異種情報源統合利用環境における Web ラッパーの設計と開発

加藤 数則 †　森嶋 厚行 †　北川 博之 ††

† 筑波大学 工学研究科　†† 筑波大学 電子・情報工学系

概要

近年，コンピュータネットワークを通じた異種分散情報源の利用が容易になるにしたがい，それらの統合利用が重要な課題となっている．我々は，WWW，リレーショナルデータベース，構造化文書リポジトリを対象とした異種分散情報源統合利用環境の開発を行っている．本統合利用環境は，ラッパーと呼ぶソフトウエアモジュールを通じて各情報源の操作を行う．本稿では，WWW データを扱う Web ラッパーの設計と開発について述べる．Web ラッパー自身がすべての Web ページの処理を行うと，転送データ量をはじめとした転送コストが一般に膨大なものとなる．そこで，本稿で述べるアーキテクチャでは，ラッパー処理の一部を担当するオブジェクト群をコンピュータネットワーク上に分散配置することにより，Web ページの転送コストを削減し，問合せ処理の効率化を図る．

Design and Development of a Web Wrapper in Environments for Integration of Heterogeneous Information Sources

Kazunori Katoh†　Atsuyuki Morishima†　Hiroyuki Kitagawa††

†Doctoral Program in Engineering, University of Tsukuba
††Institute of Information Sciences and Electronics, University of Tsukuba

Abstract

Integration of heterogeneous information sources has been one of the most important issues in recent advanced application environments. We are developing an information integration environment for the World Wide Web, relational databases, and structured document repositories. In this environment, manipulation of the information sources are performed through software modules called wrappers. In this paper, we describe design and development of the World Wide Web (or Web) wrapper. In general, Web page manipulation may cause very large data transfer cost if all the necessary pages are transferred to the Web wrapper. The proposed Web wrapper architecture uses remote objects which cooperatively take part of the wrapper's functions at Web server sites, to reduce the cost of Web page transfer.

1　はじめに

近年，コンピュータネットワークを通じた異種分散情報源の利用が容易になるにしたがい，それらの統合利用が重要な課題となっている．特に，WWW の普及により，WWW とデータベースの統合利用への要求が高まっている．

我々は，WWW，リレーショナルデータベース，構造化文書リポジトリを対象とした統合利用環境の研究開発を行なっている [1][2][3]．そこでは，構造化文書リポジトリ中の文書として SGML 文書を，また，WWW ページ記述言語として XML[4] を想定している．XML は，現在広く利用されている HTML

とは異なり，ユーザが文書ごとに文書構造を定義することが可能である．図1に現在開発中の統合利用環境を示す．まず，ラッパーが各種情報源を，我々が開発した統合データモデル WebNR/SD [2] に変換し，メディエータがそれに基づいた統合操作の機構を提供する．問合せはメディエータに投入されると，各ラッパーで処理可能な部分問合せに分解される．メディエータは各ラッパーの処理結果から最終結果を作成し，ユーザに結果を返す．

WebNR/SD は，入れ子型リレーショナルデータモデルに XML や SGML 文書等の構造化文書を扱うための抽象データ型である「構造化文書型」（**SD** 型）を導入したものである．各構造化文書は SD 型の値 (**SD** 値) として扱われる．

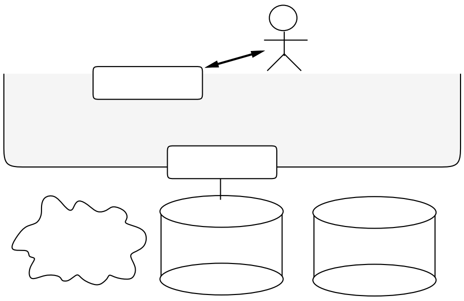

図 1. WebNR/SD を利用した統合利用環境

WebNR/SD では，通常の入れ子型リレーショナル代数演算子と文書検索関数群に加え，WebNR/SD 固有の演算子であるコンバータを用いたデータ操作が可能である．コンバータは，構造化文書と入れ子型リレーション構造の動的かつ部分的な相互変換を実現する．コンバータを利用することにより，以下のような操作が可能となる．(1) 入れ子型リレーショナル代数系を利用して，構造化文書の構造変換操作等を行なう．(2) 入れ子型リレーション構造を構造化文書に変換し，型と独立した検索やメタデータを手がかりとした検索等を行なう．(3) 異なる構造のデータを適当な抽象度で抽象化して統一的に操作する．さらに，WebNR/SD は，WWW データに対応するために，ページ間のリンク構造を扱うためのデータ型である Hlink 型を導入し，WWW のナビゲーション操作やリレーションを通じた WWW ページ群の操作などを実現している．これらの演算子群を組み合わせることにより，WWW，データベース，文書リポジトリに対しての総合的な問合せはもちろんのこと，これらの上にハイパーテキストビューを構築することなども可能である．

