WWW からの画像データ収集の効率化の一検討

藤原　修一†　　有次　正義†

† 群馬大学工学部情報工学科　〒 376–8515 桐生市天神町 1-5-1
E-mail: †{fujiwara,aritsugi}@dbms.cs.gunma-u.ac.jp

あらまし　WWW から容量の大きいデータを収集するには多くの時間がかかってしまうため，この処理の効率化は重要な問題である．本稿では，画像データの収集システムを考え，画像データの収集を効率良くするための手法について考察する．具体的には，WWW から画像データを収集する際に，その前処理で得られた情報を利用し，WWW からの画像データの転送コストを見積もり，それを使って負荷を分散し処理の効率化をはかる．また，そのプロトタイプシステムを使った簡単な実験を行い，基本性能を確認する．

キーワード　Web, 画像検索エンジン, 性能評価

A Consideration on Efficient Ways of Collecting Image Data from WWW

Shuichi FUJIWARA† and Masayoshi ARITSUGI†

† Department of Computer Science, Faculty of Engineering, Gunma University
1–5–1 Tenjin-cho, Kiryu, Gunma 376–8515, Japan
E-mail: †{fujiwara,aritsugi}@dbms.cs.gunma-u.ac.jp

Abstract　Since it takes long time to collect data of large size from WWW, it is desired to make this process efficient. In this paper, efficient ways of collecting image data from WWW. In the ways we intend to improve the process by making use of information obtained from pre-processing to the collecting. We try to estimate transfer rate of image data from WWW, and to reduce the total cost of processing with multi-threads. We also report some results of simple experiments and show basic performance of the ways.

Key words　Web, Image Search Engine, Performance Evaluation

1. はじめに

WWW から容量の大きなデータを大量に収集するためには多くの時間がかかる．このようなデータの収集を考える際，ボトルネックとなるのは WWW からのデータ転送時間である．また，WWW からのデータ転送時間は，データが存在するサイトによって異る．よって，WWW からのデータ転送を行う処理の効率化を考えることは重要な問題である．処理の効率化を考える際，WWW からのデータ転送コストを見積もり，負荷を分散させることによる処理の効率化が考えられる．本稿では，WWW から容量の大きなデータを収集する前に，比較的容量の小さなデータを収集し，そのときに得られた情報を利用し，容量の大きなデータの転送コストを見積もる手法を考える．それを用いて，負荷分散を行い処理の効率化を図るのが本稿の目的である．

本稿では，対象とするデータとして画像データを用い，WWW から画像データの収集を行うシステムを実装する．本システムは，画像データを収集するために 2 つの処理が含まれている．画像データの含まれている HTML 文書を取得し，その HTML タグから画像データの URL を取得する処理と，取得した URL をもとに，WWW 上から画像データを取得する処理である．この 2 つの処理のうち，画像データを収集する際にボトルネックとなるのは，後者の処理である．なぜなら HTML 文書と画像データを比較すると，一般的に画像データの容量が大きく，データ転送に時間がかかるためである．後の処理の効率化のため，前の処理でかかる HTML 文書のデータ転送時間や，取得する HTML 文書のサイズなどを利用し後の処理の効率化を考える．

画像データの転送コストの見積もりを用いて，負荷分散を行う．本システムでは，複数個の画像データの収集を並行に処理するために，マルチスレッドを用いた実装を行う．各スレッドは，画像データの URL を受け取り，画像データを取得する．見積もった画像データの転送コストを用いて，各スレッドに均等に負荷を分散させることにより，処理時間の効率化が期待で

きる．

本稿の構成は以下の通りである．2章で画像収集システムについて述べ，3章で我々の提案する画像データの転送コストを見積もる手法について説明する．4章で提案した手法を用いた場合とそうでない場合の比較実験を行い，その実験結果を報告し，最後に5章でまとめと今後の課題を述べる．

2. 画像収集システムの概要

2.1 関連研究

WWW から画像データを検索する画像検索エンジンについての研究が行われている [1] [2] [3]．これらの研究で実装されている画像検索エンジンは，ユーザが求めている画像に関係するキーワードを入力とし，画像データのリストを出力する形となる．これらの画像検索エンジンは，画像データを収集するために WWW 上を巡回するロボットを用いて，WWW の情報を収集し，取得した情報を解析することにより，インデックスを作成する．インデックスに用いるキーワードに対応する画像データとして，画像データの縮小画像であるサムネイル画像を保持している．ユーザからの問い合わせに対して，作成したインデックスを用いて検索し，そのキーワードが示す画像データのサムネイル画像のリストを検索結果として返している．

