データベースシステム 116－8
（1998. 7. 8）

オブジェクト指向データベースによる WWW-DB 連携システムの高速化と性能評価

安村 義孝

NEC C&C メディア研究所

〒216-8555 川崎市宮前区宮崎 4–1–1
tel: 044–856–2133
e–mail: yasumura@ccm.cl.nec.co.jp

WWW-DB 連携システムの高速化のために、オブジェクト指向データベースを利用した実装方法とその性能評価について述べる。従来の WWW-DB 連携システムは、複数ユーザからのアクセス集中などにより性能上の問題がある。オブジェクト指向データベース管理システムはクライアント側にオブジェクトキャッシュを持ち、高速なオブジェクトアクセスが可能であるため、HTTP サーバと同一プロセス空間でデータベースアクセスを行えばかなりの効果が期待できる。そこで、データベースアクセスのためのセッション管理をマルチスレッド制御と統合させ、HTTP サーバ内で稼働可能にした。性能評価の結果、CGI を利用した連携方法に比べて数倍の性能向上を確認した。

The Speedup Method and its Performance Evaluation of WWW-DB Integration Systems using Object-Oriented Databases

Yoshitaka Yasumura

C&C Media Research Laboratories, NEC Corporation

4–1–1 Miyazaki, Miyamae-ku, Kawasaki 216-8555
tel: +81–44–856–2133
e–mail: yasumura@ccm.cl.nec.co.jp

In this paper, the speedup method and its performance evaluation of WWW-DB integration systems using object-oriented databases are described. Conventional WWW-DB integration systems have performance problems that are due to a concentration of multi-user access. Because an application program of object-oriented databases have an object cache in client sides, it is possible to access databases in the HTTP server processes using them. Then, database session management and multi-thread control are integrated in order to be able to run on HTTP servers. As a result of its performance evaluation, this system is several times as fast as the conventional system using CGI.

1 はじめに

WWW (World Wide Web) はインターネットやイントラネット上でシステムを構築する場合に必須となり、今まで利用されてきたアプリケーションとの連携も盛んに行われるようになってきた。このようなシステムを考える場合、情報の格納場所となるデータベースとの連携が重要であり、様々なWWW-DB連携システムが開発されるようになった。通常のWWW-DB連携システムはバックエンドに関係データベース管理システム (RDBMS) を持ち、ユーザから指定された条件に基づいてデータベースの検索処理を行い、その検索結果として得られたテーブルのタプル内の属性データを、何らかの方法でテキスト形式に変換してHTML文書に埋め込むことが行われる。これらの複雑な処理を行わなければならないため、WWW-DB連携システムにはかなりの負荷がかかるが、従来のシステムではCGI (Common Gateway Interface) を利用しているものが多く、性能上に問題があった。

CGIの代わりとして、最近の商用HTTPサーバにはサーバAPIを持つものがある。サーバAPIはCGIと同様にHTTPサーバ側でアプリケーション機能を呼び出すものであるが、CGIの様にユーザからアクセスされる度に新たなプロセスを生成して処理を行うのではなく、HTTPサーバと同一のプロセス空間で実行することができる。ユーザからのアクセス要求は各スレッドに割り当てられるため、CGIのためのプロセスを生成する手間を省け、サーバマシンに負荷をかけないこと以外にも、複数のスレッドでメモリを共有できるというメリットがある。

また、バックエンドで稼働するRDBMSはサーバ側でデータベース処理を行うため、SQLによる検索やストアードプロシージャなどのデータベースへのアクセス要求が発生する度に、サーバとの通信を行わなければならない。これに対して、オブジェクト指向データベース管理システム (OODBMS) はクライアント側にオブジェクトキャッシュを持ち、データベース内のオブジェクトにアクセスする際には、一旦クライアント側のオブジェクトキャッシュに必要なオブジェクト (または、オブジェクトを含むページ) を取得してからアクセスする。これらのオブジェクトが一時ファイルへスワップアウトされない限り、2度目以降のアクセスはそのオブジェクトキャッシュにアクセスするだけなので、無駄な通信コストを避けることができる。

以上の点を考慮して、本稿では、WWW-DB連携システムの高速化を図るためにHTTPサーバが提供するサーバAPIを活用し、OODBMSのデータベースへアクセスする方式について述べる。OODBMSが利用するオブジェクトキャッシュを全スレッドで共有し、ユーザからデータベースへのアクセス単位となるセッションの管理をマルチスレッドの制御と統合した。このような構成にすることで、HTTP内で同時に実行するスレッド数を制限できるようにもなる。文献管理データベースによる性能評価の結果、CGIによるシステム構成に比べて数倍の高速化が確認できた。

