Title	分散コンピューティング環境上のWebリンク収集システムの実装
Author(s)	伊藤，正敬
Citation	
Issue Date	2002-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/362
Rights	
Description	Supervisor:林 幸雄，知識科学研究科，修士

修　士　論　文

分散コンピューティング環境上の Web リンク収集システムの実装

指導教官　　林　幸雄　助教授

北陸先端科学技術大学院大学
知識科学研究科　知識システム基礎学専攻

050011　伊藤　正敬

審査委員：　林　　幸雄　助教授（主査）
中森　義輝　教授
藤波　努　　助教授
佐藤　賢二　助教授

2002 年２月

目　次

第 1 章
は じ め に

1.1　研究の背景と動機

1.1.1　WWW の特徴

近年，インターネットの急速な発展に伴い，WWW 情報リソースは日々増大し続けている．現代社会では，WWW は一つのメディアとして社会や個人に大きな影響を及ぼすものになった．WWW 上には生活に役立つ知識や技術的なヒント，個人に関わる様々な情報が多く記述されている．

WWW の特徴として挙げられる点は，まず非常にページ数が多いという**大規模さ**である．（Web ページは，Inktomi と NEC Research Institute の発表[1] によると，2000 年の時点で 10 億ページに及んでいる．）また内容や更新・削除などの**変化が速く**，映像や音楽といったメディアコンテンツまでも含んでいるので**多様性**も特徴に挙げられる．そして，現在多くの WWW を形成する HTML は記述形式が決まっており，表示記述にはある程度ルールがあるという点で**構造的**でもある．

このような特徴を持った WWW という情報リソースから，いかに有用な情報を探し当てるかについて，現在も様々な研究が行われている[2].

1.1.2　Web 検索の現状

現在，多くの WWW を記述している HTML は情報内容までは構造化していない．つい最近まではほとんどの検索エンジンがロボットによる文字列処理によって Web ページの解析を行っていたが，自然言語は数多くの同義語・多義語が存在するため，

有用なページを発見することは難しい．　また，キーワード検索では要求表現が難しく，検索対象が絞り込めなかったり，関連性がわからなかったりすることが多い．

一方，関連性のあるページを探す手法として，WWW 特有のハイパーリンク構造に着目した検索サービスが盛んになってきている．代表的な手法としては検索エンジン Google の Page Rank[3]が上げられるだろう．また最近では，検索エンジン Lycos に採用された WiseNut[4]という検索サービスもある．Page Rank，WiseNut はともに WWW のハイパーリンク構造に着目した検索技術, システムである. その他, Yahoo!, Excite, BIGLOBE といった検索エンジンもバックエンドでは Google が動いている．

ハイパーリンクの構造解析に着目した仕組みが多く登場し，採用されていることはリンク構造解析が有用な情報を導き出す一つの手法として有効であることを示していると考えられる．

1.1.3　WWW ロボットと分散コンピューティング

Web ページやハイパーリンクは主に WWW ロボットを用いて収集する．WWW ロボットとは, ページ収集の開始点となる URL から http プロトコルを用いてリンクを辿り，ページを収集するプログラムである．

但し，どんな高性能コンピュータでも，1 台では WWW 全てを解析処理できない．Google のような商用のシステムでは一般的には大規模分散型システムを利用して負荷分散処理を行いながら，膨大な Web ページを処理していく．しかし分散コンピューティングでも，大規模で変化が速く，多様な性質を持つ WWW 全体を解析するのは難しいとされている．自分たちの利用目的にそって WWW 情報を収集するのが妥当であろう．最近では，有用なページにいかに素早く辿り着くか，という点が注目されており，1.1.2 で述べた Google や WiseNut 以外にも様々な仕組みが研究，開発されている．中・小規模で利用目的に沿った WWW 情報収集ならば，近年の PC やネットワークの高速化，IT の急速な進歩によってそれほど大規模な設備を必要としなくても，分散環境下での情報収集システムを実現できると考えられる．

しかし実際には, WWW 情報収集を対象とした分散環境下での情報収集システムの事例はあるものの，その構築方法や技術的な問題点などの詳細を公開している研究はほとんど見当たらない．そこで，本研究では実際に Web リンク収集システムを構築し，その構築過程で得られた技術的問題点などを明らかにし，得られた知見をまとめたいと考えた.

1.2　研究の目的

本研究では，分散オブジェクト技術を用いて分散コンピューティングを実現し，並列で WWW のハイパーリンク収集を行うシステムの構築を行った.

そこで，本論文では分散コンピューティングによる WWW ハイパーリンク収集システムの構築方法，構築実現のための使用技術の概要，構築過程での問題点を明らかにする．また構築したシステムについて実験を行い，システムのパフォーマンスなどについても述べたい.

1.3　研究の方法

本論文では，まず WWW リンク収集システムについて，求めるべき仕様や機能について述べ，それら仕様を実現する技術概要について述べる.

そして，システム全体の処理フローについて述べ，定義した仕様が満たされているかについて検討する.

またシステムの性能を評価する実験を行い，システムパフォーマンスなどについて述べたい.

そしてシステムを構築する過程で得られた技術的問題点や知見をまとめる.

第 2 章
システム構成について

本章では，まず Web リンク収集システムとしての必要な仕様や機能について，いくつかの部分システムごとに分けて述べる．また，その仕様・機能に基づいて使用を決めた技術要素についても述べる．

技術要素については，まず使用した PC の構成概要について述べる．そして，分散環境を実現する分散オブジェクトについて概要と，本システム構築で使用した分散オブジェクト HORB について述べる．また，システムを開発するプログラミング言語 Java についての概要や利用する上での利点なども述べる．またデータベースを利用することから，データベースの概要と開発言語との接続方法について順に述べていく．

2.1　Web リンク収集システムの仕様・機能

2.1.1　Web リンクの収集

本システムの核となる処理は，クライアント PC から WWW サーバに接続して URL 先の Web ページを読み込み，その Web ページを解析してハイパーリンクを抜き取る作業である．この一連の作業を行うためには，インターネット，HTML 解析によりよく対応した開発言語を用いる必要がある．また，システムが複雑になれば，複数処理を並列で行う必要がでてくると考えたため，並列処理可能であることも開発言語選択の一つの要因に挙げた．

2.1.2　分散環境の構築

どんなに高性能のコンピュータでも 1 台では収集できるリンクの数は限られてくる．そこで，ネットワーク上に分散システムを構築し，複数台 PC によるリンク獲得を考えた．また逆に分散システムによる処理が可能ならば，それほど高性能な PC を用意する必要はないことも利点と考えられる．分散処理では，Web ページにアクセスして解析する複数台 PC と，それら複数台の PC を管理，監視する必要があり，負荷分散の仕組みを取り入れることとした．そこで本システムでは， Web リンクを直接収集する PC を 10 台用意し，その 10 台のタスク管理を行う PC を 1 台用意した．この PC 構成による負荷分散の仕組みを Master－Slave 方式とした．以後，Web リンクを収集する PC を Slave PC とし, Slave PC のタスク管理などを行う PC を Master PC とする．

2.1.3　データベースとの連携

また WWW は非常に大規模であり，ハイパーリンクを収集するとなるとさらに膨大なデータ量を扱わなければならない．大容量データを取り扱うためにはデータベース（以下，DB とする）を構築する必要があると考えた．但し，使用する開発言語に対応した DB を選択しないと，プログラムと DB の接続時にエラーなどが出てしまう可能性があるので，DB 選択には注意すべきである．また本システムは複数台 Slave PC から獲得した URL データが送られてくる可能性があるので，トランザクション処理を備えた DB が望ましい． その他，DB は仕様によっては DB 操作が独自の言語によってでしか操作できないものもある．DB の利用のしやすさなどを考え，DB の選択時に SQL に準拠したものを選択することとした．

以上で述べたシステムの仕様に基づく技術要素ついて，以下にポイントをまとめる．

（1）使用する PC について

- 分散システムを構築するために複数台用意できる
- それほど高性能でなくても良い

（2）分散システム技術について

- 複数 PC が扱え，並列処理を可能にする仕組みが備わっている

（3）開発言語について

・ インターネット，HTML 解析に対応している
・ 並列処理可能である
・ 分散システムが構築可能である

（4）DB について

・ 容量制限が大きい
・ 開発言語との相性が良い
・ トランザクション処理を備えている
・ SQL に準拠している

以上の仕様に基づき，システムを構成する要素について述べたい.

2.2　使用 PC の概要

本システムは Slave PC を 10 台，Master PC を 1 台，DB サーバを 1 台，計１２台用意した.

本章では構成する PC のハードスペックや OS などについての概要を説明する.

2.2.1　使用 PC のハードスペック

本研究で使用した PC の基本仕様を下記表 2.1 に記す．但し，MasterPC はメモリを１９８M，DB サーバ（DB サーバ）は HDD を４０G に増設した.

マシン名	HITACHI FLORA370
CPU	Pentium2－400MHz
メモリ	128M
LAN	10BASE-T
HDD	6G

表 2.1．使用 PC の基本仕様

2.2.2　OS

PC にインストールした OS は全て VineLinux2.1.5 を使用した.

VineLinux は Version2.1.5 からパッケージのバージョンアップやパッチ修正を apt によって行っている．本システムで使用した PC もインストール後 apt によってパッケージの更新を行った.

また，参考として，使用 PC の写真画像を載せる.

図 2.1．使用 PC の写真

2.3　分散システム技術

分散環境を実現する手法や技術は複数存在するが，ネットワークコンピューティングに伴う複雑さをより軽減する技術の一つに分散オブジェクトがある．分散オブジェクトは関数のパラメータや戻り値に複雑な構造を渡すときに，その複雑さを隠蔽するためにオブジェクトを利用するものである[5].

そこで，本システムの分散環境を構築する技術として分散オブジェクトを用いることとした．以下に，分散オブジェクトの概要，使用した分散オブジェクト技術の詳細などについて記述した.

2.3.1　分散オブジェクトの概要

ネットワーク上のどこにでも存在することができるオブジェクトのことを分散オブジェクトという．分散オブジェクトは独立して存在するコードであり，遠隔のクライアントからはメソッド呼び出しによってアクセスされる．クライアントは，分散されたオブジェクトがどこに位置にするのか，あるいは実行されるプログラムがどの言語で記述されているのかを知る必要は無い．分散されたオブジェクト間の通信は透過的に実行される[5][8].

クライアント・オブジェクトがサーバ・オブジェクトに対して行う処理は，クライアント側のサーバ・オブジェクトの代理オブジェクトに対してメソッド呼び出しを命令する．この代理オブジェクトのことを Proxy オブジェクトという．そして，サーバのメソッドが呼び出された結果を Proxy オブジェクトに返却する．また，サーバ側にも Proxy オブジェクトと同様な働きをする Skeleton オブジェクトというオブジェクトが存在する.

また，一般的に分散オブジェクトは様々な言語が利用されるため，開発においてオブジェクトを他のプログラムから利用するためのインターフェースを記述するのに使われる言語が必要とされる．そのオブジェクトが備えるメソッドやプロパティなどの情報を定義するのに使う言語を Interface Definition Language（以下，IDL とする）という.

2.3.2　HORB の概要

本システムは，分散オブジェクト技術の一つである HORB によって分散コンピューティングを実現している．HORB は，独立行政法人　産業技術総合研究所の平野博士が開発した, オープンソースの Java 用分散オブジェクトアーキテクチャである．

HORB は, ネットワークコンピューティングに関わる様々なコストを低く押さえ, かつ容易に実現することができることを目指して開発されたとされている[6].

分散ネットワークプログラミングはポータビリティ（移植性）とインタオペラビリティ（相互運用性）が重要な要因として挙げられる．（ポータビリティは，そのプログラムが様々な種類のマシンで実行できるという意味である.）HORB はこの 2 つの要素を達成した分散言語システムの一つである．

HORB は Java で開発された分散オブジェクトであり，Sun Java と１００％の互換性がある．そのため，あらゆる Java 処理系で動作可能であり，Java の利点の一つであるオープンプラットフォームでもある．また IDL の記述は不要となっている．

2.3.3　HORB アーキテクチャ

HORB は, HORBC コンパイラ, HORB サーバ（Object Request Broker の一つ), HORB クラスライブラリで構成されている[6]．HORBC コンパイラでコンパイルしてできた Java オブジェクトは分散環境で使える．HORB は Sun の提供する Javac コンパイラ，Java インタプリタ，Java システムクラスとともに動作する．Sun のプログラム・ソースコードには，HORB ツールキットのためにいかなる変更も加えられていない．それは HORB のポータビリティを保つためと，Sun のソースコードライセンスに束縛されないためである．つまり HORB はプラットフォーム固有のネイティブメソッドを使っておらず，マシンアーキテクチャ独立である[6].

2.3.4　HORB による基本的なサーバ・クライアントシステム

HORB も基本的な仕組みは一般的な分散オブジェクトと同じである．

ここにサーバとクライアントマシンがあると仮定し，サーバクラスとクライアントクラスを置くとする．プログラムを開始させると，まずクライアント・オブジェクトがサーバ・システム内にサーバ・オブジェクトを生成する．次にサーバ・オブジェクトのメソッドを呼び出す．２つのオブジェクト間のメソッド呼び出しをシームレスで透過（トランスペアレント）にするために Proxy と Skeleton，および ORB が用いられる．

Proxy，Skeleton オブジェクトは HORBC コンパイラによって生成されるので，ユーザはクライアントとサーバ･オブジェクトを記述して，サーバ側で ORB を起動し，クライアント側のプログラムを動作させるだけでよい．

以下に HORB の全体像を図 2.2 に示す．

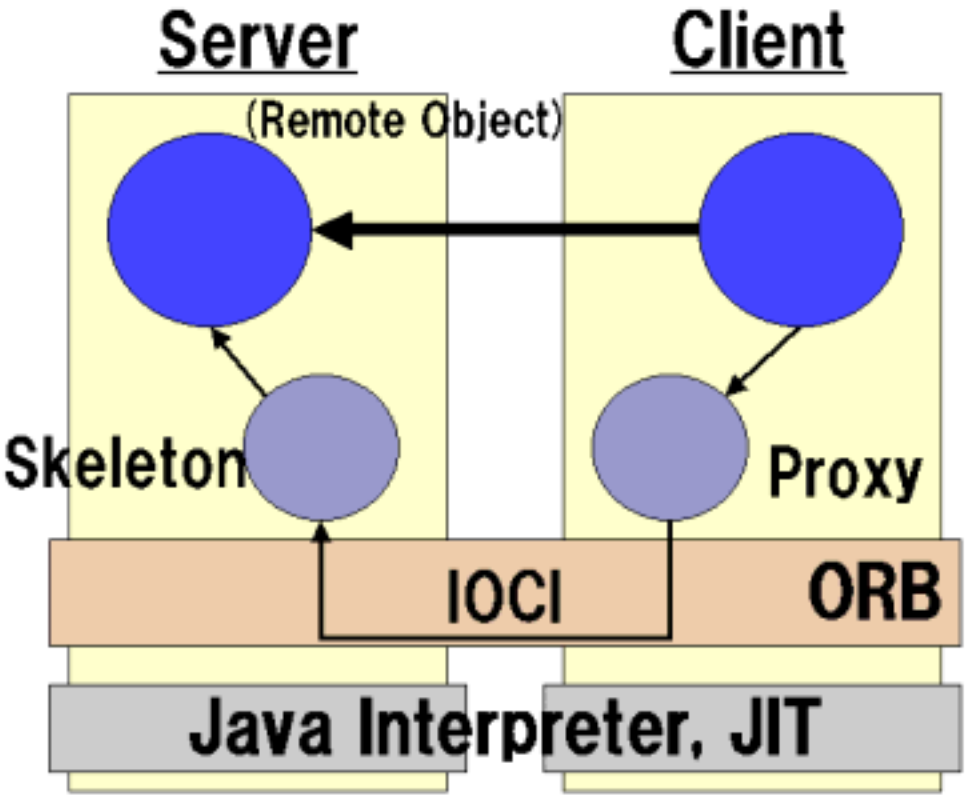

図 2.2．HORB の全体像

2.3.5　CORBA，RMI との比較

HORB と HORB 以外の分散オブジェクトとの比較について述べる[7].

