

XML Web サービスのための大域的ファイル・サービスの提案

新城 靖† 阿部 聡‡ 板野 肯三†

† 筑波大学システム情報工学研究科 コンピュータサイエンス専攻

‡ 筑波大学大学院修士課程理工学研究科

URL: <http://www.cs.tsukuba.ac.jp/~yas/>

E-Mail: <yas@cs.tsukuba.ac.jp>

要 旨

この論文は、XML Web サービスのための大域的ファイル・サービスを提案している。このファイル・サービスでは、データはケーパビリティによりアクセス可能である。ケーパビリティはデータへの参照として働くので、ケーパビリティの利用は XML Web におけるデータの表現力を高める。大域的ファイル・サービスは、既存のケーパビリティを元に、それより弱いケーパビリティを作成する機能を提供する。この機能は、悪意があるプログラムである可能性がある遠隔のコンポーネントによる予期しないアクセスを防ぐ。この論文は、大域的ファイル・サービスの実現の1つとして、コールバック・ファイル・サーバについて述べている。このファイル・サーバを利用することにより、XML Web サービスのサーバは、従来のローカル・プログラムとより簡単に連携させることができる。

Proposal of a Global File Service for XML Web Services

Yasushi Shinjo† Satoshi Abe‡ Kozo Itano†

†Department of Computer Science, University of Tsukuba

‡Master's Program in Science and Engineering, University of Tsukuba

URL: <http://www.cs.tsukuba.ac.jp/~yas/>

E-Mail: <yas@cs.tsukuba.ac.jp>

Abstract

This paper proposes a global file service for XML Web Services. In this file service, data can be accessed with capabilities. Since capabilities act as references to data, using capabilities enriches the power of data expression in XML Web Services. The global file service provides a facility to produce a weaken capability based on an existing capability. This facility prevents unexpected access by remote components that can be by malicious programs. This paper describes the call back file server that implements the global file service. By using the server, the servers of XML Web Services can work together with conventional local programs more easily.

1 はじめに

現在、Web サービス、または、XML Web サービスと呼ばれる技術が注目を集め始めている。Web サービスとは、Web of services の意味で、ネットワーク上に分散されたソフトウェア・コンポーネントを統合して利用することである [14]。この時、コンポーネント間のデータのやり取りには、XML が使われる。また、通信プロトコルも HTTP 上に定義された XML-RPC や SOAP が使われる。

Web サービスは、従来の (バイナリ形式を用いる) RPC (Remote Procedure Call) が使われていたような場所で利用されている。その他に、Web サービスは、従来人間が Web ブラウザを通じてしか利用できなかった機能を、プログラムの中から使うために利用されている。そのような Web サービスの例としては、Google Web API や Amazon Web Services があげられる。その他に、Web サービスは、単体のアプリケーションの機能を拡張するために遠隔のサーバを呼び出すような形態で使われている。このように、現在の利用方法では、多くの場合、1 度に 1 つの Web サービスのコンポーネント (サーバ) が利用されている。今後は、単体のサーバだけでなく、複数のサーバを連携させることが重要になっていくと思われる。

複数の Web サービスのコンポーネント (サーバ) を連携させる場合、次のような機能が求められる。

1. 機能を制限する。たとえば、連携している別のコンポーネントには、特定のアクセス、たとえば、書き込みを禁止し、残りの機能だけを利用させる。こうすることにより、もし仮にそのコンポーネントに悪意があったとしても、被害の拡大を防ぐことができる。
2. 参照を含むデータ構造を扱う。処理すべきデータを値としてそのまま渡すだけでなく参照として渡すことで、分散型データ構造を構築することができる。
3. 通信オーバーヘッドを減らす。複数のコンポーネントを連携させる場合、クライアントをハブにしてデータを流すよりも、コンポーネント間で直接データをやり取りした方が通信量が削減できる。
4. XML Web サービスのコンポーネントの開発を容易にする。従来のローカル・プログラムと比較して、XML Web サービスのコンポーネン

ト (サーバ) の開発は困難である。従来のローカル・プログラムの構造を大きく変更することなく XML Web サービスのコンポーネントとして利用できるようになれば有益である。

5. ファイルからデータを入力し、ファイルに対して結果を出力するような既存のローカル・プログラムを変更することなく XML Web サービスのコンポーネントの連携させる。ローカル・ファイルの内容を、任意のタイミングで遠隔の XML Web サービスのコンポーネントがアクセスできれば便利である。