2 WWW-DB連携システム

WWWでは、インターネット上に存在するアクセス可能な資源について、その公開してある形態と所在がURL (Uniform Resource Locator) によって示されている。ユーザがサーバに蓄積されているHTML文書を閲覧する場合には、クライアント上のWWWブラウザを用いてURLをサーバに投げ、サーバではそのURLによって指定された文書ファイルをクライアントに送信する。返却された文書ファイルには画像などの付加的なデータが添付されることが多く、それぞれのデータに対して同様の処理が行われる。また、同一サーバ内や他サイトにある文書ファイルへのリンクが付けられている場合があり、それらのリンクをユーザが自由に辿るという方法でサービスが行われることになる。

しかし、WWWが単体で実現できることには限界があるため、ユーザの要望に合わせて、動的に情報提供するためにはデータベースとの連携が必要になる [6]。WWWのサーバはいわば書庫のようなものであり、リンク付けされたHTML文書を辿っていき、目的の情報を取得するという利用形態に限られてしまう。サーバに存在する豊富な情報から検索などにより目的の情報を得たり、データを更新するようなことは単体の機能としては実現できない。そこで、HTTPサーバが提供するCGIを利用すると、サーバ上であらかじめ用意しておいた手続きを実行することができるようになる。HTTPサーバは、WWWブラウザから送られてきたアクセス要求がCGIを利用するものであると、CGIプログラムを実行するためのプロセスを起動して処理が終了するのを待つ。処理結果はHTML文書としてWWWブラウザに返却する。

通常のWWWシステムでは、サーバに蓄積されているHTML文書を単に返却するだけであったが、データベースとの連携を考慮してCGIを利用する場合には、HTML文書を動的に生成しなければならないためにこの手順が少し変わる。データベースを利用する場合には、テキストやイメージ、表データなどのコンテンツデータをデータベースに格納しておき、それらを合成することによってHTML文書を動的に生成して送信する。この様子を図1に示す。ただし、この図では全ての資源をデータベースに格納しているが、テキストやイメージは別ファイルとして管理し、データベース内にはファイル名だけ保持するという形態の方が主流である。

一般的なWWW-DB連携システムの構成を図2に示す。処理手順は次のようになる。

1. ユーザはWWWブラウザを利用して、HTMLのFORM機能などによりデータベースへ問合せを発行する際の検索条件を、URLに付加するParameterに指定してHTTPサーバにアクセス要求を出す。
2. このアクセス要求を受け取ったHTTPサーバは、CGIによりゲートウェイプログラムのためのプロ

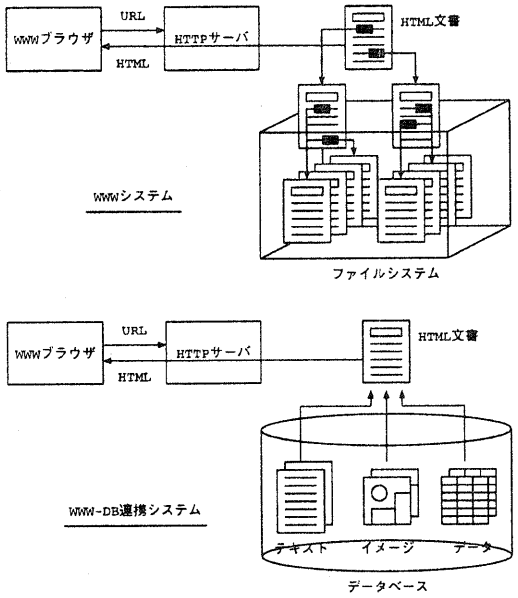

図 1: WWW システムと WWW-DB 連携システム

セスを起動する。

3. Parameter に指定された検索条件から SQL の問合せ文を生成し、データベースサーバに対して問合せを発行する。
4. データベースサーバ内で検索処理を行い、結果をゲートウェイプログラムに返却する。
5. 結果データをテキスト形式に変換し、HTML テンプレート内の指定された位置に変換したデータを埋め込んで、HTML 文書として HTTP サーバを経由して WWW ブラウザに返却する。