まず RMI と比較すると，HORB は RMI の 2 倍の実行処理速度を備えている．また RMI は並列処理プログラミングを可能とする非同期メソッド機能がサポートされていない.

次に CORBA と比較した場合は，HORB は全て Java で記述するため IDL を必要とせず，システム開発作業が軽減される．また非同期メソッドについては，CORBA の仕様は複雑であるため容易に並列処理プログラミングができない．また，HORB は CORBA の IIOP をサポートしている.

これらの比較について表 2.2 にまとめて記述する.

分散オブジェクト技術	比較したときの HORB の利点
RMI	実行速度が 2 倍
	非同期メソッドをサポート
CORBA	IDL の記述不要
	非同期メソッドの記述が容易
	CORBA　IIOP をサポート

表 2.2．HORB と RMI，CORBA の機能比較

2.3.6　HORB プログラミング TIPS

以下では，本システムを構築するにあたり，HORB プログラミングの中でも特に重要な技術について簡潔に述べる[9][10][11][12].

（1）Server オブジェクトの呼び出し方

以下に簡単な String オブジェクトをやり取りする，サーバ・クライアントプログラムを記述する.

［Server. Java］

```
1 : public class Server{
2 :     public String message(String name){
3 :             return “Hello, “ +name+ “!”;
4 :     }
5 : }
```

［Client. Java］

```
1 : class Client{
2 :     public static void main(String args[]){
3 :             String host = “***”;
4 :             Server_Proxy server = new Server_Proxy(“horb://”+host+”/”;
5 :             String result = server.message(“World”);
6 :             System.out.println(result);
7 :     }
8 : }
```

Client.java の４行目の ServerNAME_Proxy で Proxy オブジェクトを生成している．これにより，Client.java の 5 行目のように一般的なメソッドの呼び出しと同じように扱うことができる.

ここで，Client.java の 4 行目の「Proxy オブジェクトの生成」について流れ図を用いて説明する．

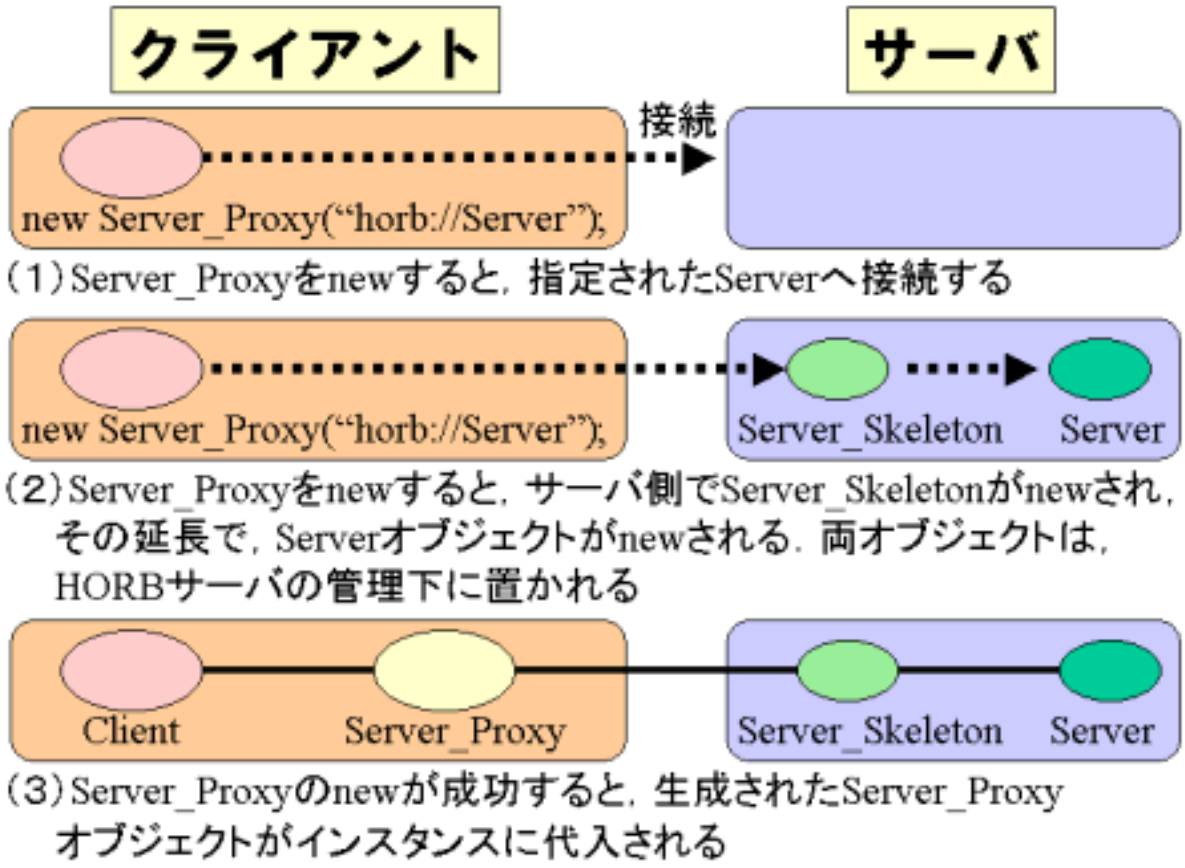

図 2.3. Proxy オブジェクトの生成

また，Client.java の 5 行目にある「message メソッドのリモート呼び出し」についても流れ図を示す.

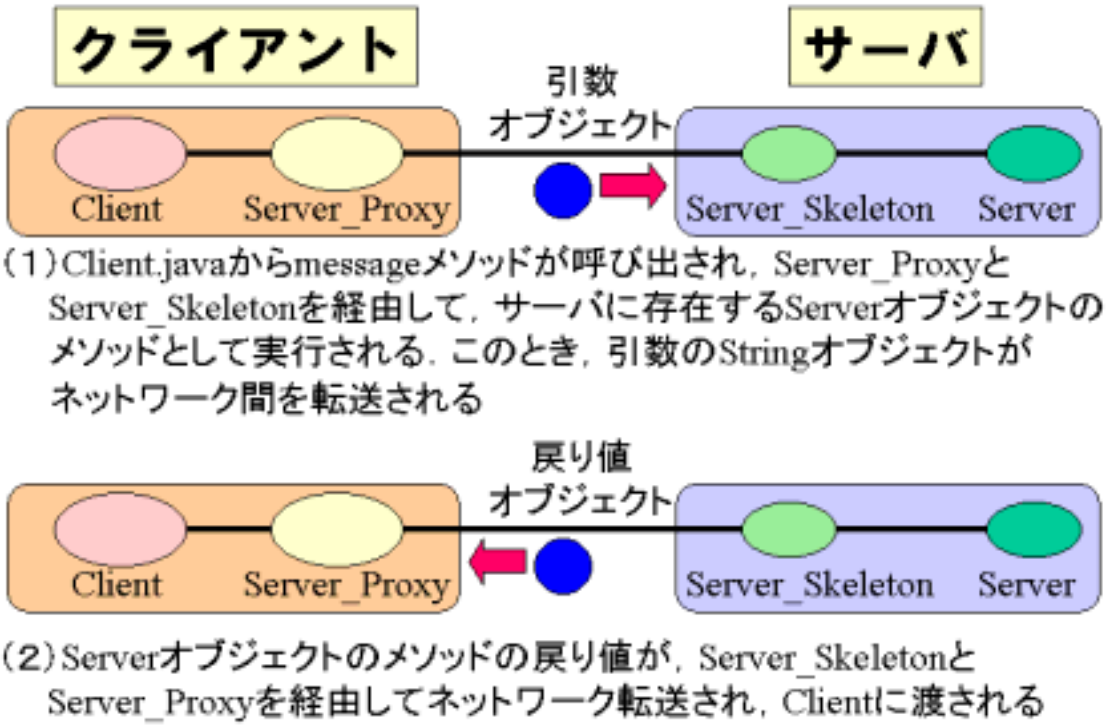

図 2.4. サーバ・メソッドのリモート呼び出し

（2）ネットワーク環境で動かすための条件

ネットワーク環境で動かすためにはクラスファイルの配置など，いくつか行わなければならない条件がある．その条件について，上記（1）のプログラム例を用いて述べる．

① クライアントとサーバのクラスの配置

クライアントとサーバにそれぞれ配置するクラスファイルを下記表にまとめた．

クライアント側に配置するクラスファイル	Client.class
	Server_Proxy.class
サーバ側に配置するクラスファイル	Server_Skelton.class
	Server.class

表 2.3．クライアントとサーバへのクラスの配置

② サーバマシンのホスト名の確認

サーバ名は DNS に登録されているマシンの名前を記述する．記述方法は様々だが，ソース例でいえば，Client.java の３行目の***の部分である．

2.3.7　非同期メソッド

非同期メソッドは並列処理を行うために必要な技術である．非同期メソッドとは，リモートオブジェクトのメソッドを呼び出し，そのメソッドの処理が完了する前に，呼び出し側でも他の処理ができるというものである．これによりシステム全体のスループットが向上する．Java では，この処理の仕組みをマルチスレッドで行う．

HORB では２つの非同期メソッドがある．

（１）Oneway 呼び出し

Oneway メソッドは戻り値を必要としない非同期メソッドである．

OneWay 方式の非同期メソッドを呼び出すには, Client 側のリモート呼び出し命令文に_OneWay()を記述し，Server 側のメソッドにも_OneWay()を記述するだけである．

（２）Async メソッド

Async メソッドは戻り値を必要とする非同期メソッドである．

Async 非同期メソッドをコンパイルすると Proxy オブジェクトに，Request と Receieve メソッドが自動付加される．また，非同期メソッドの戻り値は ResultAsync という Future オブジェクトに格納される．但し，ResultAync オブジェクトに対応する非同期メソッドの処理がサーバ側でまだ完了していないときは, _Receive メソッドは，その完了をまたされる．

しかし，Async メソッドにはサーバ側の処理が完了したかを確認（ポーリング）するメソッドがある. ポーリングするには isAvailable メソッドを使用する. isAvailable メソッドは, true が返されるとサーバ側の非同期メソッドが完了していることを表す.

Async メソッドは処理を同時並行的に進めるという意味では，人間が複数人で仕事を行うモデルと対比して考えられる．そこで，負荷分散のための非同期メソッドを説明するために，荻本順三氏の Web ページ上で提案されている「親分，子分モデル」[5] について述べる. 以下, 文章などを Web ページから引用, 要約した形で記述する. また図についても著者 Web ページから引用する．

（a） 忍耐強い親分モデル（ポーリングしない方法）

親分は子分に仕事を渡した後，自分の作業をする．そのあと適当な時間になったら子分のところへ順次作業結果を取りに行く．このモデルでは，子分の中に一人でも仕事を終えていないものがいる場合，親分は待たされてしまうことになる．