このような要求に答えるために、我々は、Web サービスでアクセス可能な大域的なファイル・サービスを用いる方法を提案する。このファイル・サービスを **大域的ファイル・サービス** と呼ぶことにする。大域的ファイル・サービスは、集中型オペレーティング・システムのファイル・システムや LAN におけるネットワーク・ファイル・システムが提供しているサービスと類似の機能を、XML Web サービスという形態で提供する。

この論文で述べる 大域的ファイル・サービスの特徴は、ケーバビリティを支援している点にある。ケーバビリティとは、ファイルやディレクトリ等のオブジェクトへの参照とオブジェクトに対するアクセス権をあわせ持つものである。ケーバビリティの導入により、Web サービスの世界においてデータを参照の形で指定することが可能になる。ケーバビリティという参照の導入により、Web サービスにおいてデータの表現能力が高まる。さらに、ネットワークを流れる通信量を押えることが簡単になる。それは、大量のデータを参照の形で持ち歩き、最終的にデータを生産するコンポーネントと消費するコンポーネントの間でのみ実際のデータの転送を行えばよいからである。

大域的ファイル・サービスでは、あるオブジェクトに対するケーバビリティから、それより弱いケーバビリティを生成する機能を提供する。たとえば、任意のアクセスが可能なファイルのケーバビリティから、読みみしか許さないようなケーバビリティを生成することができる。この機能により、悪意を持っている遠隔のコンポーネントによるある種の攻撃を防ぐことができる。

現在、大域的ファイル・サービスの 1 つの実装として、Web サービスのクライアント (または、特

定のサーバ) と同一のホストで動作するものを開発した。これを コールバック・ファイル・サーバと呼ぶ。コールバック・ファイル・サーバは、ローカル・プログラムと Web サービスのコンポーネント(サーバ) を連携させるためにも有用である。たとえば、コールバック・ファイル・サーバを使うことにより、ローカル・プログラムの出力したファイルを、遠隔の Web サービスのサーバの入力として利用することが簡単に行えるようになる。

この論文は、以下のように構成されている。2 章では、関連研究について述べる。3 章では、提案する 大域的ファイル・サービスの基本的な機能について述べる。4 章では、大域的ファイル・サービスの実装の 1 つである コールバック・ファイル・サーバについて述べる。5 章では、今後の課題についてまとめる。最後に 6 章でまとめを行う。

2 関連研究

インターネット上のファイル・サービスとして最もよく使われているものの 1 つに、WebDAV (Web-based Distributed Authoring and Versioning) があげられる [4]。WebDAV では、クライアントとサーバの間は、拡張された HTTP で結ばれる。WebDAV では、通常の HTTP の命令である GET や POST の他に、COPY や DELETE といったファイルに対する操作、MKCOL などのディレクトリに対する操作、および、PROPFIND 等の属性を操作する命令を用いる。WebDAV を利用すると、データを URL という参照で渡すことは実現される。しかしながら WebDAV では、部分的なアクセス権を定義することはできない。たとえば、WebDAV でよく用いられる Basic 認証や Digest 認証を用いた場合、あるコンポーネントが別のコンポーネントにパスワードと URL を渡してしまうと、渡されたコンポーネントは、元のコンポーネントと全く同じ操作を行うことができる。また、同じ HTTP といっても、Web サービスで標準的に使われているプロトコルである XML-RPC や SOAP と WebDAV 用の HTTP では形式が異なる。したがって、Web サービスのコンポーネントから見ると複数の通信形態に対応しなければならないという問題もある。

XMethodsFilesystem は、Xmethods というパブリックにアクセス可能な Web サービスのポータル

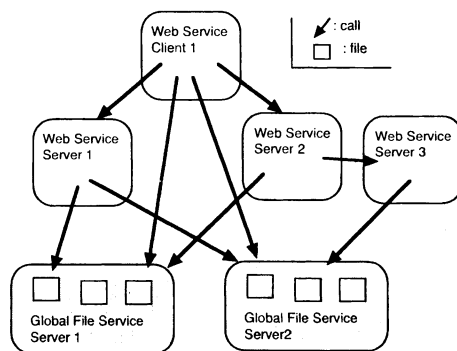


図 1: XML Web サービスのクライアント、サーバ、および、大域的ファイル・サービス

サイトが参照実装として提供しているファイル・システムである [15]。この Web サービスとして提供されているファイル・サービスは、基本的な操作 (list,read,write,remove) とパスワードによる認証機能を提供している。このシステムも部分的なアクセス権の設定を提供していない。