以上のようにして、データベース内のデータを利用して動的に HTML 文書を生成できるようになるが、いくつかの問題点がある。HTTP サーバとデータベースサーバの仲介を行うゲートウェイプログラムが CGI を通して呼び出されるため、ユーザからのアクセスの度にゲートウェイプログラムのためのプロセスが起動することになる。CGI を利用したシステムの高速化手法 [2] も存在するが、根本的な改善にはならない。データベースサーバへの問合せ発行と結果の受け取りのためのプロセス間通信のオーバヘッドもある。また、一連の処理が終わるとこのプロセスが終了してしまうため、セッション継続を行う場合にはセッション情報をどこか他の場所に記憶しておかなければならない。複数ユーザが同時にアクセスする状況を考えると、これらの問題点が更に重要視される。

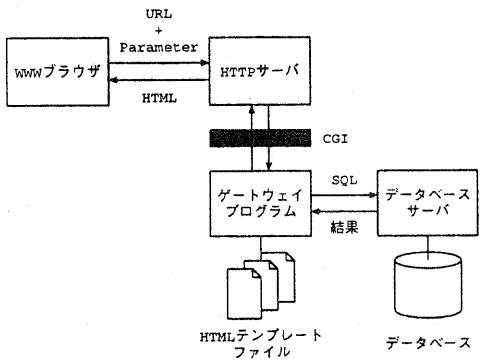

図 2: WWW-DB 連携システムの構成

3 システム構成

本 WWW-DB 連携システムでは、従来の WWW-DB 連携システムが抱えていた問題点を回避するために、次のような方針に基づいて設計を行った。

- HTTP サーバが提供するサーバ API やマルチスレッド制御を採用して、オブジェクトキャッシュの引き継ぎなどの実現により、高速性が要求される WWW アプリケーションにも適応できるようにする。
- 複数の手法 (プログラミング言語) でデータベースアクセスを記述でき、外部のクラスライブラリ等の拡張機能を容易に利用できるような構成にする。
- 複雑に関連付けられた大量のコンテンツデータを管理し、そのコンテンツを利用した WWW システム上でのサービスを容易に実現することが可能な枠組を提供する。

システム構成を図 3に示す。本システムでは、我々が開発したオブジェクト指向データベース管理システム PERCIO [7] を利用し、PERCIO が管理するデータベースにアクセスする。サーバ API または CGI を通して HTTP サーバと連携するデータベースサーバを中心として、実際のデータベースへのアクセスを行うデータベースアクセスモジュールと、CGI 経由でデータベースサーバとのやり取りを行う際に利用されるゲートウェイプロセスから構成されている。サーバ API としては ISAPI (Internet Server API) [3] と NSAPI (Netscape Server API) [5] に対応している。

データベースサーバ内では、複数のユーザからのアクセス要求をスレッドによって処理する。HTTP サーバから受け渡されたアクセス要求は制御スレッドで受け付け、該当する定義ファイルを取得する。その後、制御スレッドから実行スレッドに制御を移し、該当するデータベースアクセスモジュールを動的にリンクしてデータベース処理を行う。データベースアクセスの結果データは、HTML テンプレートファイルの指定された場所に埋め込まれて、ユーザに返却するための

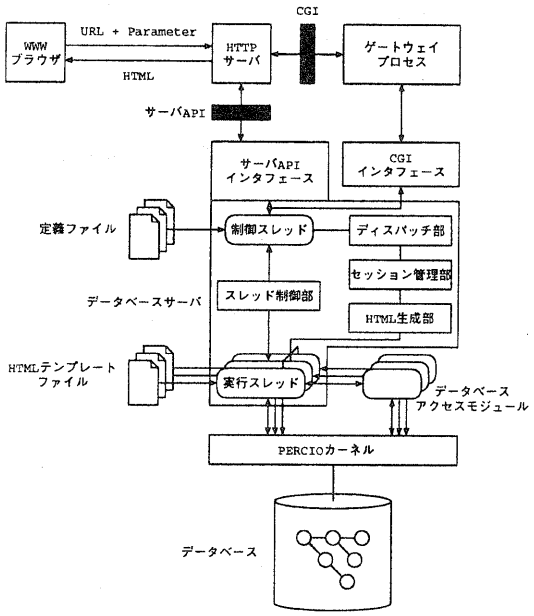

図 3: システム構成

HTML 文書が生成される。データベースサーバの各モジュールの説明は以下の通りである。

- **インタフェース部**

  サーバ API インタフェースはサーバ API のインタフェースを通して、従来の CGI インタフェースは CGI のインタフェースを通して、ユーザからのアクセス要求である URL やパラメータを HTTP サーバから受理して制御スレッドに渡す。また、データベースアクセスの後に生成された HTML 文書を各インタフェースを通して HTTP サーバに返却する。CGI インタフェースの場合は、データベースサーバがデーモンプロセスとなってサーバマシン上に常駐し、ゲートウェイプロセスとはプロセス間通信でデータの受け渡しを行う。パラメータの解析や日本語処理などもこのモジュールで行われ、処理に必要なセッション情報やディレクトリ情報も取得される。