本稿では，本統合利用環境における Web ラッパーの設計と開発について述べる． Web ラッパーの問合せ処理においては，WWW 中のデータを参照する必要があるが，全ての WWW データの処理を Web ラッパーで行なうと，WWW ページの転送量は一般に膨大なものとなる．この転送量を削減し問合せ処理の効率化を図るために，本アーキテクチャでは分散オブジェクトを用いた問合せ処理を行なう．また，実際に分散オブジェクトを用いた実験環境でのデータ転送量，処理時間などを示す．

2 WebNR/SD

2.1 SD 型と Hlink 型

SD 型の値（SD 値）は，文書構造を表す DTD と，その DTD に従ったタグ付きテキストから構成される（図 2）．テキスト中で同じ名前のタグで区切られた部分を要素と呼ぶ．

```
report  = seq(title, authors, body, ref)
authors = rep(author)
body    = rep(chapter)
chapter = seq(chaptitle, rep(section))
section = seq(sectitle, rep(para))
    . . .
ref     = rep(refitem)
refitem = seq(title, authors)
    . . .
```

```
<report><title>Element Focusing</title>
<authors><author>...</author>...</authors>
<body><chapter><chaptitle>prolog</chaptitle>
                  . . . .
```

図 2. SD 値の例 (一部)

Hlink 型は SD 型の下位型であり，文書間のハイパーテキストリンク構造を表す． Hlink 型の値（Hlink 値）は，SD 値のうち hlink 構造を持つ要素ただ一つから構成される．hlink 構造を持つ要素は，属性 href を持ち，内部構造（副要素）を持たない．次は，Hlink 値の例である．

⟨ a:**hlink**, “`<a href="http://T.ac.jp/p1.xml">` `T-Univ </a>`” ⟩

2.2 コンバータ

コンバータは，SD 値と入れ子型リレーション構造を相互に変換する演算子である．プリミティブなコンバータとして，Unpack (**U**) と Pack (**P**) がある．Unpack は，SD 値中の要素群を含む副リレーション構造を作成する．Pack は，副リレーション構造を SD 値に変換する．次式は Unpack (**U**) と Pack (**P**) の

適用例である（図 3）.

$$r_2 \quad := \quad \mathbf{U}_{A \rightarrow B(O,C[b \subset c])\ as\ x}(r_1) \qquad (1)$$

$$r_1 \quad := \quad \mathbf{P}_{B(O,C)\ as\ x \rightarrow A}(r_2) \qquad (2)$$

ID	A
1	⟨a:**seq**(b,c:**rep**(b)),"<a>d1 <c>d2d3</c>"⟩

ID	A	B	
		O	C
1	⟨a:**seq**(b,c:**rep**(b)),"<a>d1 <c>&x.1;&x.2;</c>" ⟩	1	⟨b,"d2"⟩
		2	⟨b,"d3"⟩

図 3. コンバータ **U/P** の適用例 (r_1(上) と r_2(下))

2.3 WWW データの操作

ここでは, WWW データの操作を行なうための演算子について説明する.

Export/Import

Export (**E**) は, リレーション中の SD 値の内容を持つページを WWW 中に作成し, そのページを参照する Hlink 値を得る. 逆に, Import (**I**) は, リレーション中の Hlink 値が参照する WWW 中のページを SD 値としてリレーション中に取り込む. 以下に Export と Import の例とその実行例を示す.

$$r_3 \quad := \quad \mathbf{I}_{A,U,L,G}(r_4) \qquad (3)$$

$$r_4 \quad := \quad \mathbf{E}_{A,U,L,G}(r_3) \qquad (4)$$

ID	A	U	L	G
1	⟨ e:**seq**(f:**rep**(g),h), “<e><f><g>T4</g> </f><h>T5</h></e>”⟩	http://T.ac.jp/p.xml	T Univ	a