3 大域的ファイル・サービス

この章では、大域的ファイル・サービスに求められる機能とその利用方法を示す。

3.1 ファイル、および、ディレクトリ

本大域的ファイル・サービスは、通常のファイル・システムと同様にファイルとディレクトリという 2 種類のオブジェクトを提供する。本大域的ファイル・サービスのファイルは、通常の収集型システムにおけるファイルや NFS 等のネットワーク・ファイル・システムにおけるファイルと同様に、パッシブ・オブジェクトであり、単純なバイト列を保存し、それに対してストリーミング的なアクセスやランダムアクセスを許すものである。本大域的ファイル・サービスのディレクトリは、ファイルや他のディレクトリの名前を保持するものである。いずれのオブジェクトも、Web サービスのサーバにより提供される。

図 1 に、XML Web サービスの 1 つのクライアント、3 つのサーバ、および、2 つの大域的ファイル・サービスのサーバが動作している様子を示す。XML Web サービスのクライアントは、XML Web サービス

スのサーバと大域的ファイル・サービスのサーバを呼び出し、それらが提供するサービスを利用する。XML Web サービスのサーバは、XML Web サービスのクライアントに対してサービスを提供すると同時に、大域的ファイル・サービスのサーバを呼び出し、それらが提供するサービスを利用している。大域的ファイル・サービスの中には、ファイル、または、ディレクトリが存在している。

ファイル、および、ディレクトリが提供すべき手続きは、Java のクラス、および、NFS のインタフェースを参考にして定義した [6]。主なインタフェースの集合を以下に示す。

- IGlobalFile
- IGlobalInputStream
- IGlobalOutputStream
- IGlobalRandomFile
- IGlobalDirectory

3.2 ケーパビリティ

大域的ファイル・サービスのオブジェクト（ファイル、または、ディレクトリ）は、次の2種類の方法で指定される。

- 文字列の名前による指定
- ケーパビリティによる指定

前者の方法は、通常のローカルのファイル・システムで広く用いられている方法と同じ方法である。ファイルやディレクトリの内容をアクセスするには、それを開く操作を行う必要がある。開く操作では、結果として、ケーパビリティを返す。

ケーパビリティは、一般に、オブジェクトへの参照と許された操作のリストから構成される [12] [7]。ケーパビリティは、Unix 等のオペレーティング・システムでは、カーネルの内部に保護された状態で保存されており、利用者プロセスはファイル記述子を通じて間接的にしかアクセスすることができない¹。分散型オペレーティング・システム Amoeba では、ケーパビリティがユーザ・プロセスで扱えるようになっており、他のプロセスに送ることも容易である [7]。

¹Unix でも、sendmsg() システム・コールを用いて開いたファイルをパイプや Unix ドメインのソケットを通じて他のプロセスに渡すことができる。

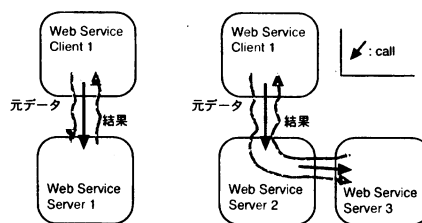


図 2: 従来の XML Web サービスにおけるデータの受け渡し

本大域的ファイル・サービスでは、ケーパビリティとして、WSDL (Web Services Description Language) [13] の文書を用いることにした。WSDL の文書は、Web サービスにおけるインタフェース記述を含む XML 形式の文書である。WSDL は、RPC におけるインタフェース記述言語と同様に、そのオブジェクトのインタフェース、すなわち、操作のリストを記述する。それに加えて WSDL による記述は、サーバやオブジェクトの位置を示す情報 (URL) を含んでいるため、それだけでオブジェクトの手続きを呼び出し、オブジェクトを操作することが可能である。したがって WSDL による記述は、オブジェクトへの参照として働く。