- **ディスパッチ部**

  データベースへのアクセス要求を、該当する実行スレッドに割り振る役割をする。実行スレッドはセッションごとに生成されるため、継続中のセッションに対応する実行スレッドにそのセッション中のアクセス要求を処理させる。新規のセッションの場合は新たに実行スレッドを生成してそれに制御を移す。実行スレッドの数はデータベースサーバ内であらかじめ指定しておくことができ、その数を越えるような場合には空きスレッドができるまで次の処理が待たされる。

- **セッション管理部**

  一人のユーザからの連続した複数のアクセス要求をセッションという一連の処理として扱うためにセッション情報を管理する。セッション情報は、実際にはセッション ID としてデータベースサーバと WWW ブラウザの間で受け渡されることにより、セッションの継続をシステム側で保証する。セッション内ではデータベースに対するオープン / クローズやトランザクションのスタート / コミットを任意の時点で行える。また、タイムアウトによるセッションの終了を行う責任も持つ。

- **HTML 生成部**

  指定された HTML テンプレートファイルにしたがってユーザに返却するための HTML 文書を生成する。HTML テンプレートファイル内には、データベースアクセスモジュールで取得したデータベース内のデータを埋め込むための指定があり、それにしたがって必要なデータをデータベースから取得して HTML 文書内に展開する。

- **スレッド制御部**

  データベースアクセスや HTML テンプレートファイルにしたがった HTML 文書の生成を行う実行スレッドを制御する。データベースサーバ内に存在する全ての実行スレッドの情報と、それらの実行スレッドが必要とするデータベースやオブジェクトキャッシュなどのリソース情報を実行スレッドが終了する (セッションが終了する) まで保持する。

4 データベースアクセス

ユーザから URL により指定された定義ファイルには、データベースアクセスで利用するデータベースアクセスモジュールと HTML テンプレートファイルなどが指定されている。このうち、データベースアクセスモジュールは実際にデータベースへのアクセスコードを記述したモジュールであり、任意のクラスのエクステントに対する問合せや、オブジェクト間のナビゲーションなど、通常の PERCIO アプリケーションのコードと同様に記述する。データベースアクセスモジュールのコードは ActiveX コントロールにコンポーネント化され、データベースサーバ内でユーザからのアクセス要求を処理する実行スレッドにより動的にロードされる。

データベースアクセスモジュールは ActiveX コントロールであるため、そのコードを記述するためのプログラミング言語には任意のものが利用可能である。現在のところ PERCIO では C++、Java、SQL の各言語の API を持っており、これらの API を利用した ActiveX コントロールを作成すれば、本システムのデータベースアクセスモジュールで利用可能になる。また、指定されたクラスのエクステントに対してメンバ変数

の値により検索し、検索結果をHTML文書のテーブル形式に変換するというような典型的なものについては、あらかじめ汎用検索のActiveXコントロールを提供している。このデータベースアクセスモジュールを利用すれば、プログラミング言語によるコーディングが不要になる。

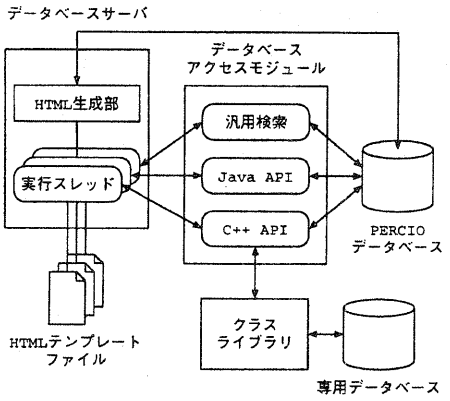

図 4: データベースアクセス

このデータベースアクセスの様子を図示すると図4のようになる。実行スレッドが定義ファイルに指定されたデータベースアクセスモジュールを動的にロードし、データベースにアクセスを行ってその結果を取得する。データベースアクセスモジュールは任意の言語APIを利用できるため、例えばPERCIO上にC++で作成されたクラスライブラリをC++ APIで記述したデータベースアクセスモジュールから利用するということもできる。その後、HTMLテンプレートファイルにしたがってHTML文書を生成するが、その際に再びデータベースへのアクセスが発生する。これはデータベースサーバのデータベースアクセス機能によるものである。