文献 [1], [2] では，HTML 文書を解析し画像データを検索する手法に加え，画像データ自身の内容に基づく画像検索を行っている．文献 [1] は，始めにキーワードで画像を検索し，その検索されたリストから，ユーザが画像を指定することにより，指定画像に類似している画像を検索する．文献 [2] は，キーワードの他に画像の大きさや，写真か図の判別，また人の顔の有無などを，ユーザが検索時に指定することにより，その指定に適した画像データを出力している．

文献 [3] では，上記の研究とは異った手法で画像データを収集している．それは，画像データが含まれている HTML 文書を取得するため，WWW ロボットを用いる代りに，既存のテキスト検索エンジンを利用している点である．テキスト検索エンジンから得られた，キーワードの検索結果を表す HTML 文書から HTML タグを解析することにより，画像データの URL を取得し，その URL の表す画像データを収集している．収集してきた画像データを，画像内容によってクラスタリングすることにより，正解画像を求めている．ゆえに WWW ロボットを利用することや，インデックスを作成し全てのキーワードに対するサムネイル画像を保持することなく，求めるキーワードの画像データを検索することができる．

2.2 システム設計

本研究では，画像データの収集システムを実装し，WWW からの画像データの収集の効率化について検討する．上記のように，WWW 上の画像データを収集する手法として，1) ロボットなどを利用して WWW の情報を収集し，インデックスを作成する手法と，2) 既存のテキスト検索エンジンを利用する手法が挙げられる．両者の違いは，1) ではロボットを利用し，WWW 上を巡回することにより，WWW 上の画像データを収集し，その情報をローカルに保存するのに対し，2) ではテキスト検索エンジンの検索結果を用いることにより，あるキーワードに対する少数の画像データのみを取得する点である．

本システムは，文献 [3] で提案されている画像収集システム Image Collector を参考に実装を行う．よって既存のテキスト検索エンジンを利用し，WWW から画像データを収集する手法を用いてシステムの実装を行う．本システムと Image Collector との異る点は，画像データの URL を取得する部分で HTML 文書の取得にかかる情報を利用することにより，画像データの転送コストを求め，その転送コストを用いて，画像データの収集の効率化を考えた点である．以下に本システムの処理手順，及びシステムの概要について説明する．

（1） テキスト検索エンジンにキーワードを与え，検索結果の HTML 文書を取得する．

（2） 得られた HTML 文書から，HTML タグを解析し (<a href=URL> から URL を取り出す) 検索結果の URL リストを作成する．

（3） URL リストの各 URL にアクセスし，画像の URL を含む HTML 文書を取得する．

（4） 取得した HTML 文書から HTML タグを解析し (<img src=URL> から URL を取り出す) 画像データの URL リストを作成する．

（5） 画像データの URL リストの各 URL にアクセスし画像データを取得する．

以上の処理手順を逐一行っていたのでは処理時間がかかる．例えば 3,4 の処理では，複数個の URL の HTML 文書を取得し，HTML タグを解析することにより画像データの URL を抽出している．5 の処理では，複数個の画像データの URL を受け取り，画像データを取得している．

よって上の処理手順を，逐次に処理を行う部分と，並行に処理が可能な部分とに分けて考えるために，3つの処理部分に分割した．1つ目はテキスト検索エンジンとのアクセス部分で 1,2 の処理を行う．2つ目は画像データの URL 取得部分で 3,4 の処理を行う．最後に画像データ取得部分で 5 の処理を行う．テキスト検索エンジンとのアクセス部分は，逐次に処理を行い，画像データの URL 取得部分と画像データ取得部分は，並行に処理を行う．

本システムでは，この2つの部分で並行処理を行うためにマルチスレッドを用いて実装を行った．図 1に本システムのシステム構成図を表す．

図 1は，データの流れと処理を行うオブジェクトを表している．図中の MainServer というオブジェクトは，テキスト検索エンジンとのアクセス部分の処理を行う．キーワードを検索エンジンに与え，キーワードに対するテキスト検索エンジンの検索結果の一覧を表示する HTML 文書を取得する．この HTML 文書には，キーワードに対する検索結果を示す URL が含まれていることから，HTML タグを解析することにより検索結果の URL を抽出し URL リストを作成する．この URL リストを ParallelServ オブジェクトにわたす．ParallelServ オブジェクトは，MainServer オブジェクトから URL リストを受け取り，URL リストを QueueA オブジェクトに挿入する処