ケーパビリティは、盗聴や偽造などから保護されなければならない。まず、ケーパビリティを受け渡す場合には、暗号化された通信路や安全な通信路を用いる。WSDL による記述は、数kバイトの XML 形式の文書になる。実際のシステムでは、WSDL そのものではなく、それを取得するために必要な URL を用いることもある。この URL は、乱数を用いて予測されないものにする。

ディレクトリは、ケーパビリティと名前の対応関係のリストを保持し、名前解決機能を提供する。名前解決とは、名前から対応したケーパビリティを検索する操作である。その他にディレクトリには、一覧操作がある。ファイルやディレクトリを新たに作成する場合には、そのオブジェクトの文字列の名前を、既存のディレクトリに登録する。

3.3 参照によるデータの受け渡し

従来の Web サービスでは、データを主に値で渡していたことになる (図 2)。大域的ファイル・サー

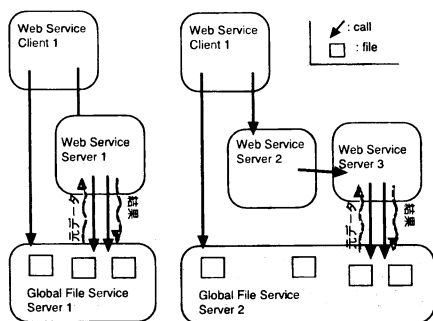


図 3: 大域的ファイル・サービスを用いた XML Web サービスにおけるデータの受け渡し (参照の利用)

ビスを利用すれば、データをキーバビリティ、または、文字列の名前とディレクトリのキーバビリティの組で渡すことができる。いずれの場合も、データを値ではなくて参照で渡すことを意味する (図 3)。

Web サービスにおける参照の導入は、プログラミング言語において参照 (ポインタ) が利用可能になったことと同等の効果をもたらす。この利点は、第 1 に、Web サービスにおけるデータの表現能力が高まることにある。第 2 に、大きなデータを加工する場合に、ネットワークを流れるデータ量を削減することが容易になる。図 2 と図 3 の右側では、Server 2 は、自分自身ではほとんど何もすることなく別のコンポーネント Server 3 を呼んでいる。このような場合、元データや結果のデータが大きければ、大域的ファイル・サービスを利用することでネットワークを流れるデータ量を削減することが自然に実現される。一方、大域的ファイル・サービスを使わない場合、このような通信を最適化するためには、複雑なプロトコルが必要になる。図 3 では、元データと同じファイル・サーバに処理の結果を書き戻している。その他に、結果を生成するサーバのサイトに大域的ファイル・サービスを実現し、そこにファイルを保存し、そのキーバビリティを返す方法も考えられる。

参照の導入により、Web サービスのサーバの構造が変化し、サーバが構築しやすくなるという利点もある。図 4 に、ファイルを分割する Web サービスのサーバの例を示す。このサーバは、引数として分割する行数、入力ファイルの名前、および、ディレクトリのキーバビリティを取る。そして、そのディ

```
public boolean split(int num,String inp,
    IGlobalDirectory dir){
    InputStream gfis = new GFIInputStream(inp,dir);
    BufferedReader r = new BufferedReader(
        new InputStreamReader(gfis));
    int f=0; boolean moredata=true;
    while(moredata){
        String ofn = makeOutputFilename(f++);
        OutputStream gfos = new GFOutputStream(ofn,dir);
        BufferedWriter w = new BufferedWriter(
            new OutputStreamWriter(gfos));
        moredata = copyLines(r,w,num)
    }
    return true;
}
```

図 4: ファイルを分割する XML Web サービスのサーバ

レクトリにファイルを作成し、分割したデータを保存する。GFIInputStream() と GFOutputStream() は、大域的ファイル・サービスのファイルをアクセスするためのスタブであり、それぞれ Java 標準の InputStream、および、OutputStream クラスのインスタンスを返す。それ以外の部分は、ローカル・ファイルを対象としたプログラムと全く同じである。このように、大域的ファイル・サービスを使う方法では、従来のローカル・プログラムの構造を変更することなく Web サービスのサーバとして利用することが可能になる。