データベースへアクセスした結果は、PERCIOが管理するコレクションまたはオブジェクトへの参照という形式でデータベースアクセスモジュール内に取得される。これらのオブジェクト参照がHTMLテンプレートファイルで利用され、HTML文書生成に必要なデータが得られる。HTMLテンプレートファイルの記述では、データベースアクセスモジュールでどのような方法(プログラミング言語)が利用されたのかを意識することなく記述できるようにするために、データベースアクセス結果に対するインタフェースの共通化を図らなければならない。そこで、データベースアクセスモジュールにより取得したデータをHTMLテンプレートファイルの中で参照できるように、データベースアクセスモジュール内で公開する変数名(オブジェクト変数名)と変数値(オブジェクト変数値)を指定する(ActiveXコントロールのプロパティとして公開する)。この対応関係をデータベースサーバ内のテーブルとして管理し、本システムがHTMLテンプレートファイルを使ってHTML文書を生成する際に適宜参照するようにしている。

5 HTML文書生成

データベースアクセスモジュールによるデータベースアクセスを行った後は、HTMLテンプレートファイルにしたがってWWWブラウザに返却するためのHTML文書を生成する。HTMLテンプレートファイルは、IIS (Internet Information Server) [3]に搭載されているIDC (Internet Database Connectivity)のHTMLエクステンション(.htx)ファイルに基づいており、HTML文書の生成方法を制御するキーワードが規定されている。これに加えて、オブジェクト参照とコレクションを扱えるように仕様を拡張した。

HTMLテンプレートファイルで指定可能な変数には以下のものがある。

- **オブジェクト変数**

  データベースアクセスモジュールからエクスポートされたオブジェクトのための変数である。オブジェクト変数名を“<$”と“$>”で囲んで記述する。これにより、データベースアクセスモジュールで取得したデータベース内のオブジェクトを、HTMLテンプレートファイルで参照することができる。

- **パラメータ変数**

  WWWブラウザから渡されたパラメータのための変数である。オブジェクト変数と同様に“<$”と“$>”で囲んで記述する。さらに、変数名の先頭に“pdt.”を付けることによってそのパラメータにアクセスできるようになる。

- **HTTP変数**

  HTTPサーバに接続しているWWWブラウザや操作環境などの情報を保持している変数である。また、クライアントから送信される全てのヘッダが利用できる。HTMLテンプレートファイルでHTTP変数を利用する場合には、オブジェクト変数と同様に“<$”と“$>”を囲んで記述する。

- **組み込み変数**

  上記の変数以外にも、システム固有の変数を提供している。コレクション操作やセッション管理などに利用される。

HTMLテンプレートファイル内で、各オブジェクトのメンバ変数へのアクセスにはオブジェクト変数を利用する。オブジェクト変数はオブジェクトまたはコレクションへの参照なので、メンバ変数の値を取得するためにドット記法を利用する。オブジェクト変数にバインドされたオブジェクトのメンバにアクセスする場合に、

<$ (オブジェクト変数名).(メンバ変数名) $>

のように記述する。HTML 文書が生成されるときは、この拡張タグがデータベース中のデータ値に変換されて埋め込まれる。

オブジェクト変数がコレクションの場合には制御構文を用いる。書式は次のようである。

```
<$foreach (代入式)$>
<$do [(終了条件式)]$>
    (HTML テキスト)
<$done$>
```

(代入式) は "(variable) in (collection)" という形式で記述する。variable には任意の変数名を指定し、collection にはコレクションにバインドされたオブジェクト変数名を指定する。このコレクション内の各要素を 1 つずつ variable に指定された変数に割り当てていき、<do> と <$done$> で囲まれた範囲内で他のオブジェクト変数と同様に利用することができるようになる。この処理をコレクションの要素がなくなるまで繰り返す。<do> の中の (終了条件式) は "until (value)" の形式であり、繰り返し処理を行う最大回数を value に指定することも可能である。この foreach 文は任意にネストすることができ、深さ優先でそれらの複合オブジェクトへのアクセスに適用される。