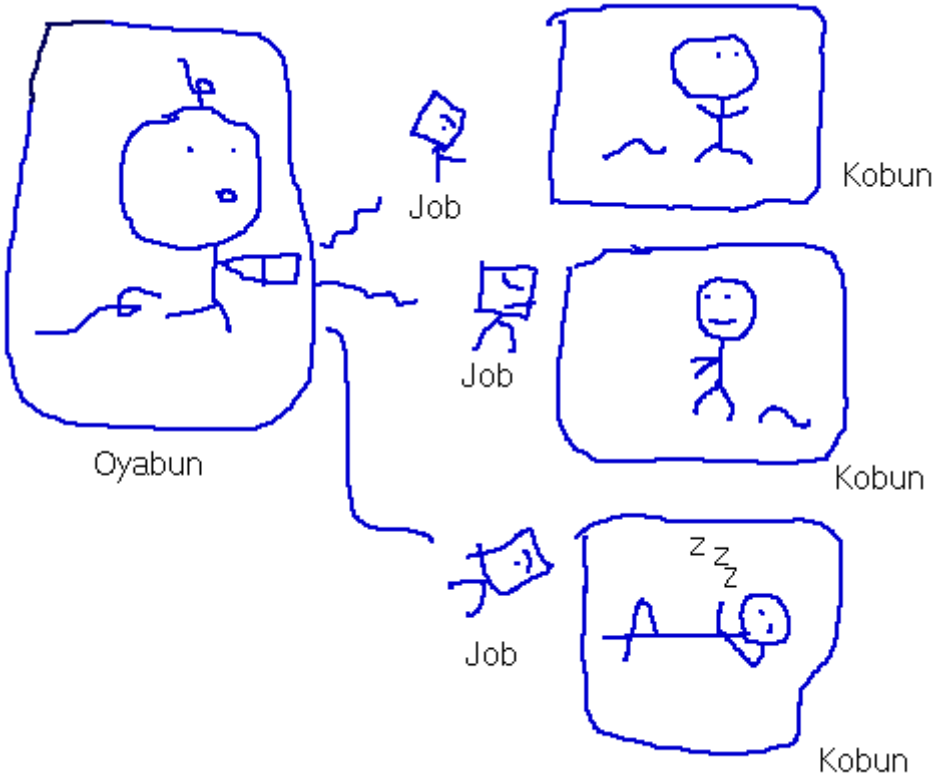

図 2.5．基本の親分子分モデル

（b） 利口になった親分モデル

上記（a）の問題を解決するには，子分に電話（ポーリング）をかけて作業が終わったかを確認すればよい．もし終わっていれば親分が仕事を取りに行く．

この電話作業をポーリングといい，isAvailable()メソッドで行う．

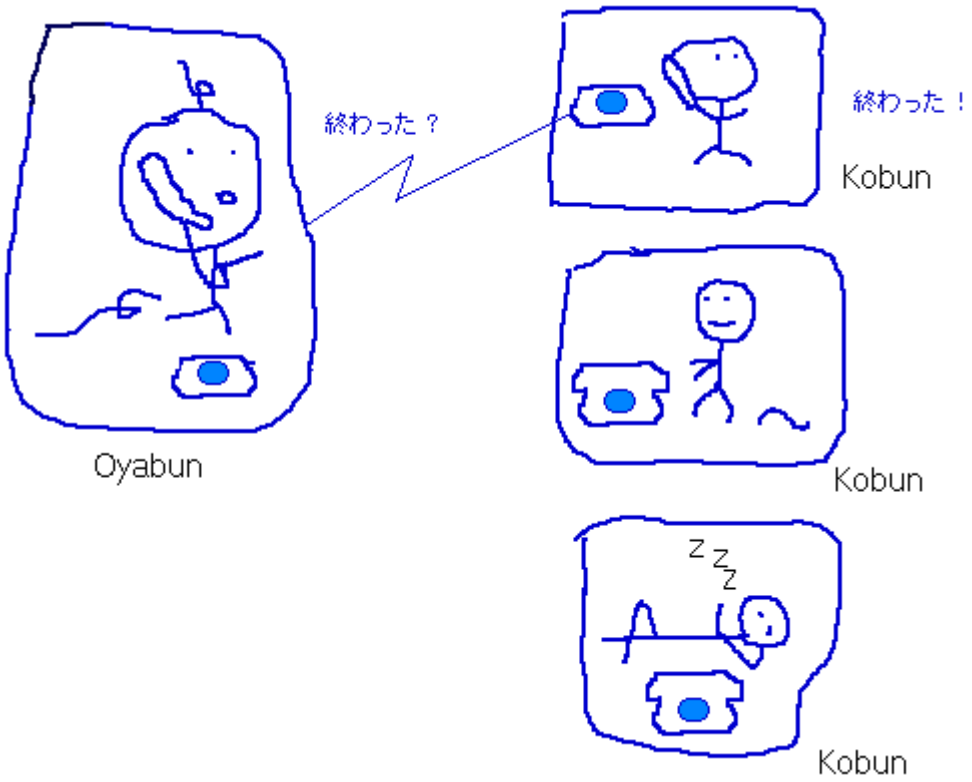

図 2.6．利口になった親分子分モデル

（c）　偉くなった親分モデル

親分の商売は繁盛して，ついには秘書を雇うことになった．子分は仕事を終えたら秘書に渡しに行くというルールに変更された．親分は子分の仕事の後始末を秘書に任せることができ，自分の仕事を止める必要が無くなる．

この方法は，setHandler()メソッドにより，親分クラスの自インスタンスとハンドル名を渡す．その後，_Request()メソッドを呼び出し，非同期処理が開始される．子分が仕事を終えると，HORB システムによって生成されている別スレッド（これが秘書である）を使って子分の run()メソッドが起動される．run()メソッドにはパラメータとして setHandler（）で渡したハンドル名を持っているため，どの非同期処理が完了したかという識別として使われる．また，非同期処理のため，run()メソッドは synchronized によって排他制御をする必要がある．

図 2.7．秘書を雇った親分子分モデル

2.4　開発言語

開発言語には，まずインターネット接続，HTML 解析が容易に行えるものとして，Perl，Java，C#などが挙げられた．（但し，C#は本システム開発当初，配布されていなかったので選択しなかった．）その中で，Java はマルチスレッドによる並列処理が可能で，分散オブジェクト技術の RMI や HORB，CORBA が利用できることから，本システム開発は Java で行うこととした．

Java にも様々な種類があるが，本システム開発では Sun Microsystems 社の JDK 1.3.1_01（開発当初で最新版のもの．2002 年 2 月現在では JDK 1.4.0 まで配布されている．）を使用した．そこで，Java に関する概要，利用する上での利点をまとめ，またプログラミングする上での TIPS などについて記述する．

2.4.1　Java の概要

Java はネットワーク処理を念頭に設計されたプログラミング言語である。世界中でインターネットが成長を遂げている中，Java が次世代ネットワークアプリケーション開発に適していることは他に類をあまりみることができない。Java を使用することで，プラットフォームからの独立，セキュリティ，国際文字セットの使用をはじめとする，多くの問題を解決することができる。これらはインターネットアプリケーションにとって必要不可欠と考えられるが，既存の他言語ではこれらの問題に対して対処が困難である。

2.4.2　本システム構築においての Java のメリット

本システム開発に Java を用いた理由やメリットについて以下に述べる．

（1）HTML 解析処理が容易

HTML ドキュメントの記述ルールはW3Cによって仕様が決められている．しかし，HTML 独特のドキュメントとしての柔軟性が非常に強いため，実際にはその仕様を守らなくても Web ページとして表示できてしまう．（あるいは，ブラウザがその柔軟性に対応している，と言った方が正確かもしれない．）そのため，Web ページを記述

する人たちがその仕様ルールを守って記述することは少ない．よって，HTML を文字列によって処理するためのプログラムコードを書くことは開発者にとって非常に難解となっている．

しかし，HTML は Web 文書の表示形式を表すものではあるが，その表示を示す“タグ”には記述ルールや意味がある．そこに注目して処理をする試みが言語仕様に取り入れられるようになってきた．Java も Version1.2.2 以降，HTML 解析・表示クラスを提供するようになり，JDK でも Version1.3 から本格的な HTML 処理技術を取り入れている．これにより，Java プログラマは文字列を直接扱わないですむようになっている．

具体的には javax.swing.text.html.parser パッケージを使用すれば，仕様をかなり逸脱した HTML ドキュメントでも読み込むことが可能である．本システムでは Document クラスと HTMLEditorkit クラスとの組み合わせによる方法で HTML を解析している．

（2）ガーベッジコレクションを所持している

C や C++といった言語ではメモリの割り当てを行うと，その領域が必要なくなった場合に，領域を解放，あるいはシステムに対して返却するのはプログラム側の責任となっている．メモリを使用した後に返却しなければプログラムはメモリリークと呼ばれる現象が起きる．（プログラムが終了した時点で空きメモリ領域がプログラムの実行前より少なくなっているような場合のことをメモリリークという．）メモリリークはシステムクラッシュなどを引き起こす可能性があるので，開発者は注意をしなければならない．

しかし，Java ではメモリリークの問題は解決されている．それはガーベッジコレクションという技術を使って，オブジェクトが使われなくなったときに自動的にオブジェクトのメモリを開放しているからである．ガーベッジコレクションは，一定時間ごとに参照されていない領域をスキャンすることによって，利用されていないメモリを自動的に解放している．さらに，ガーベッジコレクションはバックグラウンドでスレッドとして実行されているので，開発者は特に意識しなくて良い．本研究では大量の URL データを扱うため，どうしてもメモリ容量を意識しなければならないが，Java ならばメモリに関する作業を軽減できると考えた．

（3）様々な分散オブジェクト技術が容易に扱える

インターネット上にある大量の Web ページを処理するためには，複数台の PC を分散で処理することが必要になってくる．一般的にネットワーク分散型プログラミングというと CORBA が最もよく知られているだろう．CORBA は IDL によって多くのプログラミング言語に対応した分散オブジェクト技術の一つである．またそもそも Java にはリモートサーバ上の特定のメソッドを呼び出す RMI（Remote Method Invocation）が用意されている．そして，本研究で用いた HORB が提供されている．

このように，Java は様々な分散オブジェクト技術が利用できるため，将来どのような機能が必要になっても対応しやすいと考えた．

（4）DB 接続が容易

本研究では Web リンクを収集することを一つの大きな目的としているため，大量のリンクデータを格納する仕組みが必要になってくる．

DB に頼らない方法として，テキストファイルに保存することも考えられた．しかし，リンク数が大規模のためテキストファイルがすぐに大容量になって，プログラムから URL データを扱いにくくなる可能性がある．例えば，テキストデータとして扱うと，ファイルの読み込みに時間がかかったり，ファイルデータを読みこむ際にメモリ容量を越えてしまったりすることが考えられる．一方，DB を利用する場合は，データの格納領域の確保，検索が容易になる．

現在，DB は多数存在するが，有償，無償に関わらずそのほとんどには Java がアクセスするためのドライバが標準で用意されている．また Java には DB にアクセスするための JDBC，SQL などの API も組み込まれていることから，Java は DB を扱いやすいプログラミング言語であるといえる．

（5）マルチスレッド処理

本研究で構築するシステムは複数台のマシンによって分散処理をすることから，様々なタスクが同時実行されると考えた．C のように単一処理ではこれらのタスク処理が大変複雑になってしまう可能性がある．しかし，Java はマルチスレッドによって複数のイベント処理ができるので，同時に複数タスク処理が可能なプログラミングができると考えた．

2.4.3　本システム構築時のプログラミング TIPS

以下では，本システムを構築するにあたり，Java プログラミングの中でも特に重要な技術について簡潔に述べる.

（1）HTML 処理について[13]

Java には HTML の Tag を処理する HTML.Tag クラスが用意されている．HTML 内のほとんどのタグはこの HTML.Tag クラス･メソッドで処理できる. 具体的には, HTML 内の URL 取得を目的としていることから, HTML.Tag クラスの HTML.Tag.A と HTML.Tag.FRAME を使用した.

HTML.Tag.A は HTML のハイパーリンクを記述する<a>タグを処理するメソッドである．また HTML.Tag.FRAME は，<frame>タグを処理するメソッドである．FRAME タグを解析する仕組みを取り入れたのは，最近では Web ページの多くがフレームを使って記述されているためである.

具体的に HTML 処理は, HTML ドキュメントを Document クラスによってドキュメント化し，そのドキュメントを要素ツリー構造にして処理し，構造化した部品一つ一つについて<a>，<frame>タグがあるかをチェックし，もし<a>，<frame>タグがあるなら，その属性である<href>，<src>の属性値を取得する．ここで，HTML 文書の解析の流れを図 2.8 に示す.

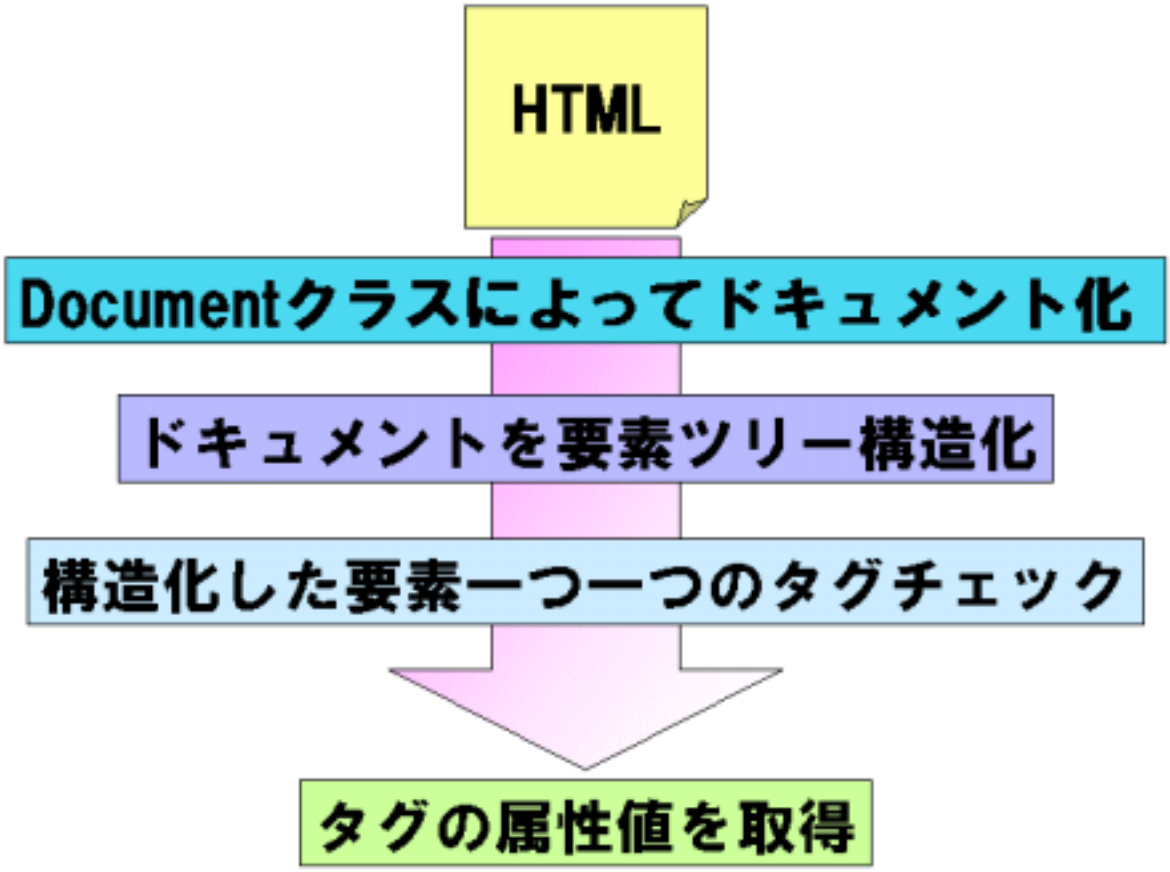

図 2.8. HTML 解析の流れ

（2）HttpURLConnection，URLConnection[13]

java.net.URLConnection は URL で指定されたリソースへのアクティブな接続を表す抽象クラスである．URLConnetion クラスは，以下の２つの特徴が挙げられる．

① Server とのやりとりについて（URL クラスと比べて）制御が自由に利く

② Java のプロトコルハンドラ機構の一部である．

（プロトコルハンドラとは，プロトコル処理の詳細を特定のデータ型処理や，ユーザインタフェースの提供や，モノリシックな Web ブラウザが実行するその他の処理と切り離す，というもの.）

URLConnection の問題点の一つとして，HTTP プロトコルに依存しすぎている点が挙げられる．例えば，転送ファイルには MIME ヘッダをつけるという前提になっ

ているが, FTP や SMTP といった従来からあるプロトコルでは MIME ヘッダは使用していない.

