日、多くの人々が PC のみならずスマートフォン等のデバイスを用いてインターネットにアクセスしている。これらの人々の中にはインターネットを巨大なデータベースとして利用しているものも多い。例えば、用語の意味を Wikipedia で調べたり、知らないことを知っている人に聞いたりして知識を得る。

研究者もインターネットで公開されている学術論文を参照して研究することがある。しかし、文献としての Web ページには永続性がない。内容が改定されたり、公開が取り消されたりするため、安心して参照することができない。そこで、文献の参照では、Web ページを参照するとき、確認した日時を明記する。該当する文献がその時点で存在していたという主張であるが、後日確認することが常にできる訳ではないので文献としての価値は低い。しかし、企業から発信させる情報は必ずしも学術論文に掲載されるとは限らず、重要な情報がインターネット上にのみ公開されることも珍しくない。よって、インターネット上の情報の価値が必ずしも低いわけではない。むしろ、価値の高い情報が存在するものの、永続的な保証のないことが問題である。そこで、本論文では、Web ページの存在保証を行うシステムを提案する。

Web ページの存在を保証するには Internet Archive などの Web アーカイブが有効である。これらのシステムの特徴は、過去に公開された Web ページを記録し、それをキーワード等の内容だけでなく、時間を指定して検索できることである。正確には、Web ページの存在保証は Web アーカイブのサブセットである。Web ページ自体を保存しなくても証明は可能である。しかし、その Web ページをアクセスした本人以外が、ページの内容を確認することも重要である。よって、本論文における存在保証は Web アーカイブと同じような機能を持つものとする。それでも Web アーカイブほど網羅的にかつ事前に収集する必要がないという点で異なる。提案システムでは、Web ページを事前に収集するのではなく、利用者が明示的にアクセスしたページのみを保証する。我々のシステムでは、キャッシュサーバを用いることで毎時的な Web ページの収集を不要としている。

本システムの効用は学術論文に限定されない。教材で OSS(Open Source Software)を使用する場合にも、OSS の古いバージョンの配布が停止されることで教材と不一致することがある。教材を改訂することが正しい対応であるが、教材を更新手間は大きい。本システムを利用することで過去の教材を確実に利用できる。また、インターネットのトラフィックを低減する効果もある。

本文の構成は以下の通りである。2 節で関連研究として Web アーカイブ等について述べる。3 節では存在保証システムについて述べる。4 節では、その評価を行う。最後に結論を述べる。

ⓒ 2012 Information Processing Society of Japan

Vol.2012-DPS-150 No.44
Vol.2012-CSEC-56 No.44
2012/3/1

2. 関連研究

Internet Archive[1]は 1996 年からのインターネットの Web およびマルチメディアコンテンツをアーカイブしており、その総データ量は Wikipedia によると 2009 年時点で 2PB に達している。また、Wayback Machine は該当 URL のページだけでなくそのつながりも再現する。

ただし、閉鎖された過去のサイトがすべて完全に見られるわけではなく、HTML ファイル以外は保存されないこともある。保存に成功したサイトであっても、年数経過によりアーカイブ上からでも見られなくなることがある。

ネット上のすべてのデータを収拾するサイトである性格上、そのホスティング環境は巨大なものである。2009 年までは HDD4 台を搭載した 800 台の Linux クラスターで運用していたが、2009 年春からは Sun Microsystems の Sun Fire X4500 63 台でホスティングされている。OS は Solaris10 で、1 台あたり 1 テラバイト HDD を 48 台搭載(＝総計 3 ペタバイト)、ファイルシステムは ZFS を採用している。施設も専用の Sun Modular Datacenter を使用している。

我々は、過去の研究において分散型サーチエンジン CSE(Cooperative Search Engine)を開発し、それを用いた Web アーカイブを提案した 2)。ここで、CSE とは、新鮮情報の検索に適した分散型サーチエンジンである。分散型サーチエンジンでは、各 Web サーバに局所サーチエンジンを設置し、それらを用いて検索時にメタ検索を行う。Web ページを収集する必要がないため公開直後の新鮮な情報でも検索できる。一方、多数のサイトにメタ検索すると応答時間が長くなり性能が低下する。そこで、検索条件に合致したページを所有するサイトのみ選択することで応答時間を集中型と同程度まで短縮した。

CSE を用いた Web アーカイブでは、J.F.Allen が提案する時区間論理 [3]に基づくクエリを処理する。ここで、X, Y を時点、A, B を時区間とすると、時区間論理では以下のような関係が考えられる。