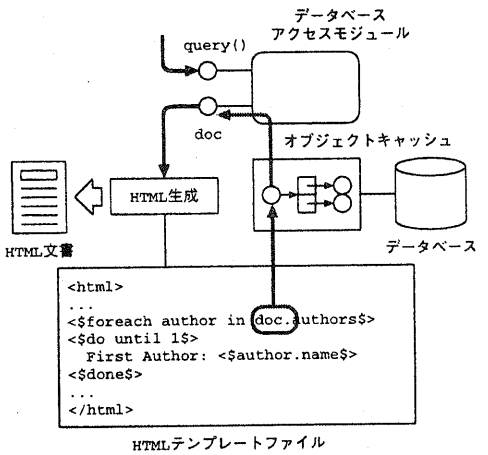

図 5: HTML 文書生成

この HTML 文書生成の内部処理の様子を図 5に示す。データベースアクセスモジュール内で必要な複合オブジェクトの基点となるオブジェクトを取得し、それを ActiveX コントロールのプロパティとしてエクスポートしてあるため、HTML 生成部はそのプロパティから基点となるオブジェクトを参照することができる。基点となるオブジェクトを取得するためには、そのオブジェクトのメンバ変数や下位のオブジェクトにも参照する場合があり、オブジェクトキャッシュ上に必要なオブジェクトが読み込まれていることが多い。その場合には、HTML テンプレートファイル内にドット記法によりオブジェクト参照があると、オブジェクトキャッシュ上の読み込まれたオブジェクトを辿ればよいので冗長なディスクアクセスが回避できる。

6 セッション管理

現状の HTTP の仕様はユーザがアクセスしているセッションの状態を保持可能なプロトコルになっていないため、他のシステムと同様に疑似的なセッション管理を実現している。疑似的なセッション管理を実現するためのセッション情報としては、ユーザ毎にシステムでユニークなセッション ID を割り当てており、これを WWW ブラウザと HTTP サーバで受け渡すことになる。ここで、セッションとはトランザクションのスタート / コミットやデータベースのオープン / クローズの上位に位置する概念である。WWW システムにおいては、クライアントとサーバの間で常にセッションを張っておくという状態を保持することが不可能であるため、タイムアウトによりセッションを切るための機構も必要である。その際には、トランザクションのアボートとデータベースのクローズの処理も自動的にシステム側で行われる。

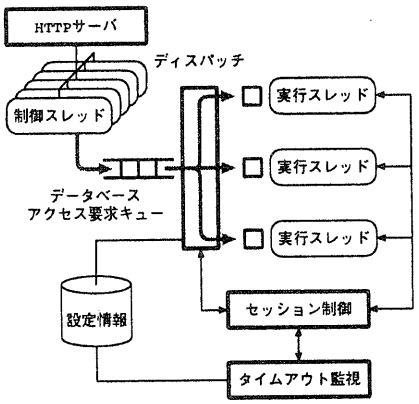

図 6: マルチスレッド制御

各セッションの処理はデータベースサーバ内で実行スレッドによって実行される。これらマルチスレッドの制御がどのように行われるかを図 6に示す。HTTP サーバは、WWW ブラウザからアクセス要求を受け付けた段階で制御スレッドを生成して処理を開始し、WWW ブラウザに HTML 文書を返却するとこの制御スレッドは終了する。制御スレッドは、現在データベースサーバ内に存在する実行スレッドが、設定情報に登録した最大スレッド数を越えていなければ、新しい実行スレッドを生成してそのスレッドに制御を移す。もし、最大スレッド数を越えていればそのアクセス要求

はデータベースアクセス要求キューに入れられ、実行スレッドが空くまで処理が待たされるが、設定情報に登録したセッションタイムアウト時間を過ぎると処理されずにエラー情報が返却される。

セッションを継続するためには、セッションそのものの情報以外にも、次のようなリソースの状態を保持しなければならない。

- HTML テンプレートファイル内で利用される変数名を保持するオブジェクト変数テーブル
- PERCIO カーネルが利用するオブジェクトキャッシュ
- 継続させるトランザクション
- 上記のトランザクションを実行中のスレッド

データベースアクセスモジュールでエクスポートされた変数名や関数名などはオブジェクト変数テーブルに登録されており、これをセッション中で共有することにより、WWW ページ間にまたがる処理を実現する。また、オブジェクトキャッシュやトランザクションの実行状態を保持するためには、HTML 文書を生成した後にもセッションを開始したスレッドとスレッド内のトランザクションを終了させずに残しておく。次のアクセス時には、このスレッドに処理を割り当てて同一トランザクションを保証する。