また本システムでは，httpURL を扱うので，URLConnection の抽象サブクラスである HttpURLConnection クラスも使用している．HttpURLConnection クラスはリクエストメソッドの取得・設定，リダイレクトに従うかどうかの決定，ステータスコードとメッセージの取得，プロキシサーバが使用されているかどうかの判定を行うなどのメソッドが定義されている.

本システムでは指定 URL 先のページを解析する前に HttpURLConnection クラスのインスタンスによって URL 先へ接続し，HTTP サーバが正常な状態で応答するかを確認させている．具体的には，URL クラスのインスタンスの openConnection によってサーバに接続し URLConnection クラスのインスタンスに渡す．そして接続されている URLConnection クラスのインスタンスを HttpURLConnection クラスのインスタンスに渡す．その HttpURLConnection クラスのメソッド getResponseCode() によって HTTP サーバの状態をチェックする．以下にソース例を載せる.

```
URL url = new URL("http://www.jaist.ac.jp");
URLConnection uc = url.openConnection();
HttpURLConnection hc = (HttpURLConnection)uc;
int code = hc.getResonseCode();
```

（3）Hashtable，Vector オブジェクト[14]

Vector クラスは, 要素数をデータの格納に必要な数に自動的に増減することが可能な配列である．また Vector クラスは要素の挿入，削除，検索に対応したメソッドを用意している.

Hashtable クラスは関連するキーを使って格納されたデータを検索可能なデータ構造のことである．Hashtable は項目とキーを関連付けておくことにより，キーを使って項目を検索することが可能である．Hashtable のキーには任意の型のオブジェクトを使用することもできる．但し，キーは全てユニークでなければならない.

（4）マルチスレッド[15]

本システムは，いくつかの並列処理を行うことを想定して設計した．並列処理を実現するには非同期メソッドとマルチスレッド処理が必要である．

例えば，クライアントの処理について言えば，「クライアントがサーバに命令をする処理」と「サーバから返ってきた処理をクライアントが受け取る処理」を並列で行う．また分散コンピューティングを構築することから，複数台のサーバから同時に結果が返ってくる可能性もある．これらに対応する方法としてマルチスレッドは有効な方法であろう．

2.5　データベース

DB 選択候補には，MySQL，PostgreSQL（以下，Postgres とする）が挙がった．どの DB も Java の対応ドライバを所持しており，言語と DB の接続は容易である．しかし，MySQL はトランザクションを備えていないため候補からはずした．

よって本システムでは DB にはトランザクション処理を所持し，容量制限が比較的大きい Postgres7.1.3 を使用した．ちなみに VineLinux2.1.5 は標準で Postgres7.1.1 がインストールされているが，セキュリティやバグ修正の問題などから 2001 年 9 月現在で最新のバージョン 7.1.3 を取り入れた．そこで，本章では Postgres の概要や利点などを記述する．また，DB への接続方法として JDBC を用いている．よって JDBC の概要，プログラミング TIPS についても本章で記述する．

2.5.1　PostgreSQL の概要

Postgres はカリフォルニア大学バークレー校において，DB 開発プロジェクトの中で生まれた．Postgres は DB 標準言語である SQL92 をサポートしており，トランザクション，サブクエリー，主キーなどの SQL92 の重要な機能はほとんどサポートしている．また Postgres は無償で利用でき，ソースコードが公開されており，UNIX や Linux など様々なプラットフォームに移植されている．そして，Java をはじめ，C や Perl など多くのプログラミング言語をサポートしている．

2.5.2　JDBC の概要

JDBC とは，現在広く使用されているリレーショナルデータベース（以下，RDB とする）へアクセスするための Java の API 仕様である．Data Base Management System （以下，DBMS とする）に依存しない Java プログラムを記述するための標準仕様となるべく，1996 年 2 月に Java Soft 社が発表した．Java RMI，Java IDL とともに Java Enterprise API として位置付けられ，JDK バージョン 1.1 からコア API として組み込まれている．

JDBC は具体的に以下の項目を行う API を提供する.

① DBMS に接続する.

② DBMS に SQL 文を送り，実行させる.

③ 実行結果を受け取る.

JDBC により，運用中の RDB 既存資産をそのまま Java アプリケーションで活用することができ，異なる DBMS に対しても単一のインターフェースでアクセスすることが可能である．さらに，作成した Java アプリケーションはプラットフォーム非依存である.

JDBC には２つの API があり，一つは上記で述べた JDBC API で，もう一つは JDBC ドライバへのインターフェースである JDBC ドライバ API である.

JDBC API は DB 製品の種類に関わらず基本的には同一なので, 開発者は DB の違いにあまり煩わされることはない.

一方，JDBC ドライバは DB に直接アクセスするので，個々の DB 製品による違いは JDBC ドライバが吸収する．したがって，JDBC ドライバは DB 製品ごとに用意しなければならない．JDBC ドライバには 2001 年 12 月現在では 4 種類のタイプがあり, ODBC ドライバを経由するもの, そうでないものなど, 実装方法に違いがある.

Postgres に付属する JDBC ドライバは，このうちダイレクトドライバと呼ばれるもので，Java だけで記述されており，しかも ODBC などに頼らず直接 DB に接続できる．このタイプの JDBC ドライバの利点は, ネイティブメソッドを含まないのでアプレットからも利用できること，プラットフォームを選ばず可搬性が高いこと，ODBC などを経由しないので性能が良いことなどが上げられる．欠点としては, すべてを Java で記述しなければならないのでクラスライブラリが大きくなりがちな点である.

2.5.3　JDBC プログラミング TIPS

以下では，DB や JDBC に関する特に重要な技術的要素について簡潔に述べる．[16][17][18][19]

（1）LIKE 検索

文字列データ型を扱う際，文字列マッチ検索方法して LIKE 述語がある．

本システムで取り扱うデータは URL であるため，文字列検索を頻繁に行うことが考えられる．なぜならば，URL データを大量に処理するため，どうしてもドメインやパスなどによって文字列検索を実行せざるを得ない．（LIKE 検索などを行わないとメモリ不足やデータ処理に非常に時間をとられてしまい，システム全体としてのパフォーマンスの低下つながる可能性がある．）また LIKE 検索とともに，Postgres 独自の仕組みとして正規表現がある．UNIX のシェルなどと同様に扱える機能で，特殊な文字を使って文字パターンを指定するものである．

以下に LIKE 検索と正規表現を用いた SQL 文の検索例を記述する．

（検索例）”abc”で始まる文字列を探す SQL 文

SELECT * FROM tablename WHERE text LIKE ‘abc%’;

（2）executeQuery,　executeUpdate

Java アプリケーション上で SQL 文を実行するには大きく２つのメソッド命令文がある．（実際には命令ごとに最適なメソッドがあるが，開発の労力を軽減させるために本システム開発ではこの２つの命令文のみで行った．）

一つは SELECT 文を実行する executeQuery メソッドである．SELECT 文は DB から結果が返ってくるので，結果を受け取る ResultSet クラスと併用する必要がある．

記述例としては，

```
ResultSet rs = stmt.executeQuery(query);
```

（stmt は Statement クラスのオブジェクト，query は DB に命令する内容を記述した String オブジェクト）

また，SELECT 文以外は命令を実行させるだけなので，executeUpdate メソッドを用いる．記述例としては，

```
stmt.executeUpdate(query);
```

（３）トランザクション処理

トランザクションは DB 操作の単位である．一般的にトランザクション処理とは，ある一つのテーブルに複数人のユーザが同時にアクセスすることによって起こる諸問題を防ぐための「排他制御」のことを言う．トランザクション処理は SQL92 に準拠しており，Postgres でも利用できる．

本システムでは複数台のマシンによるテーブルへの同時アクセスが十分にありうるので，トランザクション処理を取り入れることとした．

① トランザクション処理の開始

Postgres でトランザクション処理を開始するには，

```
BEGIN;
```

で開始を宣言し，

```
END;
```

でトランザクション処理を終了する．

② トランザクションの隔離レベル

トランザクションの隔離方法にはいくつかのレベルがあり, Postgres 特有の方式もある．本項目では，本システム開発時に使用した方法について記述する．

（a） リードコミット

他のユーザによって呼び出されて，まだコミットしてないデータを他のトランザクションが読み出せることによってデータの不整合性が発生してしまうダーティリードを防ぐレベルである．リードコミットはデフォルトに設定されているトランザクションレベルである．

（b） シリアライザブル

リードコミットレベルでは未コミットの値の影響を受けないが，他のトランザクションがコミットした値の影響をうけてしまう Non-repeatable Read を防ぐレベルである．

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

と宣言することでシリアライザブルレベルに変更できる．

（c） 明示的なロック

Postgres に限らずどの DB でも，同じテーブルの同じ行のように，全く同じオブジェクトに対して同時に更新をかけることはできない．何らかの方法でこれらの更新を調停する必要があり，これを同時実効制御と呼ぶ．そして同時実行制御の方法として最も一般的に使われているのがロックである．

ロックにはテーブル全体をロックするテーブルロックやアクセス対象の行のみをロックする行ロックがある．当然，行ロックの方が衝突する確率が低いため，Postgres は行ロックを採用している．

ロックには暗黙的なロックと明示的なロックがあり，システムが必要に応じてかけるのが暗黙的なロックであり，利用者が LOCK コマンドなどを使ってかける LOCK が明示的なロックである．

また明示的なロックは本システムの設計上，テーブルに対して LOCK をかける方法で行った．LOCK の使い方は，

LOCK TABLE tablename IN lockmode;

また，lockmode については，本システムではテーブルに対して同時に 1 個のトランザクションだけがロックを獲得できる SHARE ROW EXCLUSIVE MODE というロックモードを使用した．

（4）JDBC ドライバについて

Java アプリケーションからネットワーク経由で Postgres にアクセスするには，Java プログラムを実行するマシン上にも JDBC ドライバをインストールし，CLASSPATH の設定をする必要がある．

（5）主キーについて

一般的な DB と同様に，Postgres でもテーブルには主キーを設定できる．主キーは一つまたは複数の列から成り，それらを連結した値はそのテーブルの中で唯一であるという制約を受ける．（よって主キーには NULL を入力することは許されない．）

主キーを設定する SQL 文の例を記述する．

CREATE TABEL tablename (i INTEGER PRIMARY KEY);

第 3 章
システム全体の処理フロー

本章では，構築した Web リンク収集システムの全体像について述べる．

3.1　システムの全体像

システム全体の流れを図 3.1 に示す．

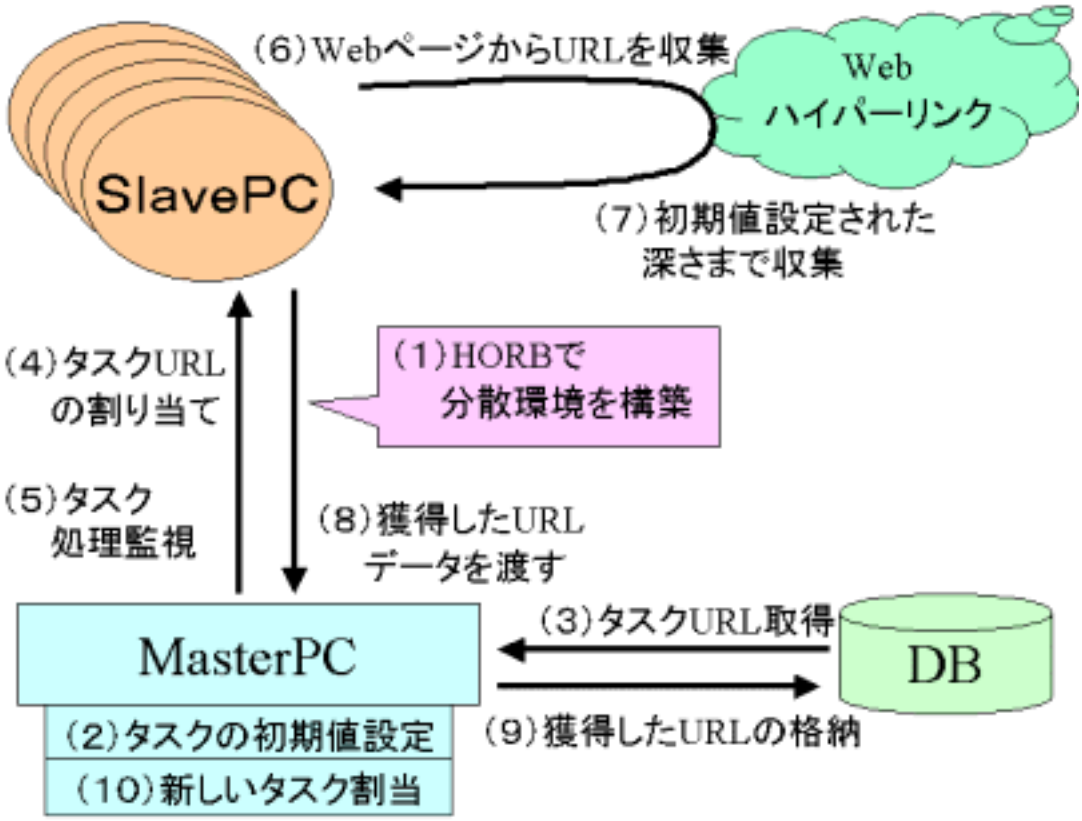

図 3.1.　システム全体の処理フロー

システムの流れについて図のカッコ番号に沿って説明する.

（1）HORB で分散環境を構築

本システムは Mater-Slave による負荷分散の方式を取っている. この分散環境を実現するために分散オブジェクト技術である HORB を用いている.

Master は，Slave に初期値設定に基づいたタスク割り当てを行い，また Master は Slave のタスク処理を監視する．Master がタスクの割り当て，監視を行い，獲得した URL データを格納することで，Slave は割り当てられたタスク URL 先の Web ページを解析することだけに専念させることができる．これにより並列処理による負荷分散を可能としている.

複数台から結果を取得するために，非同期メソッドの使用とマルチスレッドプログラミングを実装している.

（2）タスクの初期値設定

Master は Slave のタスク量を決める URL 数とリンク収集の深さの初期値変数をもつ.

ここで深さとは，渡された URL からハイパーリンクをたどっていく回数である. 深さの値が大きくなれば解析するページ数も多くなる.