ID	A
1	⟨ a:**hlink**, “ T Univ ” ⟩

図 4. リレーション r_3(上) と r_4(下)

Navigate

Navigate (**N**) は, パラメータで指定されたパス正規表現（以後, パス式と呼ぶ）に基づいて WWW のリンクをたどり, 条件を満たす WWW ページ群を参照する Hlink 値を求めるものである.

$$r_6 \quad := \quad \mathbf{N}_{A \rightarrow .(\rightarrow .)* \Rightarrow B(\rightarrow .)* \rightarrow C[''paper''],D}(r_5) \qquad (5)$$

ここで, $A \rightarrow .(\rightarrow .)* \Rightarrow B(\rightarrow .)* \rightarrow C[''paper'']$ がパス式であり, D は, 新たな属性名である. また, A は r_5 の Hlink 型の属性名と一致していなければならない.

パス式の構文規則は図 5 のようになる.

```
path_reg_expr ::= label p_expr {'|' label p_expr}
p_expr        ::= link page [ '[' condition ']' ]
                  | p_expr { p_expr }
                  | p_expr { '|' p_expr }
                  | p_expr '*' ['/'num'..'num'/']
                  | '(' p_expr ')'
link          ::= '− >' | '=>'
page          ::= label | '.'
```

図 5. パス正規表現の構文規則

パス式では, 英文字列とピリオドがページを表し, → が同じ WWW サーバにあるページへのリンク (ローカルリンク) を, ⇒ は異なる WWW サーバにあるページへのリンク（グローバルリンク）を表す. * は 0 回以上の繰返しを表す. */num_A .. num_B/ は num_A 回以上 num_B 回以下の繰返しを表す. また，パス正規表現では，各ページの直後に “[ページの内容に関する選択条件]” を記述可能である．ハイパーテキストリンク構造が図 6 のようになっている時, 式 (5) の結果は r_6 となる. 新たに追加された属性 D は下位属性 B と C を持つ，副リレーション値を格納する．このリレーション値の各タプルには，属性 A 中の Hlink 値が参照するページから始まるリンク構造をたどる各パスのうち，パス式が受理可能なパスの B と C に対応するページを参照する Hlink 値が格納される．

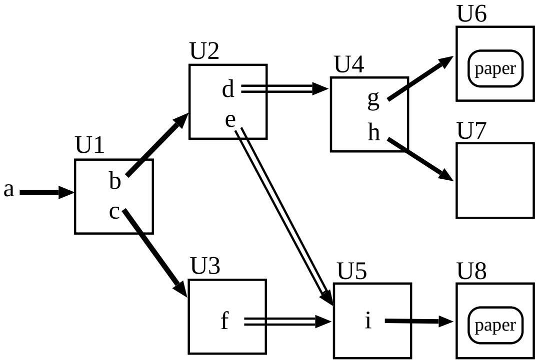

図 6. ハイパーテキストリンク構造の例 (Un は URL, a,b... はリンク要素内の文字列)

r_5:

ID	A
1	⟨ a:**hlink**, “a” ⟩

r_6:

ID	A	D	
		B	C
1	⟨ a:**hlink**, “ a” ⟩	⟨ a:**hlink**, “ d” ⟩	⟨ a:**hlink**, “ g” ⟩
		⟨ a:**hlink**, “ e” ⟩	⟨ a:**hlink**, “ i” ⟩
		⟨ a:**hlink**, “ f” ⟩	⟨ a:**hlink**, “ i” ⟩

図 7. リレーション r_5 （上）と r_6 （下）

URL

URL generator (**URL**) は，他とは重ならない新たな URL 群を作成する．これらは新たな WWW ページの作成に利用される．

3　Web ラッパー

3.1　機能概要

本統合利用環境では，Web ラッパーは，メディエータが生成された時にメディエータと同じ計算機上に一つだけ生成され，メディエータからの要求を受け付ける．

Web ラッパーは以下の機能を提供する．

1. 統合スキーマ上に WWW のリレーションビューの提供：本統合利用環境における統合スキーマでは，WWW は Hlink 型の属性を持つ単項リレーション群としてモデル化されている．このリレーションはその後で，Import，Navigate などの演算を行なうために必要となる．
2. Export，Import，Navigate，URL 演算の部分的処理の実行：メディエータからの要求に従い，これらの演算子を実行する．