参照の導入による問題点は、ごみ集め (Garbage Collection) を行う必要が生じることである。本大域的ファイル・サービスでは、開かれたファイルについては、有限の TTL (Time To Live) を設定することで対応する。TTL を過ぎてもファイルを利用し続ける場合には、リフレッシュを行う必要がある。開かれていないファイルで、ディレクトリに名前が登録されているものについては、TTL を無限に設定することで従来のファイル・サービスと同様に利用者による明示的な削除に期待することもできる。しかしながら、大域的ファイル・サービスでは、参照元と参照先の管理組織が別になることも考えられるため、従来のローカルのファイル・システムや LAN 用のネットワーク・ファイル・システムと比較して参照が失われやすい。このような状況に対応したごみ集めは、今後の重要な課題の 1 つである。

3.4 メタ・レベル・インタフェース

大域的ファイル・サービスでは、ファイルとディレクトリは、一般的なファイルやディレクトリとして

のインタフェースの他にメタ・レベルのインタフェースを持つ。メタ・レベルのインタフェースは、ファイルやディレクトリの内容ではなく、メタ情報を操作するものである。メタ・レベルのインタフェースは、次のような手続きを含む。

属性の設定 ファイルの属性（読み込み許可、書き込み許可、削除可能許可、最終更新時刻、TTL、最大の大きさなど）を設定する。

属性の取得 現在のファイルの属性を返す。

リフレッシュ TTL を再設定する。

より弱いカーナビリティの取得 いくつかのファイルの許可属性削除したカーナビリティを得る。たとえば書き込み許可を外したファイルのカーナビリティを生成する。

ファイルを開く ファイルの内容をアクセスするためのカーナビリティを返す。

ファイルを開く操作では、次のいずれかのカーナビリティが返される。

- ストリーミングな入力可能な状態のファイル
- ストリーミングな出力可能な状態のファイル
- ランダムアクセスにより入力、出力、または、入出力可能な状態のファイル

このように、ファイルを開くことは、より弱いカーナビリティを生成する方法のもうひとつの方法である。

3.5 高度なサービスの提供

大域的ファイル・サービスでは、従来のローカル・ファイルと同様に、ファイルという単純なオブジェクトを提供している。したがって、従来のローカル・ファイルに対して入出力を行うようなアプリケーションを XML Web サービスのサーバとして提供することが容易である。このようなサービスとしては、機械翻訳、(クロス) コンパイル、TeX 等の文書処理、コンピュータ・ウイルスのスキャン、自動バックアップなどが考えられる。

単純にファイルを入出力するようなサービスに加えて、大域的ファイル・サービスでは、堆積可能ファイル・システム (stackable file system) により高度な機能を付加することも容易である [5] [8] [9]。たとえば、一般的な機能としては、圧縮、暗号化、キャッシング、複製、バージョン管理、および、tar

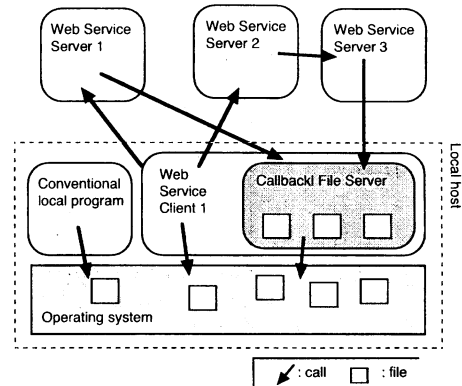


図 5: コールバック・ファイル・サーバ (大域的ファイル・サービスの実現の 1 つ)

等のアーカイブの展開があげられる。その他に、内容に依存したものとしては、文字コード変換、grep 等によるフィルタリング、イメージや音声等の形式の変換があげられる。

3.6 Java 言語のためのスタブ

Web サービスのサーバの開発は、Java 言語で使われることが多い。これを支援するために、本研究では、Java 言語用のスタブをクラスライブラリとして提供している。このクラスライブラリは、既に図 4 で示したように、Java 言語標準の FileInputStream や FileOutputStream と似せて作成した。したがって、そのような Java 言語標準の機能を使うアプリケーションと類似の構造で Web サービスのサーバを開発することが容易になっている。

なお、大域的ファイル・サービスは、他の Web サービスと同様に、WSDL を提供する。したがって、このクラスライブラリを利用しなくても、Web サービスに対応した任意のプログラミング言語やスクリプト言語から利用可能である。

4 コールバック・ファイル・サーバ

大域的ファイル・サービスの 1 つとして、コールバック・ファイル・サーバを実現した (図 5) [1]。これは、Web サービスのクライアント (または、特定のサーバ) と同一のホストで動作するファイル・