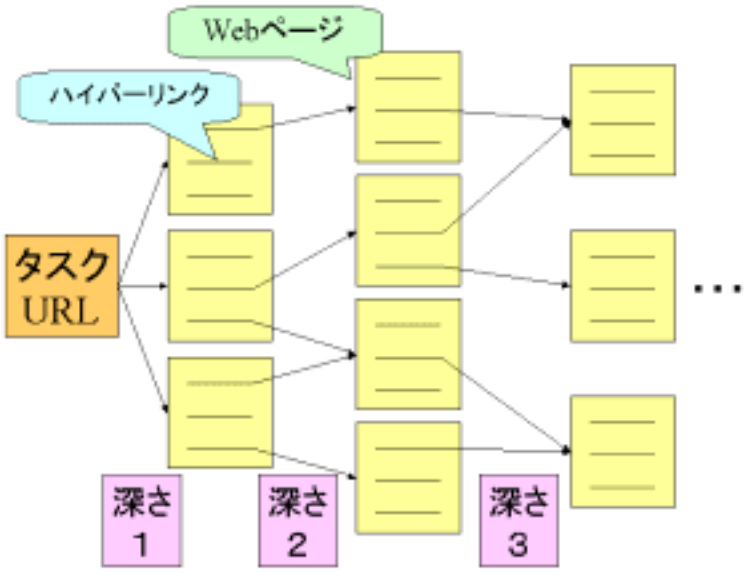

図 3.2. リンクの深さ

（3）タスク URL 取得

DB に格納されている，まだ解析されていない URL 群から初期値設定の数だけ URL を取得する．（この URL をタスク URL とする）

（4）タスク URL の割り当て

DB から取得したタスク URL を各 Slave に割り当てる．

（5）タスク処理監視

タスクを割り当てた後，Master は Slave がタスク処理を終えたかを監視する．

（6）Web ページから URL を収集

Slave は割り当てられたタスク URL 先の Web ページを解析し, Web ページに存在するハイパーリンクを獲得する．

（7）初期値設定された深さまで収集

Slave はタスク URL とともにリンクの深さも渡され，その深さまで Web ページを解析し続ける．

（8）獲得した URL データを渡す

Slave が獲得した URL データを Master に返す．この時，Master は別スレッドをたてて Slave からの URL データを受け取る．

（9）獲得した URL の格納

（8）で受け取った URL データは，いくつかのチェック事項にかけられた後，DB へ格納される．

（１０）新しいタスクの割当

Master は，Slave がタスクを渡したことを確認した後，DB から新たな URL タスクを取得して Slave に再び渡す．

新しい URL タスクは，解析していない URL が登録されているテーブルからラン

ダムで選択することとした.

本システムは（1）～（１０）の処理を繰り返す.

3.2　システムの詳細事項

3.2.1　Web リンク獲得処理時のチェック項目

Slave がタスク URL 先を解析する直前にいくつかのチェックを行っている．ここでそれらチェック項目について述べる.

（1）http チェック

獲得した URL には https や ftp など, 様々な URL が含まれている. 今回は httpURL のみを対象としているため，http 以外は解析しないようにした.

（https はセキュリティ要素を含むプロトコルであり，一般的にユーザ名やパスワードなどを必要とするため，接続できない）

（2）解析済み URL チェック

Slave が一度解析したページを再び解析しないようにチェックをしている．但し，これは Master からタスク URL が渡され，戻り値を Master に返すまでの間でのみ有効なチェック項目である．これは，戻り値を Master に返すと VM を破棄するのでオブジェクトを保持できないためである．また，解析済 URL データを DB へ格納する方法も考えられるが，この方法だと Java が大量のデータを扱わなければならなくなるので，パフォーマンスが低下すると考えられる.

（3）ストップリストチェック

本システムではストップリストを２つ設けた.

一つは Yahoo！や goo などといったハイパーリンクが多く張ってある, いわゆるポータルサイトを集めたものである. Yahoo! などのポータルサイト URL が解析対象になった場合は，解析作業をスキップするようにした.

もう一つは，タスク URL 先に接続した際，Web が正常に存在していないなら，その URL を DB のストップリストに登録する仕組みを取り入れた. ちなみに Web の正常でない状態とは, サーバ上にファイルがない, リダイレクション設定がされている, 接続に非常に時間がかかる，というものである.

また，ストップリスト URL と同じフォルダ以下にある URL にも接続しないようにした．これは，同じフォルダ内にあるファイルは同じ状態である可能性が高いと考えたためである.

例えば, http://www.jaist.ac.jp/~masa-i/aaa/bbb.html という URL がストップリストに登録されたら, http://www.jaist.ac.jp/~masa-i/aaa/ 以下の URL にはアクセスしない.

3.2.2　マルチスレッド処理

本システムは，いくつかのスレッドに分けて処理させている．

Maser では，大きく２つのスレッドが動いている．一つは Main メソッド処理を行うスレッドで，もう一つは Slave からの戻り値を受ける run メソッド処理を行うスレッドである．

また Slave は，タスク URL 先の Web ページを解析するスレッドと Web ページが正常な状態で存在するかを確認するスレッドで構成している．

これらのスレッドと各スレッドの役割について表 3.1 に記す．

また，この他にもバックエンドではガーベッジコレクションなどのスレッドが動いているが，それらはプログラミングで特に意識しなくても自動で立てられるスレッドなので，ここでは特に述べない．

役割	スレッド	各スレッドの仕事内容
Master	Mainメソッド スレッド	DBからタスクURLの取得
		Slaveへのタスク割り当て
		Slave監視
	runメソッド スレッド	SlaveからのURLデータの受取り
		URLデータをチェック
		URLデータをDBへ格納
Slave	タスクURL先の Webページ解析	割り当てられたURL先のWeb ページを順に解析していく
		初期値設定された深さまで Webページを解析する
	タスクURL先の HTTP URL確認	Webページを解析する前に，URL先に接続してWebページが正常に存在するかを確認する

表 3.1．各スレッドとスレッドの仕事

Master の Main メソッドでは，DB からタスク URL を取得，Slave へのタスクの割り当て，Slave のタスク処理の監視を行う．また Master の run メソッドは Slave からの戻り値である URL データを受け取り，データのチェック，格納を行う．

Slave は，Master の Main 処理から URL タスクを受け取り，URL 先の Web ペー

ジを解析する．但し，URL 先の HTML が正常に存在しているかを確認するために，別スレッドをたてた．別スレッドによって確認作業を行うのは，Web 上にはリンク先が正常に存在しない場合が多々あるため，HTTP サーバにアクセスし状態を確認する作業をするようにした．

3.2.3　テーブルの構成

本システムには DB を設けている．ここで，DB 内のテーブル構成について簡単に述べておく．

（1）インプット URL テーブル

これは Slave PC に渡すタスク URL 群を登録しておくテーブルである．システムを動かすには，始めからこのテーブルにデータセットを登録しておかなければならない．（4 章で行う実験ではこのテーブルに 100 個のデータセットを登録した．）

テーブル構成は以下のようにした．

URL の No.	URL
Integer 型	Text 型

表 3.2.　インプット URL テーブルの構成

（2）ストップリストテーブル

3.2.1 で述べたストップリストを登録しているテーブルである．

システムを動かす前に，Yahoo!のようなポータルサイトや，接続してはいけない URL 先などを登録しておく．また，システム実行中にはストップリストが追加される仕組みををとりいれている．

テーブル構成は以下のようにした．

URL
Text 型

表 3.3.　ストップリストテーブルの構成表

（3）解析済み URL テーブル

Slave PC が解析した Web ページの URL を登録しておくテーブルである．このテーブルに登録してある URL 先には接続しない．

テーブル構成は以下のようにした．

URL の No.	URL
Integer 型（主キー）	Text 型

表 3.4.　解析済み URL テーブルの構成

このテーブルでは登録してある URL 先の Web ページにどんなリンクが張られているかを管理するために，URL に No を割り当てている．その No を主キーとすることで管理の効率性を上げるようにした．

（4）獲得リンク格納テーブル

（3）の解析済み URL 先の Web ページにあるハイパーリンクを一元管理するテーブルである．

テーブル構成は以下のようになっている．

解析済み URL の No	URL
Integer 型	Text 型

表 3.5.　解析済み URL テーブルの構成

このテーブルには（3）の主キーNo を登録するようにしている．この構成によって，どの URL 同士がつながっているかがわかる．

3.3　システムの各要素の処理

第 2 章 2.1 で定義したシステムの仕様が満たされているかを論述したい.

（1）Web リンクの収集について

図 3.1 の（6），（7）により，WWW サーバに接続し，HTML を読み込み解析する仕組みを構築できている.

（2）分散環境の構築について

図 3.1 の（1），（2），（4），（5），（8），（１０）により，Master-Slave 方式による負荷分散を行い，分散環境を構築できている.

（3）データベースとの連携について

図 3.1 の（3），（9）によりデータベースとの連携が図れ，URL データを取得・格納できている.

第４章
システムの評価実験の手順と結果

本システムの分散環境はMaster－Slave方式の負荷配分を採用し，MasterはSlaveに与えるタスク初期条件（初期 URL 数，リンクの深さ）を変化させることができるものとした．そこで，ハイパーリンクを収集する Slave PC の台数変化や初期条件変化による収集パフォーマンスに現れる影響について述べる．

構築したシステムの性能を評価するために以下の２つの実験を行った．

実験１：時間量と Slave PC 台数の変化によるリンク収集への影響

① 一定時間ごとに解析できた Web ページ数と獲得したリンク数についてカウントした

② ①を Slave PC2 台，5 台，10 台について行い，比較した

実験２：タスク URL 数と収集の深さの変化によるリンク収集への影響

① タスク URL 数を１，５，１０と変化させ，解析した Web ページ数と獲得したリンク数についてカウントした

② 収集の深さを２，４，６と変化させ，解析した Web ページ数と獲得したリンク数についてカウントした

③ ①，②を，Slave PC2 台，5 台，10 台について行い，比較した

本章では，それら評価実験の方法とその結果について述べる．

4.1　実験の方法

本実験では，それぞれの実験ごとにいくつかの条件を定めている．各実験の条件については下記で記述する．

但し，初期 URL データセットは日本の大学の TOP Web ページ１００個とした．また解析する Web ページは JP ドメインに限定した．

（実験１）時間量の変化と Slave PC の台数変化によるリンク収集への影響

収集する時間を 30 分，60 分，90 分とし，各単位時間での「解析できたページ数」と「獲得した URL 数」をカウントした．

また，Slave PC の台数を２，５，１０台と変えて，比較する

以下の項目は実験１の全実験に関わる事項である．

- STOPLIST に追加された URL 数についてもカウントした
- 初期値設定はタスク URL を３，リンクの深さを４とした
- 初期 URL データセットは実験毎に変更した
- 実験は 3 回行い，その結果の平均値で評価した

2 台			5 台			10 台		
30 分	60 分	90 分	30 分	60 分	90 分	30 分	60 分	90 分
初期 URL 数は３，深さ４で，各実験とも 3 回ずつ行った．								

表 4.1.　実験１の内容

（実験２）初期値設定の変化によるリンク収集への影響

この実験では「初期タスク URL 数」と「深さ」を変化させることによるリンク収集への影響について調べた.

各実験は 60 分間で「解析できたページ数」と「獲得した URL 数」で評価することとした.

以下の項目は実験２の全実験に関わる事項である.

- STOPLIST に追加された URL 数についてもカウントした
- 初期 URL データセットは実験毎に変更した
- 実験は 3 回行い，その結果の平均値で評価した

①初期タスク URL 数の変化

初期タスク URL 数を１，３，５と変化させ，解析した Web ページ数と獲得した URL 数をカウントした．この実験では，下記のような組み合わせで行った.

初期 URL 数	1	5	１０
その他の条件	Slave PC2 台，深さ２，実験を 3 回ずつ		

表 4.2.　実験２：Slave PC2 台での初期 URL 数の変化の実験条件

初期 URL 数	1	5	１０
その他の条件	Slave PC５台，深さ４，実験を 3 回ずつ		

表 4.3.　実験２：Slave PC5 台での初期 URL 数の変化の実験条件

初期 URL 数	1	5	１０
その他の条件	Slave PC１０台，深さ２，実験を 3 回ずつ		

表 4.4.　実験２：Slave PC１０台での初期 URL の変化の実験条件

②リンク収集の深さの変化

リンク収集する深さを２，４，６と変化させ，解析するページ数と獲得したリンク数についてカウントした．この実験では，どの深さでも台数２，５，１０台と初期 URL 数１，３，５の組み合わせで実験を行った．

深さ	2	4	6
その他の条件	Slave PC２台，初期 URL 数１，実験を 3 回		

表 4.5.　実験２：Slave PC２台で深さの変化の実験条件

深さ	2	4	6
その他の条件	Slave PC５台，初期 URL 数３，実験を 3 回		

表 4.6.　実験２：Slave PC５台で深さの変化の実験条件

深さ	2	4	6
その他の条件	Slave PC１０台，初期 URL 数５，実験を 3 回		

表 4.7.　実験２：Slave PC１０台で深さの変化の実験条件

4.2　実験の結果

本章では，4.1 で述べた実験条件に基づいて行った結果を以下に記述する．

4.2.1　全実験データを用いた解析

実験１，２で得られた解析 Web ページ数，獲得リンク数，ストップリストに追加された URL 数について，簡単な統計処理を行った。

その結果を表 4.8 に示す。

	解析ページ数	獲得リンク数	ストップリスト数
平均	2849.7	34639.9	135.7
標準偏差	2086.1	26052.7	115.8
範囲	7608.0	91173.0	447.0
最小	333.0	3251.0	7.0
最大	7941.0	94424.0	454.0
合計	205177.0	2494075.0	9767.0
標本数	72	72	72