3.2　分散オブジェクトを用いた処理効率化

以上に述べたような Web ラッパーに要求される機能を実現するときに考えられる最も単純なアーキテクチャは，Web ラッパーオブジェクト自身でそれらの処理を全て行なうものである．しかしこの方法では，ページの転送量が，一般に膨大なものとなる．これを解決するため，複数のオブジェクトを分散配置し，転送量の削減および処理の効率化を図る．以下では，Navigate の処理について説明する．

分散オブジェクトを用いた Navigate 処理のために Navigator オブジェクトを利用する．Navigator オブジェクトは，Web ラッパーがメディエータから Navigate 演算の処理要求を受けたときに生成され，処理を実行する．

Navigator オブジェクトは，Web ラッパーから Hlink 値とパス式を受けとり，そのパス式に基づくオートマトンを用いて処理を行なう．このオートマトンには → ，⇒ と "[condition]" で与えられたページの内容に関する選択の三種類の状態遷移がある．例として，パス式 $A \rightarrow .(\rightarrow .)* \Rightarrow B(\rightarrow .)* \rightarrow C[''paper'']$ により作成されたオートマトンを図 8 に示す．番号は名前が付けられていない（ここでは A,B,C 以外の）状態を区別するためのものである．Navigator オブジェクトは，与えられた Hlink 値が参照するページから始まり，オートマトンに受理されるパスの集合を Web ラッパーに返す．

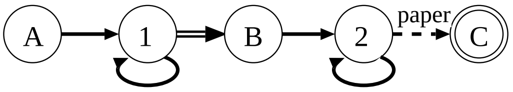

図 8. パス式
$A \rightarrow .(\rightarrow .)* \Rightarrow B(\rightarrow .)* \rightarrow C[''paper'']$
により作成されたオートマトン

複数の Navigator オブジェクトによる協調処理を実現するため，パス式より作成されたオートマトンを，同一の WWW サーバ内での処理と他の WWW サーバでの処理に分割する．分割の方法は，（1）初期状態から始まり，→ 遷移と選択遷移で遷移できる状態全てを含むように分割する．（2）それらの状態から ⇒ 遷移が可能である場合，そのリンク先を仮の受理状態として加える．（3）先ほどの仮の受理状態を初期状態とし，この処理を繰り返す．このとき初期状態が同じものは作らないようにする．これ以上の分割ができなくなれば分割を終了する．この結果，同一の WWW サーバ内での処理を表すいくつかのオートマトンに分割される．図 9 は，図 8 のオートマトンを，それぞれ同一の WWW サーバ内での処理を表す 2 つのオートマトン (X,Y) に分割したものである．

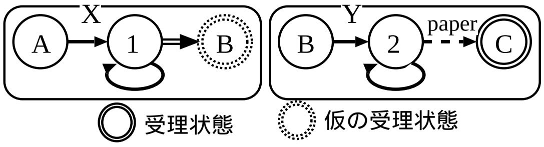

図 9. 分割されたオートマトン

分散処理を行なう際には，Navigator オブジェクトには，この分割されたオートマトンのいずれかが割り当てられる． そして一つの WWW サーバに対するこのオートマトンの処理を行なうものとする．仮の受理状態に達した時, Navigator オブジェクトを処理対象のページが存在する WWW サーバマシン上に生成し，その後の処理を行なわせる． 生成しようとするオブジェクトと同じオートマトンを持つものが既に存在する場合は，そのオブジェクトに接続し，起点のページを指す Hlink 値を渡し，その後の処理を行なわせる．ここで生成された Navigator オブジェクト が使用するオートマトンは，初期状態の識別子が，先ほどの仮の受理状態のものと同じになるものである． このように処理を続け，真の受理状態に達したパスが最終結果となる．

処理の流れを具体例に沿って説明する． パス式は $A \rightarrow .(\rightarrow .)* \Rightarrow B(\rightarrow .)* \rightarrow C[''paper'']$ とし，図 9 のオートマトンを使用する．起点となる Hlink 値は, a とする．WWW ページのリンク構造は図 10 のようになっているものと仮定する．この図で Unum は URL，α，β，γは WWW サーバ名，小文字のアルファベットは Hlink 値に変換されるリンク要素を表す．