X before Y ⇔ X < Y
X after Y ⇔ X > Y
X simultaneous-with Y ⇔ X = Y
X in A ⇔ start(A)<=X<=end(A)
A before B ⇔ end(A)<start(B)
A meets B ⇔ end(A)=start(B)
A overlaps B ⇔ start(B)<end(A)<end(B) and start(A)<start(B)
A starts B ⇔ start(A)=start(B)
A during B ⇔ start(B)<start(A) and end(A)<end(B)
A finished B ⇔ end(A)=end(B) and start(A)>start(B)
A after B ⇔ start(A)>end(B)
A met-by B ⇔ start(A)=end(B)
A overlapped-by B ⇔ start(B)<start(A)<end(B) and end(B)<end(A)
A started-by B ⇔ start(A)=start(B) and end(A)>end(B)
A contains B ⇔ start(A)<start(B) and end(A)>end(B)
A finished-by B ⇔ end(A)=end(B) and start(A)<start(B)
A cotemporal B ⇔ start(A)=start(B) and end(A)=end(B)

ここで、start(A)は時区間 A の開始時点、end(A)は。時区間 A の終了時点である。例えば、時区間 A=[t_0,t_1]に対して start(A)=t_0, end(A)=t_1 である。

CSE を用いた Web アーカイブでは、各サイトにページの履歴を残すことを条件としていた。この方式では集中型のような大規模ストレージを必要としないが、すべてのサイトに過大なシステム要件を課すこととなる。その後、我々は、大規模ストレージ構築のためのツールキット VLSD(Virtual Large-Scale Disks)を開発し、Web アーカイブのために安価かつ大規模なストレージを構築可能となった。

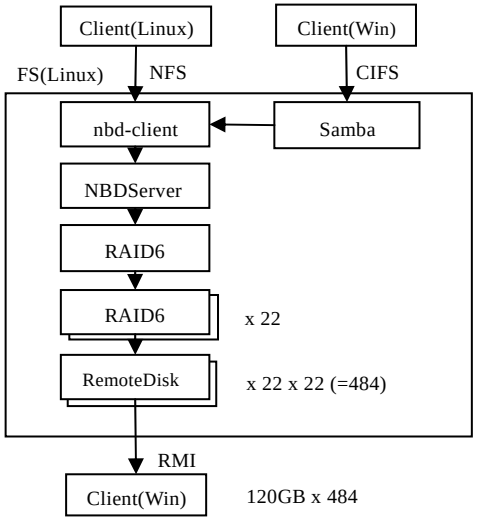

図 1. VLSD のシステム概要
Figure 1. The system overview of VLSD

VLSD は大規模ストレージ構築のためのツールキットであり、Java によるソフトウ

ⓒ 2012 Information Processing Society of Japan

Vol.2012-DPS-150 No.44
Vol.2012-CSEC-56 No.44
2012/3/1

ェア RAID 実装と NBD 実装を含む。VLSD は 100% pure Java であり、Java が動作するプラットフォームの上なら VLSD も動作する。そのため Windows や Linux が混在する環境に適している。

図 1 に VLSD を用いて分散ストレージを構成した例を示す。クライアントは 500 台存在し、その OS は Linux または Windows である。それらはそれぞれ NFS、CIFS で 1 台のファイルサーバと通信する。クライアントは同時に NBD サーバでもある。各クライアントでは空き容量を束ねた 1 つの NBD サーバが稼動する（従来のシステムでは複数の NBD サーバを稼動させなければならない場合があった）。ファイルサーバは Samba の稼動する Linux マシンである。ファイルサーバでは、クライアントの分だけ NBDDisk（後述）を作成し、22 の NBDDisk から 1 つずつ合計 22 の RAID6 を作成し、最後に 22 の RAID6 から 1 つの RAID6 を作成する。この RAID0 を NBD サーバで公開し、自分自身の NBD デバイスで参照する。

次に、本論文で提案する Web ページの存在保証システムで使用する要素技術について述べる。

要素技術の一つはプロキシサーバである。提案システムでは、一度アクセスした文書を永続的に保存することでアーカイブする。これにはキャッシュサーバを利用することが最も適している。キャッシュサーバはプロキシサーバの一種である。実際にキャッシュせずともアクセスを検出できればよいのでプロキシサーバを含める。