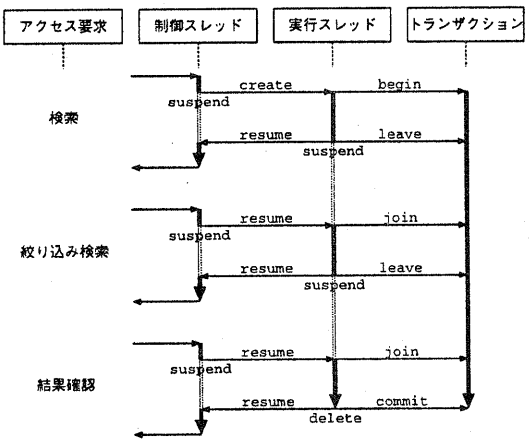

図 7: セッション継続

セッション継続における制御スレッドと実行スレッド、トランザクションの関係を図 7に示す。縦の矢印で、制御スレッドと実行スレッドの実線のところは動作中を表し、破線のところは中断中を表す。トランザクションは begin から commit まで常に実行中である。横の矢印は各スレッドからの命令を表している。この図の例では、1) 検索要求を出した後、2) 条件を追加して絞り込み検索を行い、3) 結果の確認を行っている。各アクセス要求に対する処理の流れは次のようになる。

1. HTTP サーバからデータベースアクセスを行うために制御スレッドが生成される。
2. 新規のセッションであれば新しく実行スレッドが生成されるが、継続のセッションであれば該当する実行スレッドを再開する。
3. 制御スレッドを中断して実行スレッドに制御を移す。
4. 新規のセッションであればトランザクションを生成してデータベースアクセスを行うが、継続するセッションであれば該当するトランザクションに接続する。
5. データベースアクセスが終了すると、セッションを継続する場合はトランザクションから切り離すのみであるが、セッションを継続しない場合はトランザクションをコミットする。
6. 制御スレッドを再開し、セッションを継続する場合は実行スレッドを中断するが、セッションを継続しない場合は実行スレッドを終了する。
7. 生成された HTML 文書を返却して制御スレッドが終了する。

7 性能評価

本システムのマルチクライアントにおける基本的な性能を調べるために性能評価を行った。性能比較を行う対象は、

- 本システムをサーバ API から利用する場合 (wapi)
- 本システムを CGI から利用する場合 (wcgi)
- CGI プログラムのみで PERCIO のデータベースにアクセスする場合 (cgi)

の 3 種類である。性能測定で利用した環境は、サーバマシンが NEC Express 5800/160Pro (CPU: Pentium Pro 200MHz × 2, メモリ: 128M バイト, OS: Windows NT Server 4.0)、サーバとクライアントの間はイーサネットで結ばれた LAN 環境であり、プロキシは経由していない。クライアントプログラムの構成は WWW システムのベンチマークである WebStone[4] を参考にして、WebMaster から複数の WebClient を生成し、全 WebClient のタイミングを取って同時アクセスを行うようにした。

性能評価のためのデータベースは、HTML 形式で記述された文献の管理のためのものを想定しており、文献 (1 万件) と著者 (1 千件)、文献の概要 (1 万件)、キーワード (5 百件) の各データを格納する。また、著者名をキーとして全ての著者データに対して一意になるような名前を付けてある。性能測定には、任意の著者名を与えてデータベースから検索し、該当する著者データの著者名と所属先を HTML 文書に変換して返却する、という演算を利用した。

1 クライアントでの測定結果を図 8に、3 クライアントでの測定結果を図 9に示す。これらは同じ処理を

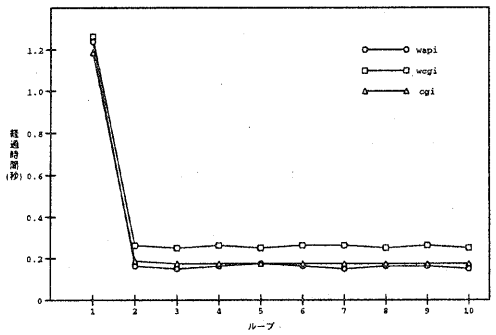

図 8: ループ回数による測定結果 (1 クライアント)