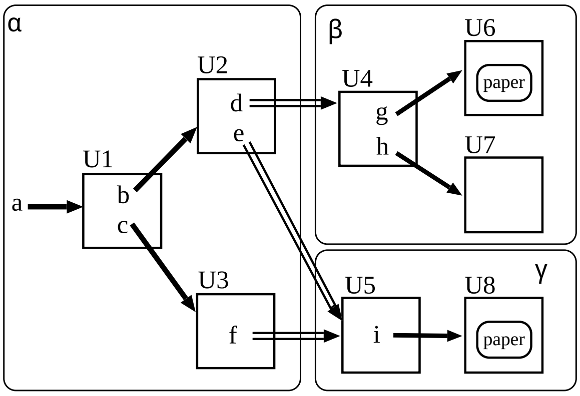

図 10. WWW ページのリンク構造例

図 11 に Navigator オブジェクトの生成，接続の流れを示す． Navigator オブジェクトは，円で示されており，その中の X,Y でそのオブジェクトが処理をするオートマトンを示した． 矢印に添えられた小文字のアルファベットは, Navigator オブジェクトの生成，接続のきっかけとなるリンク要素である．

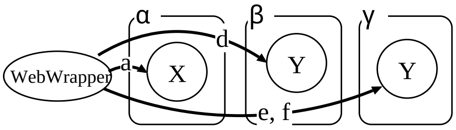

図 11.Navigator オブジェクトの生成，接続

Navigator オブジェクトは，受理状態ごとに処理結果を保持した表を作る．図 12 がその表である．例えばα_X_B は，WWW サーバα上でオートマトン X の処理を行なった結果，仮の受理状態 B に達したパスの集合である．また，属性 X-Y は，それが表の右端にある場合，これは仮の受理状態に遷移する時に，この属性中のリンクをたどったことを意味する． また，表の左端にある場合は，そのパスの直前に，この属性中のリンクをたどったことを意味する． これら二つを保持することにより，分割された処理結果を一つに結合することができる．

α_X_B

A	B	X-Y
a	d	d
a	e	e
a	f	f

β_Y_C

X-Y	C
d	g

γ_Y_C

X-Y	C
e	i
f	i

最終結果

A	B	C
a	d	g
a	e	i
a	f	i

図 12.Navigetor オブジェクトにより生成される表

4 実験

Navigate 演算処理において，検索対象の WWW ページをネットワークを介して転送し手元にもってきてから処理を行なう Web ラッパー単体による処理と，Navigator オブジェクトを分散配置することによって WWW ページの置かれているサーバマシン上で処理を行なってから処理結果だけを転送する場合とで，どれだけ処理時間の短縮やデータ転送量の削減が可能かを計測する． Web ラッパーおよび Navigator オブジェクトは Java オブジェクトして実装し，分散オブジェクト環境としては HORB[5][6] を使用した．また，今回の実験では，全ての処理を直列で行ない，並列処理は行なっていない．

4.1 実験方法

Navigator オブジェクトを分散配置した場合としない場合の，処理時間，データ転送量を計測しその違いを見る． この時，処理時間は Web ラッパーが Navigarte 処理を始めてから終るまでの時間を計測

し，データ転送量については，WWW サーバもしくはHORB デーモンから送られてきたパケットについて計測する．

実験対象の WWW ページのリンク構造は 3 段階の木構造をしており，末端のリーフページはリンクを持たない．ルートページの下に中間ノードページがあり各中間ノードページは 50 個リーフページへのリンクを有する． ルートページから中間ノードページへのリンクの個数により検索対象となるページ数を 50,100,200,400,800,1200,1600,2000, と変えている．

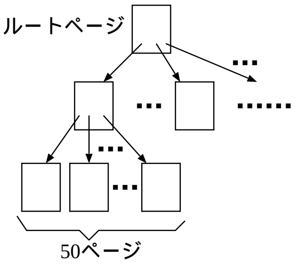

図 13 実験データ (WWW ページ) の構造

上記データに対する Navigate 処理を行なう際のパス式としては，以下のものを使用する．

$$A(\rightarrow .)* \rightarrow B$$

このパス式では起点のページ A からローカルリンクだけをたどり，到達できる全てのページが検索結果としてあげられる．

実験に使用した計算機の環境などを以下に示す．

Ultra 1 (筑波大)