プロキシサーバには Squid, DeleGate などがある。

Squid は、HTTP や FTP に対応した現在最も広く利用されているプロキシソフトウェアである。数ある Web プロキシサーバの中でも、Squid はデータのキャッシング性能が高いことで知られており、以前にアクセスしたことのあるサイトに対しては、Web や FTP などのレスポンスを速くし、トラフィックを減少させることができる。ファイルをダウンロードしている途中で接続を切断しても、そのファイルのダウンロードを前回の続きから再開（リスタート）できる機能（レジューム機能）を実装している。レジュームに対応していないサーバからファイルをダウンロードする場合でも、Squid プロキシを利用すればレジュームできることがある。

DeleGate は、多目的な用途に利用される純国産のプロキシソフトウェアである。日本語対応のブラウザがなかった当時、日本語コード変換機能付きプロキシサーバとして広く普及した。プロキシの透過性よりもデータを付加価値付きで中継する機能に重点が置かれて開発されている。HTTP, FTP, NNTP, SMTP, POP, IMAP, LDAP, Telnet, SOCKS, DNS, TLS, SSH など非常に多くのプロトコルに対応しており、幾つかのプロトコルにおいてはキャッシュ機能の延長で文字コードの変換機能を利用することもできる。海外のユーザもおり、英語によるサイトやメーリングリストも公開されている。Windows や Mac OS X、各種 Unix 系 OS などで利用できる。

Apache は世界中で最も広く利用されている Web サーバであるが、これをプロキシサーバとして動作させることも可能である。初期設定を一部変更するだけで簡単にプロキシサーバとして動作させられるため、プロキシとしても広く利用されている。

Socks はプロキシのように内部ネットワーク（LAN）からインターネットへ接続できるようにするための方式である。これは普通のプロキシと異なり、一つの Socks サーバだけで HTTP、FTP、Telnet 等の TCP/IP を利用するほとんどのプロトコルで同時に利用できる。ただしプロキシのようなキャッシュ機能が実装されていないため、キャッシュによるコンテンツ取得の高速化は実現できない。

CGI プロキシは、リクエストのたびに CGI を起動させて、ユーザの代わりに目的のサーバからコンテンツを取得する機能を持つ CGI プログラムの呼称である。厳密にはプロキシサーバではないが、プロキシと類似した機能を持つためこのように呼ばれることがある。また、実際には CGI でなくても CGI プロキシと同様の機能を持つオンラインサービスも CGI プロキシに分類される場合がある。これは匿名性を追求する場合に利用されることが多く、速度面でのメリットはない。

Coral は、P2P 技術により Web コンテンツを効率的にキャッシングし Web サーバへの負荷を大幅に軽減することを目的としたプロキシである。ニューヨーク大学で開発され、世界中の教育・研究機関等で Coral のサーバが広く運用されている。動作方式は一見すると CGI プロキシと似ており、ユーザは Web ブラウザの設定を変更することなく、閲覧する Web ページのホスト名の後に「.nyud.net:8090」を付け加えるだけでこのプロキシを利用できる。

Coral サーバは世界中で広く運用されており、ユーザからのリクエストを受けた際に P2P 技術によって適切な Coral サーバがプロキシサーバとして選択されるため、各 Web ページにアクセスするたびに接続元のホストが変化する 。このプロキシはユーザのために用意されているのではなく、あくまでも Web サーバ運営者のために用意されている。そのため、匿名プロキシではなく、Web サーバに対して様々な独特の環境変数を出力する。また、Web サイトを作成した際に Coral を使用した URL を正式なサイト URL として公開すれば、Web サイトの閲覧者が増加しても転送容量を最小限に抑えることができ、Web サイトに対する DoS 攻撃を防ぐことが可能となる。

これらのうち最も存在保証システムに適しているのは Squid である。その理由は、数あるプロキシサーバの中でも、キャッシング性能が高いことがあり、本研究で用いるキャッシュ機能に一番適していると考えたからである。

もう一つの要素技術は KVS(Key-Value Store)である。KVS はクラウドに適した簡易データベースと考えられている。クラウドは Web アーカイブのような巨大なデータの格納場所として適している。また、その信頼性を維持するために高可用な KVS が望ましい。

KVS には Apache Cassandra, HBase などがある。

ⓒ 2012 Information Processing Society of Japan

Vol.2012-DPS-150 No.44
Vol.2012-CSEC-56 No.44
2012/3/1

Apache Cassandra は、Facebook 社が自社サービス向けに開発した KVS を OSS として公開したもので、現在は Apache プロジェクトの下で開発されている。P2P 通信によるクラスター管理を行っている。そのため、特別な管理ノードを持たない点が特徴である。また、Row Key のハッシュによるデータ分散を行い、HDFS のような分散ファイルシステムは使用しない。