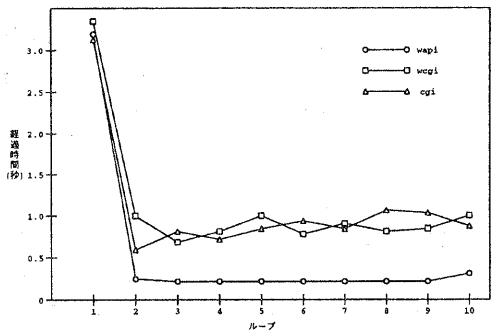

図 9: ループ回数による測定結果 (3 クライアント)

10 回ずつループで繰り返して測定したものである。図を見て明らかなように、どの方法も 1 回目のループより 2 回目以降のループが性能がよくなっているが、これは 1 回目のループで PERCIO のオブジェクトキャッシュ上に必要なデータが載り、データベースへの物理的なアクセスがほとんどなくなるためである。1 クライアントの場合はいずれも 2 回目以降のループで安定した性能が得られているが、3 クライアントの場合は wcgi と cgi は不安定になる。また、1 回目のループでどの方法もほぼ同等の性能になっており、wapi ではデータベースアクセスモジュールの動的リンクの処理が含まれていることと、ディスクアクセスにかなりの負荷がかかるために、性能差が出ないと思われる。

次に、クライアント数による測定結果を図 10に示す。図 8と図 9より 2 回目以降のループはそれほど性能差がないため、2 回目のループのみを取り上げた。いずれもクライアント数が増加するとほぼ線形に経過時間が増加しているが、wapi はあまり性能の劣化がないことがわかる。1 クライアントと 10 クライアントを比較すると、wapi が約 4 倍なのに対して、wcgi と cgi は 10 倍以上になっている。また、wcgi が cgi より性能が悪いのは、wcgi は HTML 文書生成処理などの通常の WWW-DB 連携システムの処理を含むためである。

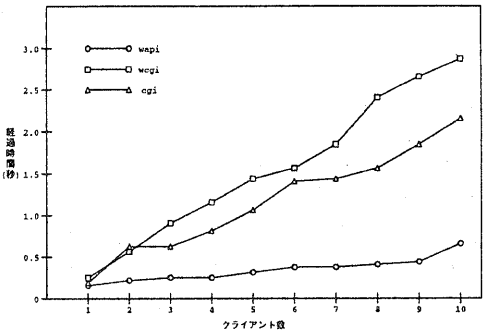

図 10: クライアント数による測定結果 (ループ 2)

8 まとめ

本稿では、オブジェクト指向データベース管理システム PERCIO を利用した WWW 連携システムの実装方法と性能評価について述べた。本システムは HTTP サーバが提供するサーバ API を利用して、HTTP サーバプロセス内にデータベース連携機能を持たせ、セッション管理とマルチスレッド制御が統合されている。性能評価の結果、マルチクライアントのアクセスにおいて、従来の CGI を利用した方式よりも数倍の高速化を実現した。これにより、イントラネットデータベースシステム [8] や WWW コンテンツ管理 [9] などの高度な WWW アプリケーションに応用することができる。

参考文献

[1] Chappell, D., *Understanding ActiveX and OLE*, Microsoft Press, 1996.

[2] Hadjiefthymiades, S.P. and Martakos, D.I., "Improving the performance of CGI compliant database gateways," *Proc. 6th World Wide Web Conference*, pp. 573–585, 1997.

[3] Microsoft Corp., *Internet Information Server*, `http://www.microsoft.com/japan/products/iis/`

[4] Mindcraft Inc., *WebStone*, `http://www.mindcraft.com/benchmarks/webstone/`

[5] Netscape Communications Corp., *Netscape Enterprise Server*, `http://home.netscape.com/ja/comprod/server_central/product/enterprise/`

[6] 日経データプロ, WWW- データベース連携システム構築法, 日経 BP, 1996.

[7] 鶴岡邦敏, 木村裕, 波内みさ, 安村義孝, "オブジェクト指向データベース管理システム PERCIO の開発と今後の課題," 電子情報通信学会論文誌, Vol. J79-D-I, No. 10, pp. 587–596, 1996.

[8] 安村義孝, "オブジェクト指向データベースによるイントラネットデータベースシステムの構築," 情報処理学会研究報告, Vol. 97, No. 64, 97-DBS-113, pp. 57–62, 1997.

[9] 安村義孝, "WWW-OODB 連携システムを用いたコンテンツ管理の実現," 情報処理学会第 55 回全国大会講演論文集 (3), 6X-3, 1997.