- **CPU** Ultra Sparc 143MHz
- **メモリ** 64MB
- **OS** Solaris 2.6

Ultra 1 (神奈川工科大)

- **CPU** Ultra Sparc 143MHz
- **メモリ** 32MB
- **OS** Solaris 2.6

共通 Java JDK1.1.5

- **HORB** HORB 1.3.b1
- **WWW サーバ** Apache 1.2.6

実験には 2 台の計算機を使用し，サーバ，クライアントの関係を交換したものについても実験を行なった． ここでサーバとは WWW サーバが存在する計算機，クライアントとは Web ラッパーが存在する計算機のことである． このため，両方の計算機上に WWW サーバ，Java 実行環境，HORB を用意し，また Navigator クラスなどのクラスファイルや，実験データの XML 文書をあらかじめ配置する．

4.2 実験結果

4.2.1 データ転送量

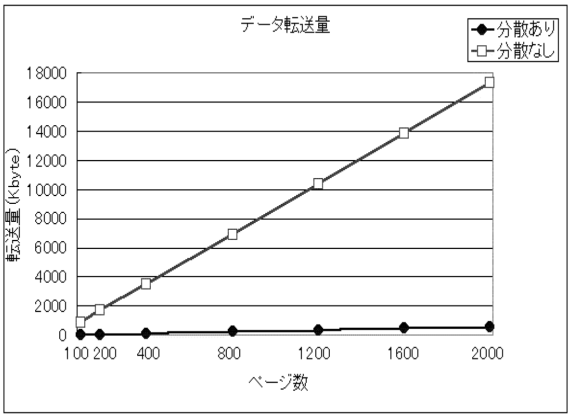

図 14. データ転送量

Navigator オブジェクトを分散配置することにより，データ転送量については大幅に削減できることが分かった．データ転送量は，分散配置した場合では，分散配置しない場合の約 3.4 ％になった．

4.2.2 処理時間

ネットワークのトラフィック状況の変化を考慮し，平日昼間と深夜の両者に対して測定を行なった．

平日昼間

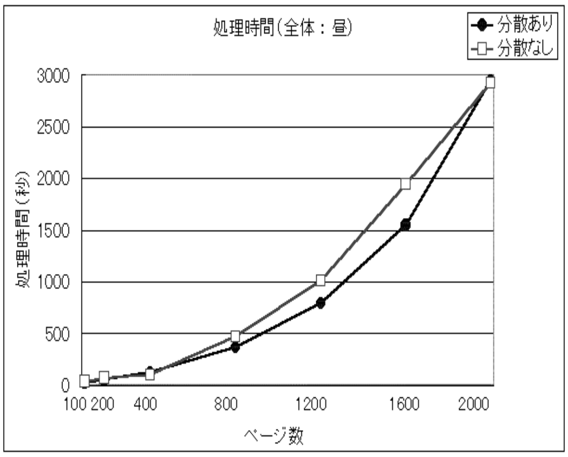

図 15. 処理時間 (平日昼間)

この実験では検索ページに関わらず，Navigator オブジェクトの分散配置を行なった方が早く処理を終えており，処理時間の短縮が計れている．処理時間の短縮率の平均は 19 %となった．これは，平日の昼間ということでネットワークの使用率が高かったためと考えられる．

深夜

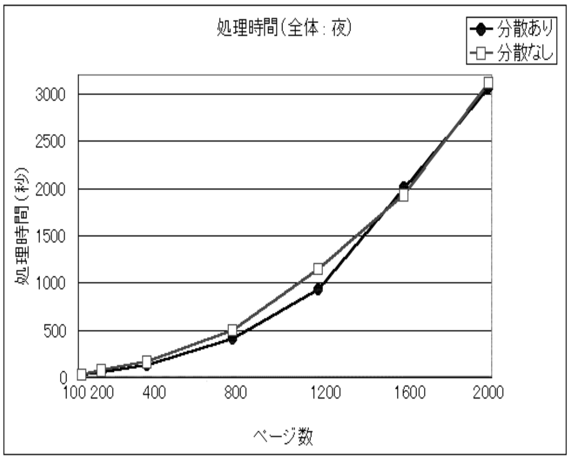

図 16. 処理時間 (深夜)