各ノード内部でのデータのストレージへの格納には、Bigtable の memtable/SSTable と類似の方法を利用する。Thrift API を提供しており、C++, Java, PHP, Ruby,Python などのアプリケーションからの利用が可能である。

HBase は、さまざまな Hadoop のコンポーネントを利用して動作する。データストレージには、Hadoop に含まれる分散ファイルシステムである HDFS を使用している。クライアントからのデータアクセスには、各サーバーのメモリーキャッシュを利用することでパフォーマンスの向上を図っている。クラスター管理機能には、Hadoop プロジェクトで開発された ZooKeeper を使用している。各サーバーがキャッシュするデータの同期処理などに ZooKeeper の機能を利用している。(Native Java、Thrift(*)、REST の API を提供しており、特に Native Java API では、Hadoop の MapReduce Job のデータ入出力先として HBase を利用する機能を提供している)

これら KVS のうち今回の用途に適するものは Cassandra である。その理由は、Cassandra のデータモデルは、4 次元、または 5 次元のデータモデルを採用しており、それぞれキーを用いて管理することができるため、本研究でも、データベース上に URL やコンテンツを整理して格納することができ、後に検索しやすいということがあるからである。

3. 存在保証システム

ここでは、存在保証システムの設計と実装について述べる。初めに、システムの要件をまとめる。存在保証システムは、過去のある時点 t に内容 c を持つアドレス u の Web ページが存在していたかどうかを回答する。また、言い換えれば、ある時点 t における u の内容 c を求める。ただし、両者には若干の差がある。前者では内容のハッシュ H(c)を保持することでユーザが保管するコピーc'との一致を検証できる。H(c)=H(c')であれば高い確率で同一とみなせる。しかし、後者では c そのものを保持する必要がある。資源の制約が小さいのは前者であるが、有用性は後者の方が高い。よって、本論文では後者の意味で存在保証システムを定義する。なお、実際の検索では時区間を指定して検索する機能も必要である。時点および時区間の関係は時区間論理に従って指定する。

ある時点 t1 にアドレス u をアクセスした結果、内容 c が求められたという事実は、その返答メッセージに最終更新日時 t0 が記録されていれば、u→c の関係が時区間[t0,t1]において存在していたことを示す。逆にいえば、将来のある時点で u→c の関係が存在しなくなることまではわからない。ある時点 t2 で内容 c が c'に変更されたとき、t3>t2 において c'をアクセスしたとき、初めて u→c'が[t2,t3]に存在したことが分かる。ここで、[t1,t2]で実際に内容が変化していても確認できない。そこで、2 つの方式が考えられる。1 つは[t1,t2]にて観測された変化がないことから実際の変化もないと考える方式である。もう 1 つはアクセスしたアドレス u を定期的に訪問することでなるべく未観測の時区間を最小化する方式である。後者はプロキシサーバを経由する自動ロボットで実現できる。なお、c'には空すなわちページの削除も含まれる。本論文では必要以上のトラフィック増加を避けるため前者の方式を採用する。よって、実際にアクセスした結果、差除されたことが判明するまで、アドレス u のページは存在していたものとみなす。後者の方式を実装することは困難ではないので、実際の運用を通じて必要性が認められた場合には、改めて実装を行う。

図 2 に存在保証システムの構成を示す。本システムでは、アドレス u の内容が c であったことを、キャッシュサーバ Squid を用いて検出する。残念ながら Squid のキャッシュは永続的でない。Squid 自身にはキャッシュを永続化する機能はない。同一文書の内容は上書きされる。すなわちキャッシュサーバでは同じアドレス u の内容が c から c'に変化すると c は破棄される。これはキャッシュ自身の意味に根差した本質的な差である。アーカイブでは c を破棄せず永久に保持する必要がある。