サーバである。コールバック・ファイル・サーバは、一般的な 大域的ファイル・サービスの機能に加えて、ローカル・ファイルを選択的に Web サービスのサーバにアクセスさせる機能を持つ。これにより、ローカル・プログラムと Web サービスのサーバを連携させることが可能となる。

Web サービスのクライアント（以下クライアント）は、アクセスさせたいローカル・ファイルの名前と初期属性（アクセス許可や TTL）をこの コールバック・ファイル・サーバのサーバ（以下ファイル・サーバ）に渡す。すると、ファイル・サーバは、ファイル・オブジェクトを生成して、そのケーパビリティ (WSDL) を返す。クライアントは、そのケーパビリティを遠隔の Web サービスのサーバに渡す。

クライアントは、このファイル・サーバ内にディレクトリを作成し、ローカル・ファイルから作成したファイル・オブジェクトを登録することもできる。この機能は、機密性を保つために役立つ。たとえば、ローカル・ファイルの名前とは別の名前で登録することで、ローカル・ファイルの名前を遠隔の Web サービスのサーバに隠すことができる。

Web サービスのサーバは、許可されていれば、与えられたディレクトリ内に新たにファイルを作成したり既存のファイルの内容を更新することができる。本コールバック・ファイル・サーバでは、そのようにして書込まれたファイルは、直接ローカル・ファイルに保存するのではなく、一時ファイルに保存する。クライアントは、書込まれたファイルの内容を検査し、問題がなければ、ローカル・ファイルに移動する。

コールバック・ファイル・サーバは、Java 言語と Web サービスのためのフレーム・ワーク Glue [3] を利用して実現した。全体で 1000 行の大きさになる。コールバック・ファイル・サーバは、現在の所、Web サービスのクライアントの権限で開いたファイルを扱うために、単独のプロセスではなく、クライアントとリンクして使うクラス・ライブラリという形態になっている。オペレーティング・システム・レベルの開いたファイルを渡す機能を利用すれば、別プロセスで動作させることもできる。

5 今後の課題

3 章では、大域的ファイル・サービスの基本的な機能について述べた。このファイル・サービスは、ファイル、および、ディレクトリという 2 つのオブジェクトを提供する。現在提供されている機能は、Java の File クラス、および、NFS のインタフェースを参考にして設定した。また、これらのオブジェクトは、名前、または、ケーパビリティにより参照される。既存のケーパビリティから弱いケーパビリティを生成できる所に特徴がある。3.3 節で述べたように、参照によるデータの受け渡しが可能になるという利点があるが、同時に、ごみ集めという課題が残された。

現在の 大域的ファイル・サービスの機能を通常のファイル・サービスと比較すると、大域的ファイル・サービスでは提供されていない機能として、同期に関する機能があげられる。たとえば、通常のファイル・システムはファイルをロックする機能を持っているが、大域的ファイル・サービスのファイルにはそのような機能はない。分散システムにおいてロック機能を提供しようとする、様々な困難な問題に直面することになるので、現在慎重に検討している。

ロック機能よりも先に、トランザクションの機能を提供したいと考えている。トランザクション機能は、J2EE (Java 2 Enterprise Edition) [10] でも標準的にサポートされている。トランザクションでは、デッドロックが起きないのでプログラミングがしやすいという利点もある。さらに、長期トランザクションやバージョン管理の機能を加えることも考えている。実現方法としては、Subversion [2] のようなバージョン管理機能を持つファイル・システムをバックエンドに利用することを検討している。

3 章で示した 大域的ファイル・サービスの基本的な機能では、ケーパビリティを中心的に使い、利用者や役割 (role) の概念を排除している。ケーパビリティの利用は、Java によるプログラミングや XML Web サービスとはよく適合することがわかった。しかしながら、どこかのタイミングで利用者認証機能との接点が生じるはずである。たとえば、コールバック・ファイル・サーバでは、ローカルのオペレーティング・システムにログインする時点で利用者認証が行われている。今後は、本大域的ファイル・サービスと大域的に有効な利用者認証の機能といかに連携させるかについて考えていく。