表 4.8.　全実験データの基本統計量

（1）1 ページあたりのリンク数について

上記表 4.8 から，解析ページ数の合計と獲得リンク数の合計によって，1 ページあたりどのくらいハイパーリンクが張られているかを計算した。

2494075÷205177＝12.16

本実験で得られた結果の範囲では，1 ページに約 12 個のハイパーリンクが張られているといえる。

（2）ストップリストの割合

上記表 4.8 から，解析ページ数の合計とストップリストの合計によって，どのくらいの割合でストップリストに追加される URL にあたるかを計算した。

454÷（7941＋454）＝0.0540

本実験で得られた結果の範囲では，約５％の確率で FileNotFound などの正常状態でない Web ページに遭遇するといえる。

（３）解析ページ数と獲得 URL の相関関係について

（１）で述べた解析ページ数と獲得リンク数の間に相関関係について，散布図と相関係数によって調べた。

以下にその結果を示す。

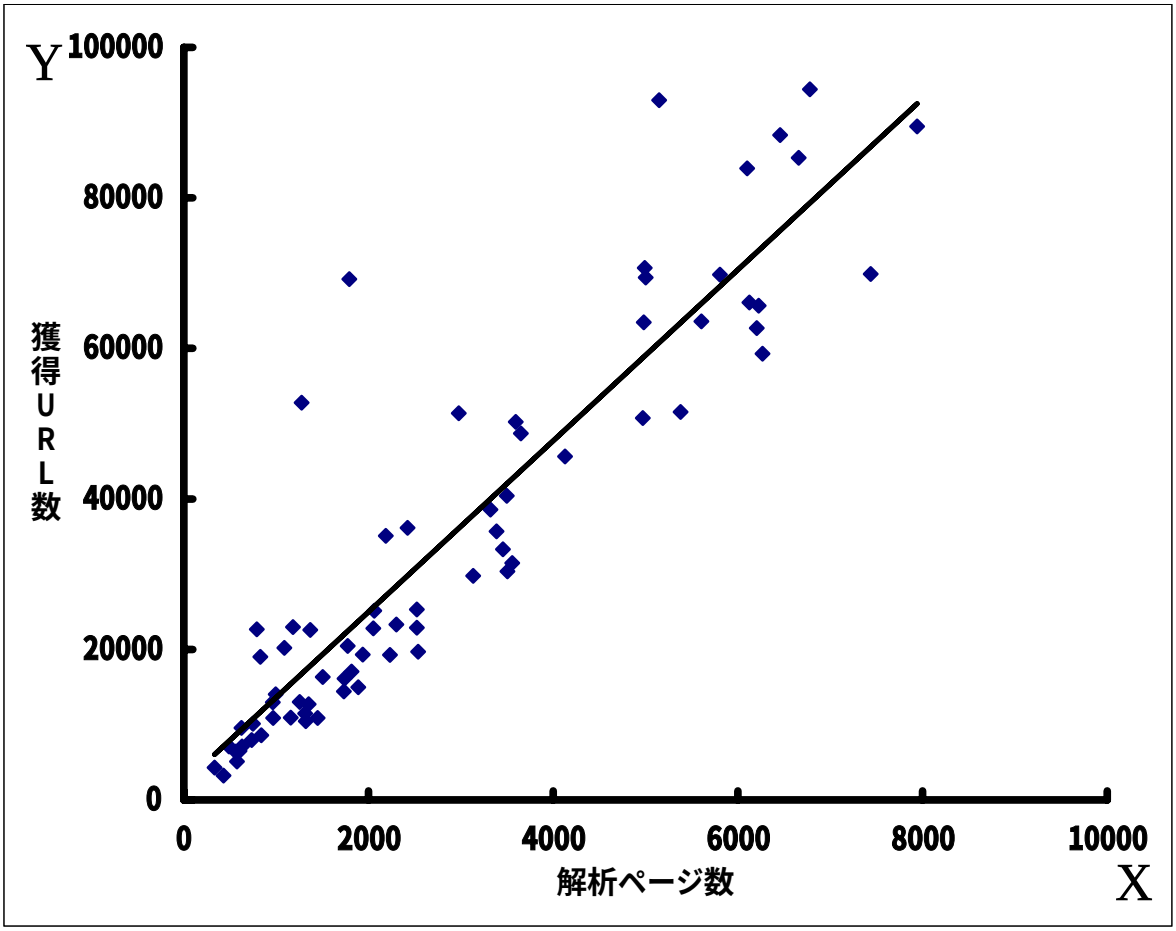

図 4.1.　全実験データによる解析 Web ページ数と獲得リンク数の散布図

相関係数 R＝0.91 となったので，解析 Web ページ数と獲得リンク数は強い相関があるといえる。

$$Y \propto 12 \quad Rx$$

（３）解析ページ数とストップリストの相関関係について

また，（３）と同様に，解析 Web ページ数とストップリストに追加される URL 数の間の相関関係についても散布図と相関係数によって調べた。

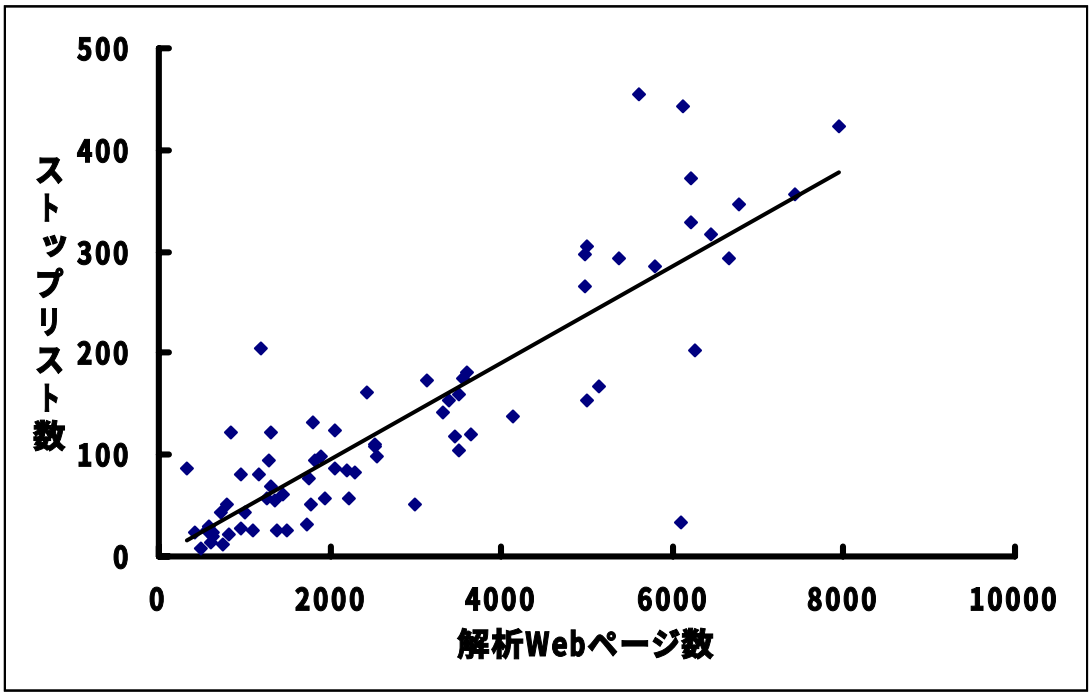

図 4.2.　解析 Web ページ数とストップリスト数の散布図

相関係数 R=0.86 となったので, 解析 Web ページ数とストップリストの増加数は強い相関があるといえる。

$$Y \propto \frac{5}{100} Rx$$

4.2.2　実験１の結果：時間量と Slave PC 台数の変化による影響

次に，実験１の結果を以下に記す。

（１）一定時間単位での解析 Web ページ数

Slave PC を 2 台，5 台，10 台と変化させて，30 分，60 分，90 分間で解析できた Web ページ数のグラフを以下の図 4.3 に示す。

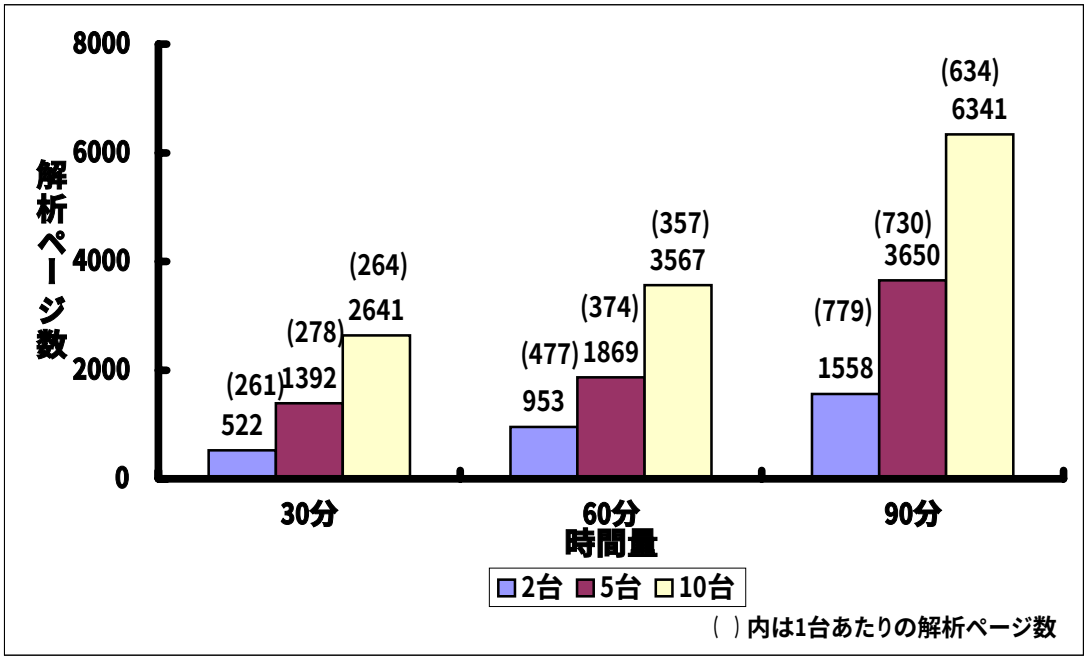

図 4.3.　　単位時間での解析ページ数

全ての台数で，時間量が増えるとともに解析した Web ページ数が増えている。

Slave PC2 台の場合は 30 分間の平均で約 500 ページずつ解析していることがわかる。

Slave PC5 台と 10 台では，30 分から 60 分の間での解析 Web ページ数の伸びはそれほど大きくない。しかし，60 分から 90 分の間では，5 台でも 10 台でも解析 Web ページ数は平均で約 2 倍の解析数に増加している。

（２）一定時間単位での獲得リンク数

Slave PC を 2 台，5 台，10 台と変化させて，30 分，60 分，90 分間で獲得できたリンク数を以下の図 4.4 に示す。

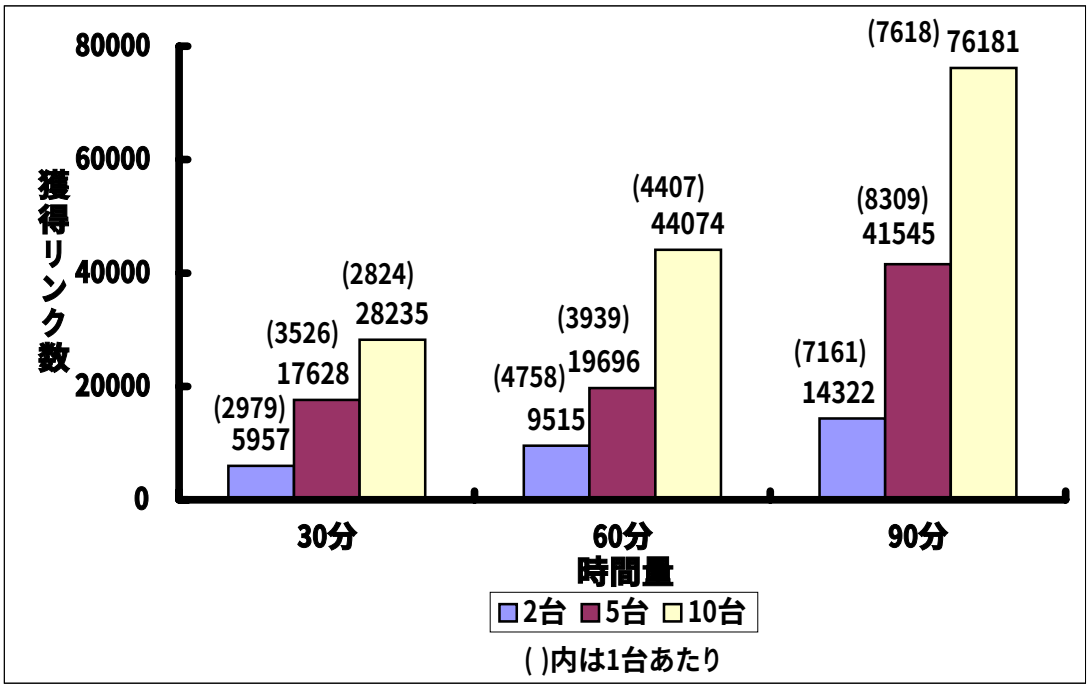

図 4.4.　　時間量変化による獲得リンク数

どの台数条件でも時間量の増加ともに獲得リンク数が増えている。

Slave PC2 台の場合，30 分ごとに約 5000 リンク増えている。

Slave PC5 台の場合，60 分の時はほとんど増えていない。

また，10 台のときも，30 分から 60 分の増加に比べて，60 分から 90 分の増加が非常に大きい。

（３）一定時間単位でのストップリストの増加数

実験条件で定めた時間単位で，ストップリストに追加された URL 数について以下の図 4.5 に示す。

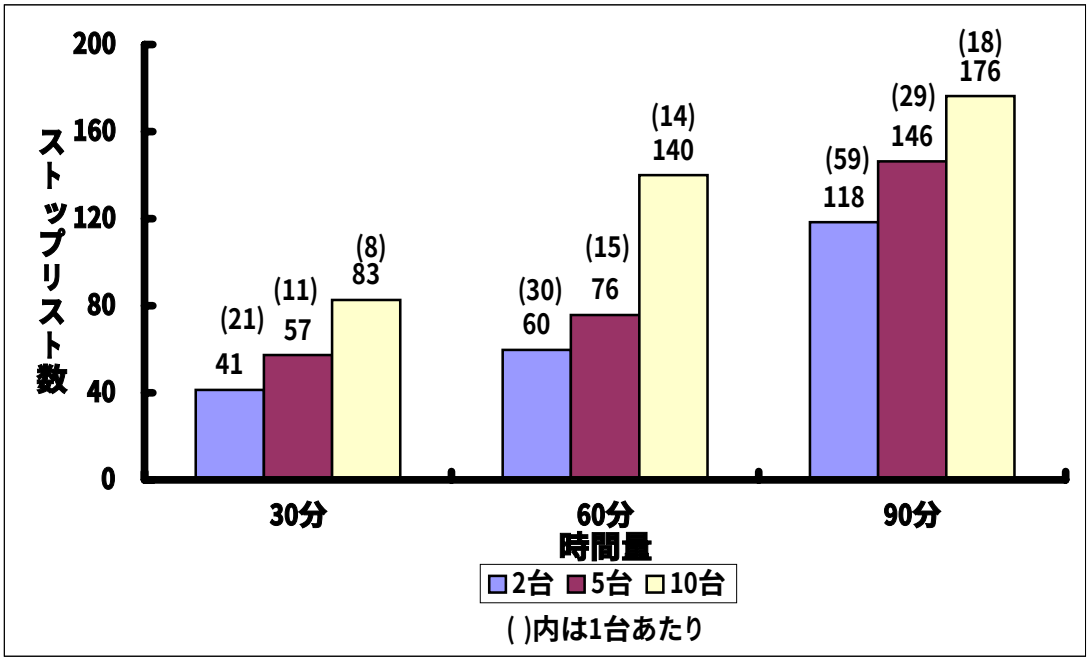

図 4.5.　時間量変化によるストップリストの増加数

各台数とも，時間の増加とともにストップリストの URL が増加していることがわかる。

Slave PC2 台，5 台とも，30 分から 60 分での増加がそれほど大きくない。しかしどちらの台数でも 60 分から 90 分の間で約 2 倍増加している。