ここでは，短縮率の平均は 16 %であるが，オブジェクトを分散配置した場合しない場合で，処理時間がほぼ同じになる場合が多く現れた．これは，実験を行なった時刻が深夜でありネットワークが空いていたため，データ転送時間の差はそれほど大きくならず，オブジェクトを分散配置するためのオーバヘッドなどにより打ち消されてしまったことが原因であると思われる．

昼夜の平均

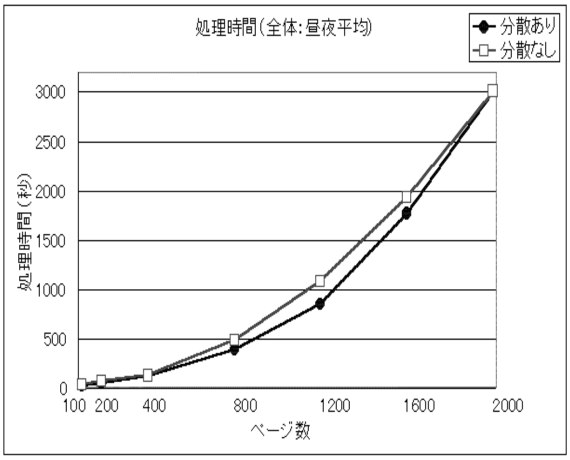

図 17. 処理時間 (昼夜平均)

図 15, 図 16 の平均をとったものである．短縮率の平均は 17 %となった．しかし，どの時間帯においても，検索ページ数が 2000 ページの場合では，処理時間がそれほど変わらなくなっている．

この要因としては以下が考えられる．すなわち，分散配置しない場合では，Navigator オブジェクトを Web ラッパーと同じ Java バーチャルマシン上に生成している．一方，分散配置を行なった方では，Navigator オブジェクトを HORB オブジェクトとして生成しているため，Web ラッパーと Navigator オブジェクト間での通信においてオーバヘッドが現れるのではないかと考えられる．また，WWW サーバと Navigator オブジェクトとが同じ計算機上にあることにより，お互いの処理がなんらかの干渉を起こしているのではないかということが考えられる．

このため，これらの要因を確認するため，以下のような 2 つの実験を行なった．

- WWW サーバの計算機と同じローカルネットワーク上の別の計算機上に Navigator オブジェクトを生成し処理させた場合．
- Web ラッパーの計算機と同じローカルネットワーク上の別の計算機上に Navigator オブジェクトを生成し処理させた場合．

これらの二つのケースでは，分散オブジェクト間での通信方式や，WWW サーバと Navigator オブジェクトとの処理の干渉の条件は同等である．結果を図 18 に示す．

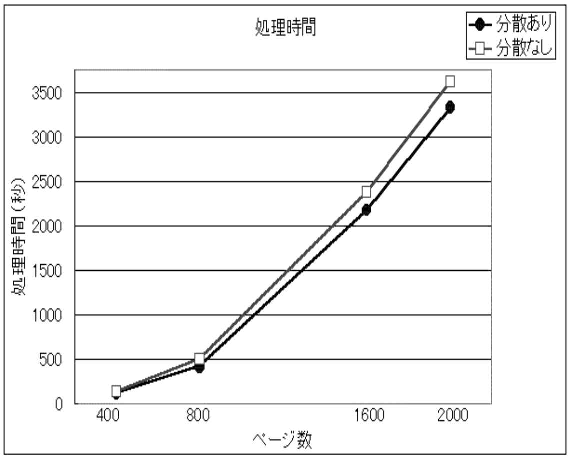

図 18. 処理時間 (通信方式などの環境が同等の場合)

この実験では，検索ページ数が増えるに従い処理時間の差は増加しており，検索ページ数が 2000 の時最大となっている．この結果により，分散配置を行なった場合では，検索ページ数が多くなると上に述

べた要因が処理時間に大きく影響することが分かった.

次に, 検索ページ数を10ページ以下と少なくした場合についての実験結果を示す.