大域的ファイル・サービスでは、バイト列という単純なファイルだけを提供している。このことは、3.5節で述べた高度な機能を追加しやすくしている。単純なバイト列の他に、永続的なオブジェクトとしてよく使われるものに、ハッシュ表やB木といった索引付けされたファイルがある。このような機能は、ローカルのファイル・サービスと同様に、大域的ファイル・サービス上のサービスとして提供できるかもしれないが、効率面を考えると特別に支援することもあると考えられる。さらに、JavaSpaces が提供しているような分散データ構造、同期、および、イベント機能も有用であると思われる [11]。今後はこのような機能を提供することも検討していく。

6 おわりに

この論文では、XML Web サービスのための大域的ファイル・サービスの機能について述べた。大域的ファイル・サービスの特徴は、ケーバビリティを支援していることである。ケーバビリティは、オブジェクトへの参照として働く。したがって、Web サービスにおいてデータを参照の形態で指定することが可能になり、データの表現力が大いに高まる。また、大きなデータを扱う場合、自然にデータの生産者と消費者の間で直接通信が行われるようになる。ケーバビリティは、また、オブジェクトへのアクセス権としても働く。大域的ファイル・サービスでは、より弱いアクセス権を持つケーバビリティを作成する機能を支援している。これにより、悪意を持っている可能性のある遠隔のサーバによる予期しないアクセスを防ぐことができる。また、大域的ファイル・サービスのための Java 言語用のクラスライブラリを利用することにより、ローカル・アプリケーションと類似の方法で XML Web サービスのサーバを構築することが可能になる。

この論文では、大域的ファイル・サービスの実現の1つとしてコールバック・ファイル・サーバについて述べた。コールバック・ファイル・サーバを使うことにより、安全性を保ちながらローカル・ファイルを遠隔にある Web サービスのサーバにアクセスさせることが可能になる。

今後の課題は、トランザクション機能や利用者認証機能を付け加える事である。また、ごみ集めに関する問題を解決する必要がある。

参考文献

- [1] 阿部聡, 新城靖, 板野肯三: XML Web サービスのためのコールバック用ファイルサーバ, 情報処理学コンピュータシステム・シンポジウム ポスターセッション (2003). <http://www.ipsj.or.jp/sig/os/cs2003/pd/abe-s.pdf>.
- [2] Collins-Sussman, B., Fitzpatrick, B. W. and Pilato, C. M.: *Version Control with Subversion*, O'Reilly (June 2004).
- [3] Glass, G.: *Web Services: Building Blocks for Distributed Systems*, Prentice-Hall (2001).
- [4] Goland, Y., Whitehead, E., Faizi, A., Carter, S. and Jensen, D.: *HTTP Extensions for Distributed Authoring - WebDAV*, RFC2518 (1999).
- [5] Guy, R. G., Heidemann, J. S., Mak, W., Page, Jr., T. W., Popek, G. J. and Rothmeir, D.: Implementation of the Ficus Replicated File System, pp. 63-72 (1990).
- [6] Microsystems, S.: *NFS: Network File System Protocol Specification*, RFC1094 (March 1989).
- [7] Mullender, S. J., van Rossum, G., Tanenbaum, A. S., van Renesse, R. and van Staveren, H.: Amoeba: A Distributed Operating System for the 1990s, *IEEE Computer*, Vol. 23, No. 5, pp. 44-53 (1990).
- [8] Shinjo, Y. and Kiyoki, Y.: The Object-Stacking Model for Structuring Object-Based Systems, *1992 Int. Workshop on Object Orientation and Operating Systems*, Dourdan (France), IEEE Comp. Society Press, pp. 328-340 (1992).
- [9] 新城靖: 外部プログラムとキャッシングを利用した堆積可能汎用フィルタ・ファイル・オブジェクトの実現方式, 情報処理学会論文誌, Vol. 41, No. 6, pp. 1708-1722 (2000).
- [10] Sun Microsystems: *Java 2 Platform Enterprise Edition v 1.4 API Specification* (2003).
- [11] Sun Microsystems: *JavaSpaces Service Specification Version 2.0* (2003).
- [12] Tanenbaum, A. S.: *Distributed Operating Systems*, Prentice-Hall (1995).
- [13] The World Wide Web Consortium: *Web Services Description Language (WSDL) 1.1* (2001). <http://www.w3.org/TR/wsd1>.
- [14] W3C Working Group: *Web Services Architecture* (February 11, 2004). <http://www.w3.org/TR/ws-arch/>.
- [15] XMethods: *XMethods Filesystem Service* (1999). <http://www.xmethods.net/xfs/>.