10 台の場合は，2 台や 5 台の時は違い，徐々にストップリストの増加が現状している。

（4）1 分間での解析ページ数の比較

実験条件で定めた時間で解析できた Web ページ数について, 1 分間で解析できたページ数に直したグラフを図 4.6 にしめす。また，同じ内容の表も下記に記す。

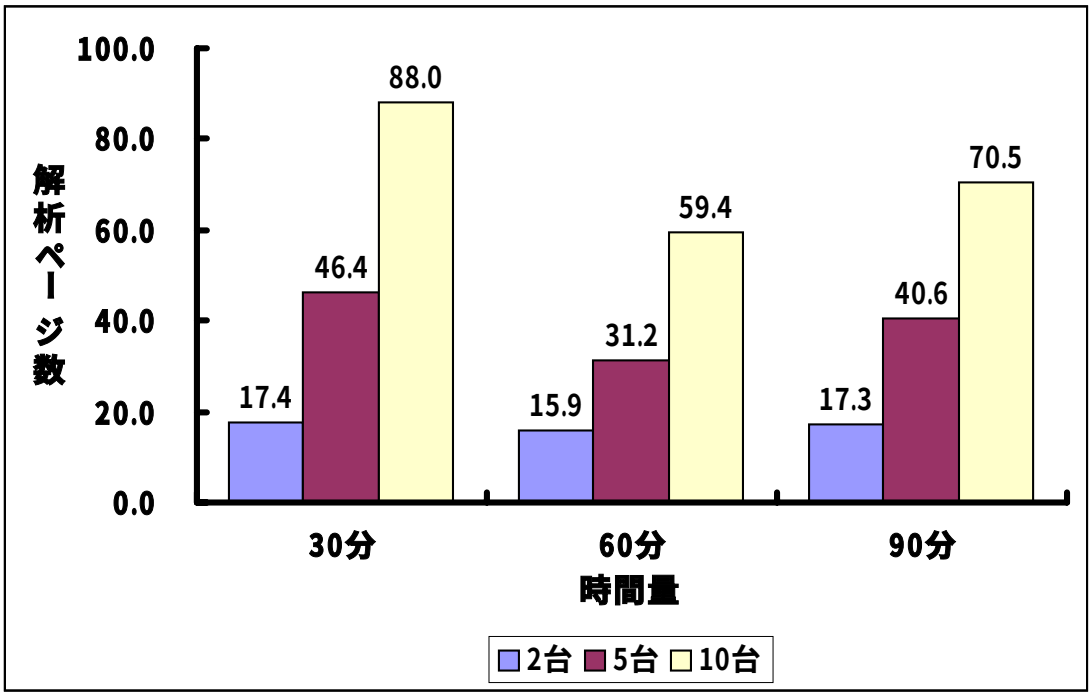

図 4.6　1 分単位での解析 Web ページ数

	30 分	60 分	90 分
2 台	17.4	15.9	17.3
5 台	46.4	31.2	40.6
10 台	88.0	59.4	70.5

表 4.9.　　1 分単位での解析 Web ページ数

Slave PC2 台の場合はあまり時間量の影響を受けず，解析できるページ数は変わっていない。

Slave PC が 5 台と 10 台の場合は時間量が増加すると解析できるページが減っている傾向がある。

また，60 分の平均値が，どの台数の場合でも一番低くなっている.

（5）1 台あたりの解析 Web ページ数

各台数条件でのパフォーマンスを測るために，（4）のデータを Slave PC の台数で割り，その結果を下記の図 4.7 に示す。

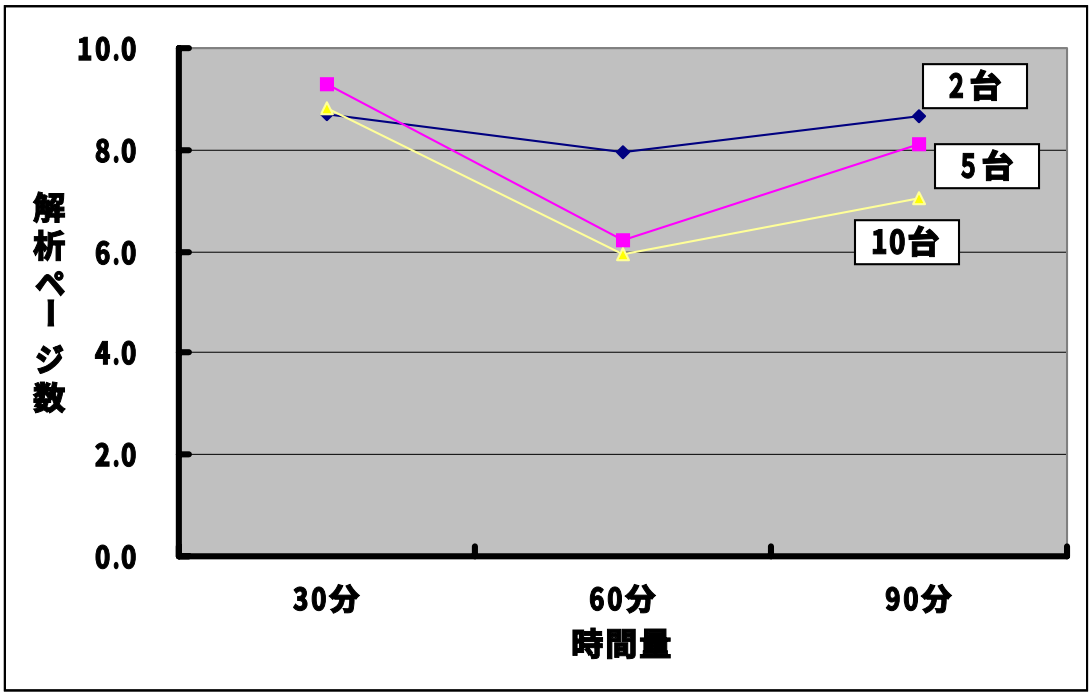

図 4.7.　1 台あたりの 1 分間で解析できる Web ページ数

	30 分	60 分	90 分
2 台	8.7	7.9	8.7
5 台	9.3	6.2	8.1
10 台	8.8	5.9	7.0

表 4.10.　1 台あたりの 1 分間で解析できる Web ページ数

この結果から，時間がある程度たつと，1 台あたりのパフォーマンスは台数が多いほど落ちていることがわかる。

また，60 分の平均値がどの台数の場合でも一番下がっている.

4.2.3　実験２の結果：初期タスク量変化，深さ変化による影響

（1）初期タスク URL 量の変化

Slave PC に渡すタスク URL 量の変化によって解析できる Web ページ数に変化があるかを調べた．その結果を示す．

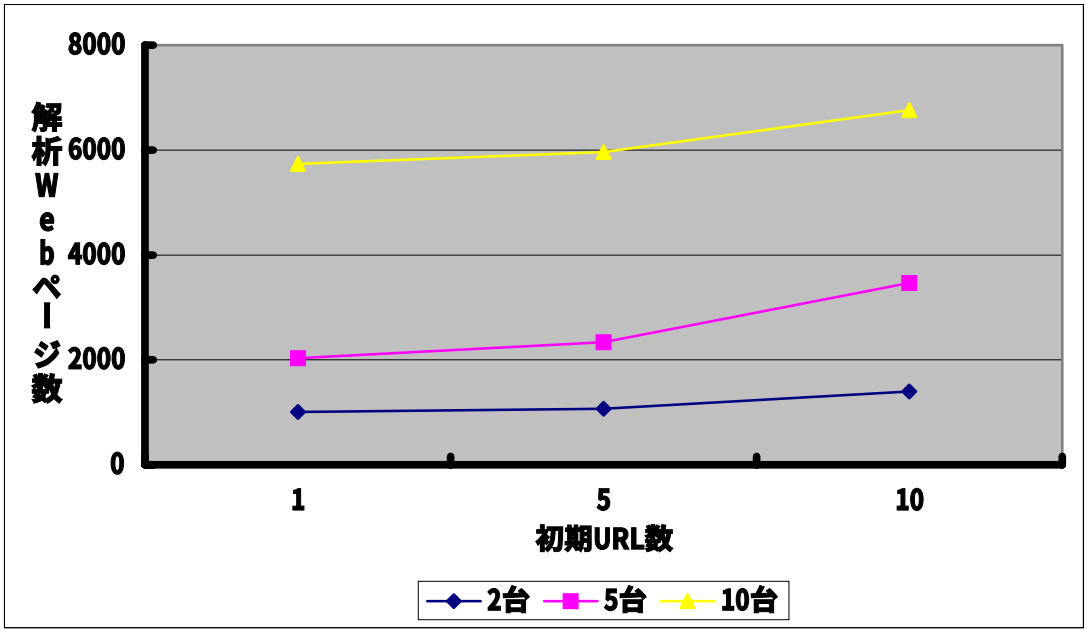

図 4.8.　初期タスク URL 量の違いよる解析 Web ページ数

	1	5	10
2 台	1005.7	1068.0	1396.3
5 台	2031.0	2336.3	3466.3
10 台	5736.7	5963.7	6763.0

表 4.11.　初期タスク URL 量の違いよる解析 Web ページ数

どの台数においても，URL 数が大きくなるほど，解析できる Web ページ数が増えていることがわかる．

次に，上記データについて 1 台あたりの Web ページ解析数を見てみる．

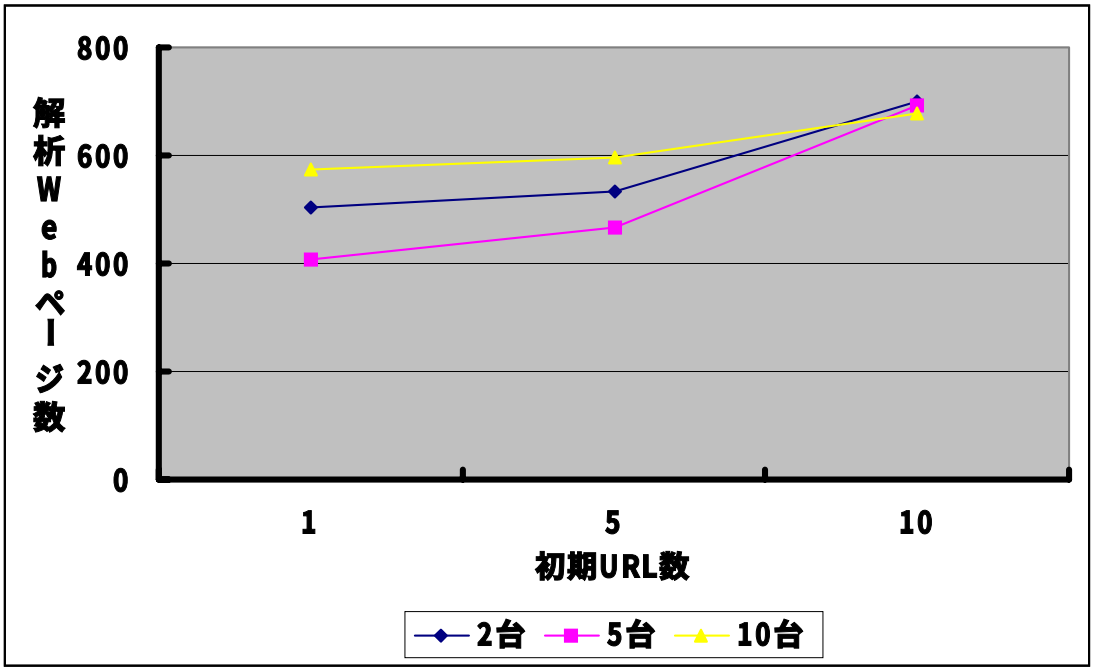

図 4.9.　初期タスク URL 数の違いによる 1 台あたりの解析 Web ページ数

	1	5	10
2 台	502.8	534.0	698.2
5 台	406.2	467.3	693.3
10 台	573.7	596.4	676.3

表 4.12.　初期タスク URL 数の違いによる 1 台あたりの解析 Web ページ数

どの台数でもタスク URL 数が大きいほど, 解析できる Web ページ数が多くなっている．また，深さが１０の時はどの台数でもだいたい同じ数のページ数を解析している．

（2）リンク収集の深さの変化

まず，深さの違いによる台数条件ごとの解析できた Web ページの総数をグラフに表す。

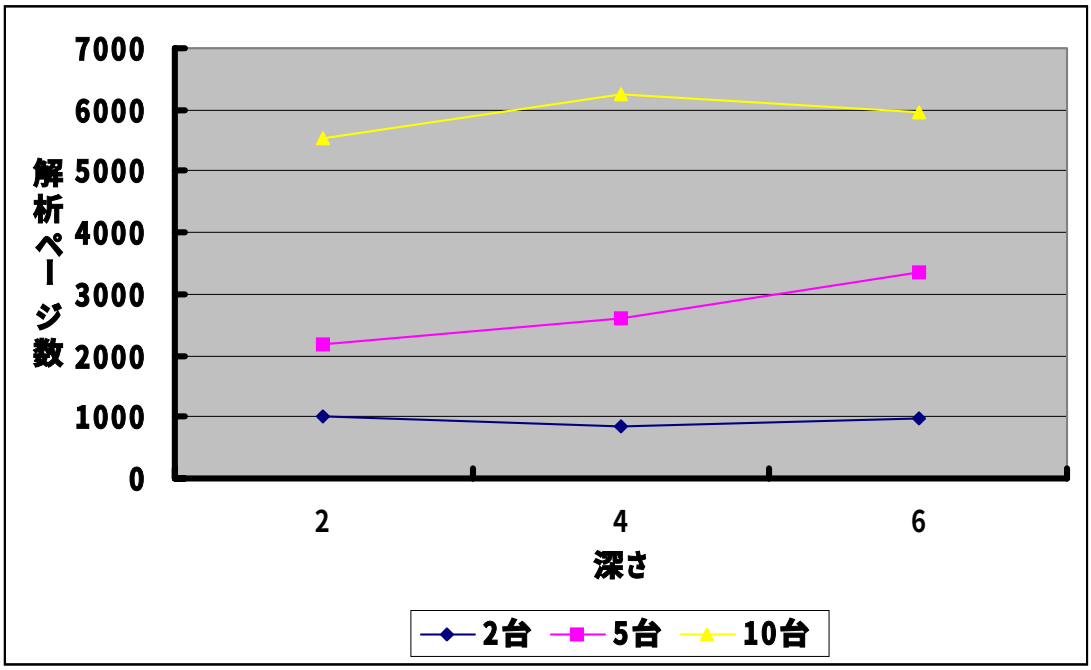

図 4.10.　リンク収集の深さの違いによる解析 Web ページ数

	2	4	6
2 台	1005.7	852.7	962.7
5 台	2168.0	2600.5	3358.3
10 台	5535.3	6263.0	5963.7

表 4.13.　リンク収集の深さの違いによる解析 Web ページ数

Slave PC2 台の場合は，深さを変えてもそれほど変化は見られない．

Slave PC5 台の場合は，深くなるにつれて解析できた Web ページ数が多くなっている．10 台の場合は，深さ 2 の場合よりも深さ 4 や 6 の方が解析できたページ数が増えているが，深さ 4 よりも深さ 6 の方が解析 Web ページ数が減少している．