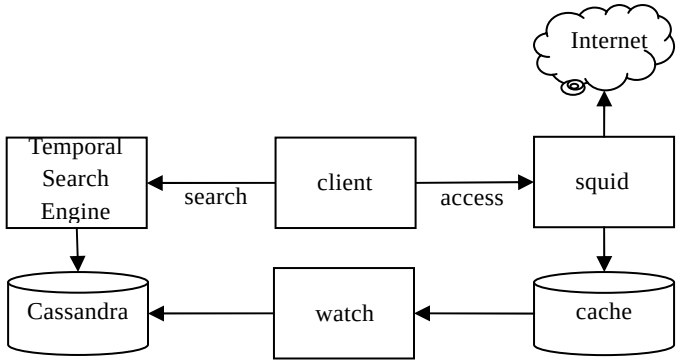

図 2. システム概要
Fig.2 System Overview

そこで、我々は store.log を監視し、追記された内容からキャッシュの更新を検出す

ⓒ 2012 Information Processing Society of Japan

Vol.2012-DPS-150 No.44
Vol.2012-CSEC-56 No.44
2012/3/1

る。キャッシュが更新されると監視システム（図 2 中の watch）がキャッシュの内容をアーカイブにコピーする。

長期間運用していると store.log が設定ファイルのログの最大サイズを超えたり、ファイルシステムの定める最大サイズを超えたりする可能性がある。このような場合に logrotate が行われる。Logrotate では極短時間でログファイルを切り替える。このとき、監視プログラムが新しい store.log をできる検出ようにする。具体的には記録しているサイズより実際のファイルサイズが小さくなった時、言い換えればシークエラーが発生した時、logrotate が行われてものと考える。

Cassandra を用いてアーカイブするために以下のようなデータ構造を採用する。Cassandra では、カラム＜スーパーカラム＜カラムファミリ＜キースペースというようなデータモデルを持つことができるため、これを採用する。

カラムの name を最終更新日時（月日時分秒）、value をコンテンツに指定する。次にスーパーカラムのキーに最終更新日時の年を指定する。さらに、カラムファミリに URL を指定し、キースペースのキーを設定することで、データベースをひとつのまとまりとして管理することができる。表で表すと表 1 のようになる。

表 1　Cassandra のデータ構造
Table 1. The date structure in cassandra

<table>
<tr><td colspan="6">KeySpace</td></tr>
<tr><td>key</td><td colspan="5">ColumnFamily</td></tr>
<tr><td rowspan="9">cassandra</td><td>key</td><td colspan="4">SuperColumns</td></tr>
<tr><td rowspan="3">URL</td><td>key</td><td colspan="3">Column</td></tr>
<tr><td rowspan="2">year</td><td>name</td><td rowspan="2">value</td></tr>
<tr><td>last update time</td></tr>
<tr><td rowspan="3">http://xxxx</td><td rowspan="2">2010</td><td>10.11.12:36:20</td><td>xxxxxxxx</td></tr>
<tr><td>11.15.11:15:30</td><td>xxxxxxxx</td></tr>
<tr><td rowspan="2">2011</td><td>10.09.08:05:40</td><td>xxxxxxxx</td></tr>
<tr><td>11.10.15:15:20</td><td>xxxxxxxx</td></tr>
<tr><td rowspan="3">http://yyyy</td><td rowspan="2">2010</td><td>05.10.14:20:50</td><td>yyyyyyyy</td></tr>
<tr><td>06.15.18:25:30</td><td>yyyyyyyy</td></tr>
<tr><td rowspan="2">2011</td><td>04.08.20:16:10</td><td>yyyyyyyy</td></tr>
<tr><td>10.20.10:08:20</td><td>yyyyyyyy</td></tr>
</table>

Cassandra のアーカイブが大きくなると単一ノードでは格納できなくなる。そのような場合はノードを増やす必要がある。物理的なノードを増やす場合、VLSD を用いてと他のマシンの空き容量を共有することができる。また、Amazon EC2 のような IaaS を用いてノードを増やすこともできる。その場合、なんらかの理由で停止したノードのデータは失われる。そこでアーカイブの損失を回避するために予め複製を持って運用することが望ましい。運用を楽にするには、あらかじめ Thin Provisioning を行っておくとよい。なお、アーカイブの運用方法については本論文の範囲外とし、本論文では可能性を示唆するにとどめる。

4. 考察

ここでは、提案した存在保証システムについて考察する。

現在、監視システムの監視周期を 1s としている。監視周期が長いと、その間にアクセスされるページが多くなり、アーカイブへのコピーに時間がかかる。しかし、一般論として、インターネットへのアクセスは LAN のアクセスより遅延が大きい。それゆえ、同数のファイルをコピーする時間は LAN の方が早い。

Squid のキャッシュサイズはデフォルトで 50MB しかない。これは Linux ディストリビューション DVD などを短期間でも保存できる容量ではない。また、DVD をダウンロードしている間も異なるクライアントからの要求に応える必要がある。よって、DVD サイズの容量をクライアントの数に応じて複数キャッシュできるサイズを確保する必要がある。具体的には、表 2 のようなコンテンツを想定している。