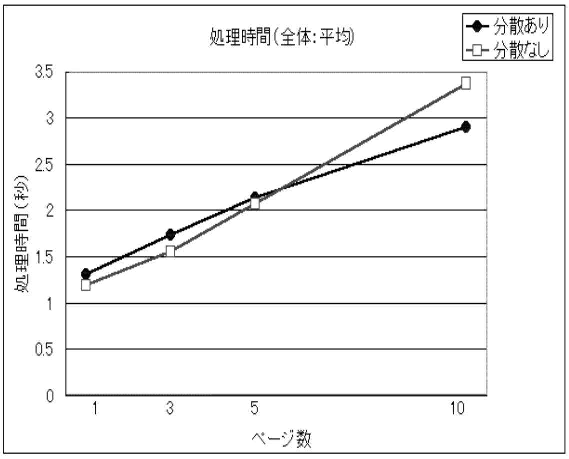

図19. 処理時間 (検索ページ数： 10ページ以下)

ここでは, 検索ページ数が5ページ以下の場合ではNavigatorオブジェクトを分散配置しない方が処理時間が短くなっている. これは, オブジェクトの分散配置にかかる時間は検索ページ数に関わらず一定であるため, 検索ページ数が少ない時には, ページ転送時間の削減がオブジェクトの分散配置による処理時間の増加よりも小さくなることによる.

このことにより, 処理時間を短縮するためには, パス式の長さや, ネットワークの混み具合などから分散配置をするかしないかを動的に判断する仕組みが必要であると考えられる.

5　まとめと今後の課題

WebNR/SDに基づいた異種情報源統合利用環境におけるWebラッパーの設計について述べ, 特に, 分散オブジェクトを用いたWWWページ転送量の削減方式を提案した. また, 分散オブジェクトを用いた処理の実験評価について述べた. WWWのナビゲーション操作は, WWWに対する問合せにおける本質的な操作の一つである[7][8]. したがって, 本稿の手法は, WebSQL[9]などのWWW問合せ言語の枠組にも適用可能と考えられる.

今後の課題としては, 下記がある.

- Navigatorオブジェクト内での処理アルゴリズムの効率化
- 複数のWWWサーバを検索対象とし, 複数のNavigatorオブジェクトによる分散並行処理をおこなった場合の評価
- Navigatorオブジェクトの効果的な配置方式の検討
- パス式評価順序の最適化

謝辞

評価実験にあたり御協力頂いた神奈川工科大学助手鈴木孝幸氏に深謝いたします. 本研究の一部は, 文部省科学研究費補助金特定領域研究「高度データベース」(08244101), 基盤研究(C)(09680321), 電気通信普及財団, ならびに筑波大学「東西言語文化の類型論」特別プロジェクトの助成による.

参考文献

[1] 加藤数則, 森嶋厚行, 北川博之 “WebNR/SD異種情報源統合利用環境の研究 – WWWに対する問合せ処理機構の設計と開発,” 情報処理学会第56回全国大会講演論文集, 1998 3月, pp. 252-253.

[2] A. Morishima and H. Kitagawa, ”Integrated Querying and Restructuring of the World Wide Web and Databases,” *Proc. International Symposium on Digital Media Information Dase (DMIB’97)*, Nov. 1997.

[3] 森嶋厚行，北川博之, “構造化文書とデータベースの統合利用のためのデータモデルNR/SD+とその問合せ処理,” 情報処理学会論文誌, Vol. 39, No. 4, 1998年4月, pp. 954-967.

[4] W3C, “Extensible Markup Language (XML) Version 1.0,” World Wide Web Consortium Recommendation, `http://www.w3.org/TR/REC-xml`.

[5] S. Hirano, “HORB Home Page,” `http://ring.etl.go.jp/ openlab/horb-j/WELCOME.HTM`.

[6] S. Hirano, “HORB: Distributed Execution of Java Programs,” *Worldwide Computing and It’s Applications ’97 (WWCA’97)* Tsukuba, March 1997, pp. 29-42.

[7] S. Abiteboul and V. Vianu, “Queries and Computation on the Web,” *Proc. 6th International Conference on Data Theory (ICDT’97)*, 1997, pp 262-275.

[8] A. O. Mendelzon and T. Milo, “Formal Models of Web Queries,” *Proc. 16th ACM Symposium on Principles of Database Systems (PODS’97)*, 1997, pp. 134-143.

[9] A. O. Mendelzon, G. A. Mihaila, and T. Milo, “Querying the World Wide Web,” *Proc. Symposium on Parallel and Distributed Information Systems (PDIS’96)*, December 1996, pp. 80-91.