次に，システムのパフォーマンスを評価するために，台数条件ごとの Web ページ解析総数をその台数で割った結果をグラフにした。

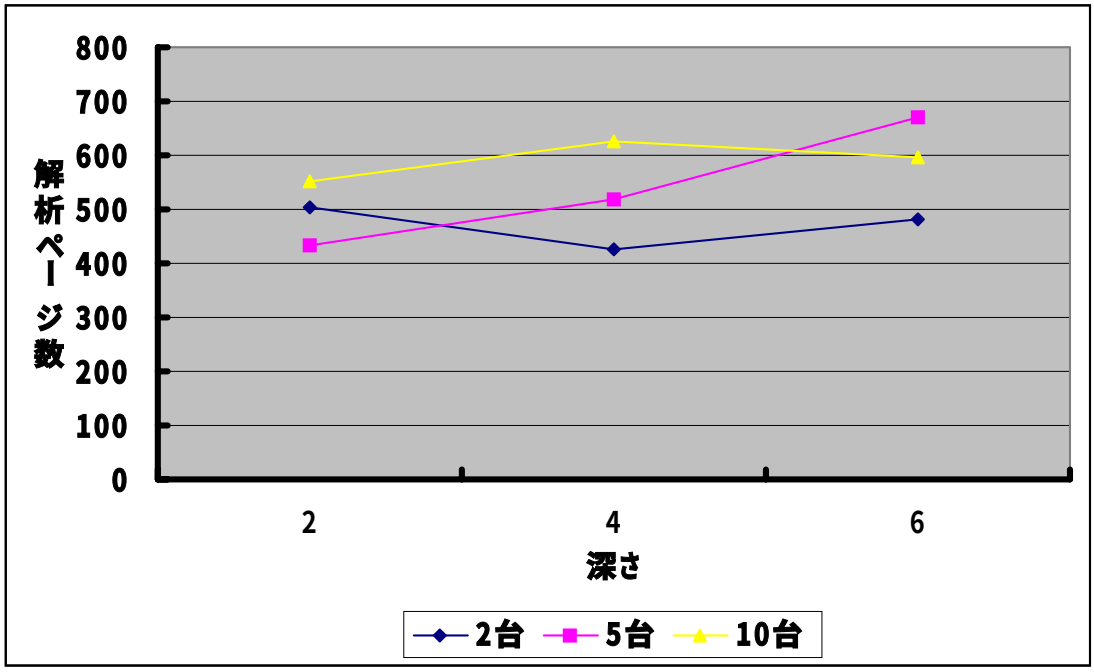

図 4.11.　リンク収集の深さの違いによる 1 台あたりの解析 Web ページ数

	2	4	6
2 台	502.8	426.3	481.3
5 台	433.6	520.1	671.7
10 台	553.5	626.3	596.4

表 4.14.　リンク収集の深さの違いによる 1 台あたりの解析 Web ページ数

1 台あたりで見てみると，台数にかかわらず解析できるページ数が近い事がわかる．

Slave PC2 台の場合は，リンクの深さとともに解析できる Web ページ数が減っているものの，それほど大きな変動ではない．

Slave PC5 台の場合は，深さが大きくなると解析できる Web ページ数が多くなっている．10 台の場合も 5 台と同様であるが，5 台の場合と比べて増加量が小さい．

第 5 章
結果のまとめと考察

5.1　実験結果のまとめと考察

4 章で行った実験結果を以下にまとめ，その考察を行う．

○Slave PC 台数と時間変化の影響

初期値設定などを同条件にして，ある一定時間ごとの解析 Web ページ数をカウントした結果，1 台あたりの解析数は Slave PC の台数が多い程落ちていることがわかった．（考察については 5.1.1 で述べる）

○タスク URL 数とリンク収集の深さの影響

タスク URL 数の数量が多くなると，どの Slave PC 台数の場合でも 1 台あたりの解析できる Web ページ数が多くなる. また, リンクを収集する深さについては, Slave PC2 台では，深さを変化させても 1 台あたりの解析 Web ページ数にあまり変化は見られない．しかし，Slave PC5 台，10 台の時は深さが大きくなると 1 台あたりの解析 Web ページ数が増えた．（考察については 5.1.2 で述べる）

○ハイパーリンクとエラーページの割合

全実験データから得られた獲得リンク数を解析したページ数で割った結果，Web1 ページあたり約１２のリンクが存在していることがわかった．また Web リンクを収集する過程でエラーページに接続する割合は約５％である．

5.1.1　台数変化の影響とシステム全体のパフォーマンス

1 台あたりの解析 Web ページ数を見てみると，台数が多い程 1 台あたりの解析ページ数が小さくなっている．より長い時間でデータを取り続けた場合，台数が多い場合ほど 1 台あたりのパフォーマンスは下がると考えられる．

この原因として考えられるものに，Master が Slave に渡すタスクが関係していると考えられる．本システムでは，1 回目のデータセットは大学の TOP ページを渡しているが，2 回目以降のタスク URL はそれまでに Slave が獲得したリンクの中からランダムで選択しているため，収集している時間が長くなると同じページを解析してしまう可能性がでてくる．台数が多くなれば重複率も大きくなるので，結果に影響が現れたのではないかと考えられる．またその他の原因として，収集時間がたつにつれてストップリストなどのチェック量が増えていくので，チェックする時間分はどうしてもパフォーマンスは低下する．

5.1.2　初期値設定の変化によるパフォーマンスへの影響

タスク URL 数，リンクの深さはともに大きい値を設定する方がより多くの Web ページ数を解析できる結果となった．タスク URL 数を多く設定することは広い範囲の探索収集を行うことである．タスク URL 数とリンクの深さをともに大きく設定すると初期 URL 数に応じた非連結領域を合わせたものとして，パフォーマンスはあがる．

それでは，タスク URL 数とリンクの深さをより大きく設定すれば必ずパフォーマンスは良くなるのかというと，そうとは言い切れないだろう．複数台で長い時間をかけて収集すれば，異なる Slave が同一の連結領域を同時期に収集する可能性がでてくる．そういった収集の重なりは，重複したページを何回も探索することになるため，全体のパフォーマンスは下がることになるだろう．

5.1.3　システムパフォーマンスへのその他の影響について

実験結果の中でいくつかシステム設計・構築上の問題以外の要因が関係してパフォーマンスに影響しているものがあると考えられる．例えば 4.2.2（1）で，60 分間での Slave PC5 台と 10 台の解析ページ数がそれほど増加していない．これは，上記考察で述べた要因以外にも，収集時間帯によるネットワークトラフィックなどが関係しているのではないかと考えられる．この例以外にもいくつかネットワークトラフィックが関係していると思われる結果が得られている．今回の実験は限られた時間の中で連続して行っているので，実験を行った時間帯などに結果が影響を受けてしまったのだろう．こういった要因を除去するには，実験回数を増やし，時間帯・曜日などを考慮した実験を行うとよいと考えられる．

また，もう一つ大きな要因としては，5.2.1 で述べている，サーバの不正処理の問題がある．この問題に対しては，プログラムの中でタイマースレッドによる監視や，Thread#interrupt で割り込みをかけるなどを行っているが，最後まで不正処理を解決できなかった．そのために，システムとしてのロスがどうしてもでてしまう．

5.2　システムの問題点

本システムを構築する過程で解決できなかった技術的な問題についていくつか述べたい.

5.2.1　無反応なサーバへの接続

ネットワーク上でサーバが生きていて，80 番ポートは開いているものの，http daemon が動いていないようなサーバに接続してしまうと，Java プログラムがだいたい 10 分から 20 分程度（多くは 12 分強である．この時間は OS のデフォルトタイムのようである）止まってしまう.

この問題に対して, 例えば Java の Socket#setSoTimeout などでタイムアウトを設定しても接続は解除されない．なぜなら Timeout はサーバがネットワーク上に存在しないことを前提に行われるメソッドだからである.（システム構築過程で, URLConnection に Socket#setSoTimeout を組み込んで調べてみたが, やはりこの問題の解決にはつながらなかった.）さらに，例えば Thread クラスによって接続処理をスレッド化したとしても，スレッドが接続をあきらめるまで Thread は残ってしまう（つまり正常に終了しないことがある）．もしシステム設計上，どうしてもシステムを終了させたいときは，最終的には（あまり望ましい手段ではないが）JavaVM ごと破棄するしかない.

5.2.2　データベースの Web リンクデータ処理

本システムは DB サーバが 1 つでのため，たとえば Slave PC10 台分のデータが一度に DB サーバ送られてくるとデータの解析･格納処理に非常に時間がかかる. また, 取り扱うリンクデータが非常に膨大なため，URL データ 1 件ごとにチェック事項を行っていると非常に時間がかかってしまう. データベースの分散化やデータ処理を高速化させる画期的なアルゴリズムの導入が必要であろう.

5.2.3　CSS の解析処理

本システムで組み込んでいる HTML 解析手法では，スタイルシート（CSS）で記述された HTML ページを解析できない．最近ではスタイルシートで書かれる Web ページも増えつつあるので，解析できる手法に変える必要があるだろう．（CSS を読み込んでしまっても，すぐにエラーをはいてしまう）

5.3　より良いシステムにするには

大容量のデータ処理には複数台マシンによる並列処理が望ましい．今後，ネットワークの発展と IT の高度化により，分散コンピューティングが活発になっていく兆しがあるが，分散オブジェクトはネットワークコンピューティングをより容易に実現できる技術の一つであるといえるだろう．技術の選択の際には，開発言語とあわせての選択が重要である．本システムでの分散の仕組みは，シンプルな 1 Master－N Slave 方式を採用したが，その他の分散方式を採用し比較してみる価値はあるだろう．分散方式の比較例としては, Web Ants[21]のように 1Master－２Slave 方式を 4 組つくり，Web リンク収集させて，パフォーマンス評価を比較する，などが挙げられる[20].

DB については大容量データを扱うには欠かせないが，取り扱うデータが大きく複数から同時アクセスがあると処理パフォーマンスが落ち込み，システム全体への影響を及ぼす可能性がある．データの格納方法についても分散化や高速処理可能なアルゴリズムの適用が必要であろう．

謝辞

本研究をご指導いただきました林幸雄助教授，及び多くの研究に関するアドバイスなどをいただきました林研究室の松久保潤氏に感謝いたします．また，梅津亮氏にはシステム開発に関わる諸問題に対してご助言をいただき，この場をお借りしてお礼申し上げます.

参 考 文 献

[1] Inktomi WebMap, http://inktomi.om/webmap/

[2] 原田昌紀, WWW サーチエンジンの技術動向, 信学技報, SSE2000-228, pp.17-22, 2001.

[3] Junghoo Cho, Hector Garcia-Molina, Lawrence Page, Efficient crawling through URL ordering, Proc.of 8th World Wide Web Conf., 1999.

[4] WiseNut Search Engine White Paper, http://www.wisenut.com/corp/pdf/WiseNutWhitePaper.pdf

[5] 「Java 分散オブジェクトを使う理由」, http://www.njk.co.jp/otg/Study/Javado-wake/

[6] 平野聡, ネットワークコンピューティングの魔法のじゅうたん：HORB Flyer's ガイド, HORB ver2.0.

[7] HORB Open　HORB とは, http://www.horbopen.org/whatis.html

[8] モジュールからオブジェクトへ, http://www.njk.co.jp/otg/Study/horb/appendix.html

[9] 分散オブジェクト開発（Java Socket & HORB）, http://www.njk.co.jp/otg/Study/horb/

[10] はじめての分散オブジェクト（HORB の目指すもの）, http://njk.co.jp/otg/Study/horbintro/

[11] HORB と遊ぼう（１）～（６）.
http://www.atmarkit.co.p/fjava/rensai/horb01/
http://www.atmarkit.co.p/fjava/rensai/horb02/
http://www.atmarkit.co.p/fjava/rensai/horb03/
http://www.atmarkit.co.p/fjava/rensai/horb04/
http://www.atmarkit.co.p/fjava/rensai/horb05/
http://www.atmarkit.co.p/fjava/rensai/horb06/

[12] 中原真則，平野聡，HORB プログラミングマジック，bit 別冊
Java プログラミング例題集，p.19-40，共立出版株式会社，1997.
[13] Elliotte Rusty Harold，戸松豊和，田和勝，Java ネットワークプログラミング，p209-408，オライリー・ジャパン，2001.
[14] Mark C. Chan, Steven W. Griffith, 舟木将彦訳, Java プログラミング 1001Tips, p674-686，オーム社開発局，2001.
[15] Scott Oaks，Henry Wong，戸松豊和監訳，西村利浩訳，JAVA スレッドプログラミング，オライリー・ジャパン，1997.
[16] 石井達夫, PC UNIX ユーザのための PostgreSQL 完全攻略ガイド, 技術評論社, 2001.
[17] WEB+DB PRESS Vol.2，p2-35，技術評論社，2001.
[18] WEB+DB PRESS Vol.5，p2-23，技術評論社，2001.
[19] 朝井淳，すぐわかる SQL，技術評論社，2001.
[20] 村岡洋一ほか，Internet 広域分散強調サーチロボットの研究開発，1999 年度成果報告書，2000. http://www.etl.go.jp/~yamana/DWR/
[21] WebAnts，http://polarbear.eng.pgh.lycos.com/webants/

発 表 論 文

[1]　伊藤正敬，林幸雄，ORB・分散コンピューティングを用いた Web リンク収集，
電子情報通信学会　総合大会，早稲田大学，3 月 27，28，29 日，2002