表 2　コンテンツの容量
Table 2. Capacity of contents

content	size	URL
Ubuntu	783MB	http://cdimage.ubuntulinux.jp/releases/8.04/ ubuntu-ja-8.04-vmware-i386.zip
Fedora	605MB	http://fedoraproject.org/ja/download-splash?file=http://download. fedoraproject.org/pub/fedora/linux/releases/16/Live/i686 /Fedora-16-i686-Live-Desktop.iso
CentOS	4.4GB	http://ftp.riken.jp/Linux/centos/6.0/isos/i386 /CentOS-6.0-i386-bin-DVD.iso

これにより、単一オブジェクトのサイズには最低でも一般的な DVD-R の容量である 4.7GB 程度の容量が必要だと考えられる。また、キャッシュに割り当てるメモリのサイズは、このシステムを同時に利用するユーザー数に応じて上昇する必要があるが、

ⓒ 2012 Information Processing Society of Japan

Vol.2012-DPS-150 No.44
Vol.2012-CSEC-56 No.44
2012/3/1

DVD-R サイズのコンテンツを複数同時に格納できる容量が必要だと言える。よって、本システムでは、10GB 程度の容量を確保し、複数のコンテンツをダウンロードできるようにすることとする。

キャッシュサーバには個人情報も保存される。保存された個人情報を他者が検索してしまうことを避ける必要がある。個人情報の検索を避けるには 2 つの方法がある。１つは検索時に検索結果から省く方法であり、もう１つは収集時に省く方法である。一般論として後者の方が安全である。個人情報はよく HTTPS あるいは POST で送受信される。よって、HTTPS あるいは POST メッセージをキャッシュしないことで個人情報の保存を避けることができる。しかし、この方法では HTTP かつ GET でアクセスしたパスワードつきの内容はアーカイブされてしまう。そこで、確実な方法は wget などにより（例えばパスワードなし等）異なるコンテキストにおいてもダウンロード可能であることを確認することである。

5. まとめ

本論文では、過去にアクセスした Web ページの存在を保証するシステムを提案し、その設計および実装について述べた。本システムでは、一度ユーザによってアクセスする必要がある点で、予めすべてのページを収集するアーカイブシステムとは異なる。しかし、利用者が増えるに従いアーカイブシステムと実質的に等しいものとなる。その意味ではソーシャルなアーカイブといえる。

今後は、自動的な更新確認、プライバシー保護、リンクの再現などについても実装を行う。自動的な更新確認では、アーカイブされたページの更新を比較的短い周期でチェックする。更新時間の精度を高めると同時に、ページが削除された時間も正確になる。プライバシー保護はアーカイブシステムにおいて重要な課題の一つである。一般公開されたパージであっても不適切な内容を含むページはアーカイブから抹消する必要がある。このような場合は、収集時に排除することは困難であるため検索時に排除する仕組みが必要となる。リンクの再現では、Wayback Machine と同様にある時点でのリンクを再現することで、過去の Web を閲覧可能とする。

参考文献

1) Internet Archive, http://www.archive.org/
2) Minoru Uehara, Nobuyoshi Sato: "Information Retrieval based on Temporal Attributes in WWW Archives", In Proceedings of the 11th International Conference on Parallel and Distributed Systems(ICPADS 2005), pp.756-761, (2005.7.20-22)
3) J. F. Allen: "Towards a general theory of action and time", Artificial Intelligence, Vol.23, pp.123-145, (1984)
4) プロキシサーバ入門, http://www.cybersyndrome.net/pg.html
5) IBM Linux at IBM, http://www-06.ibm.com/jp/domino01/mkt/cnpages7.nsf/page/default-001B017C
6) HMaster Japan, http://jp.hmaster.info/2010/06/hbase2-628.html
7) Cassandra Wiki, http://wiki.apache.org/cassandra/FrontPage_JP
8) API – Cassandra Wiki, http://wiki.apache.org/cassandra/API
9) Yuuta Ichikawa, Minoru Uehara: "Distributed Search Engine for an IaaS based Cloud", In Proc. of the 6th International Conference on Broadband, Wireless Computing, Communication and Applications(BWCCA2011), pp.34-39, (2011.10.26-28, Barcelona, Spain)

ⓒ 2012 Information Processing Society of Japan