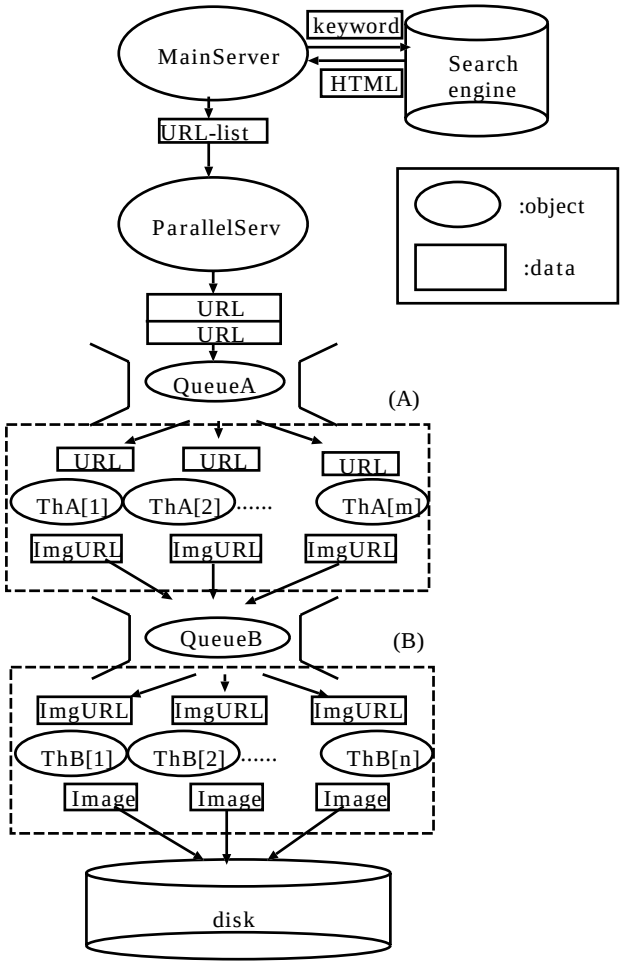

図 1　システム構成図

理を行う．

図中に表されている点線で囲まれた (A) の部分は，画像データの URL 取得部分を表し，(B) の部分は，画像データ取得部分の処理を行う．(A),(B) の部分で並行処理を行うために，マルチスレッドを用いて実装を行った．図中の (A),(B) のように，マルチスレッドを実現するためにスレッドオブジェクトを配列に格納している．

各スレッドは，処理対象の URL を取得し，処理を行い，処理が終了した後に URL を取得し，処理を開始する．(A) の処理部分で画像データの URL が得られたなら，その後に (B) の部分で処理を行う．このデータの受け渡しを，キューを介在させて行う．図 1の (B) の部分では，ThreadA オブジェクトが取得した，画像データの URL を QueueB オブジェクトに格納し，ThreadB オブジェクトがそれを QueueB オブジェクトから取得している．このとき ThreadB オブジェクトは，ThreadA オブジェクトから画像データの URL が挿入されるのを待ち，挿入された後に QueueB から画像データの URL を取得して処理を開始する．

各スレッドは，処理対象の URL を受け取り，URL が示すデータを取得する．スレッドの処理時間は，URL が示すサイトによってデータ転送時間が異る．よってスレッドが処理するのにかかる処理時間は，スレッド毎に異る．画像データの URL 取得部分と画像データ取得部分の処理時間は，最も処理時間のかかるスレッドに依存することが考えられる．よって本稿では，画像データの URL を取得する処理部分で WWW からの画像データの転送コストを見積もることにより，全てのスレッドに均等に負荷を分散させることを考える．

3.　画像データの転送コストの見積もり

WWW 上の画像データを収集するにあたりボトルネックとなるのは，画像データの転送コストである．これは，画像データが存在するサイトによって異る．本システムの画像データを収集する処理部分は，マルチスレッドを用いて実装を行っている．よって全体の処理時間は，最も処理時間のかかるスレッドに依存することが考えられる．このため，画像データの転送コストを見積もり，スレッドに均等に URL を与えることによる処理時間の効率化を考える．

具体的には，画像データの転送コストを見積もるために，画像データの含まれる HTML 文書を取得する際にかかる，データ転送時間や HTML 文書のサイズなどの情報を取得する．それを利用し転送コストの見積もりを求め，画像データの URL に転送コストを付与する．求められた全ての URL の転送コストの総和を求め，それをスレッド数で割ることにより閾値を求める．この情報を各スレッドに与えることにより，スレッドはこの値を参考に画像データの URL を取得し処理を行う．

本章では，始めに転送コストの見積もりを行わないときに用いる，ばらまき法について述べる．次に，転送コストを見積もるために画像データの URL 取得部分で取得する情報について述べ，その情報を用いた，転送コストを見積もる手法について説明する．その後，見積もった転送コストを利用した各スレッドへの画像データの振り分けについて説明し，最後に，URL を格納するキューの変更点について説明する．

3.1　ばらまき法

2 章で述べたシステムは，スレッドは処理対象の URL を取得し，その URL に対して処理を行う．各 URL に対応するデータの転送時間は，そのデータが存在するサイトによって異る．このとき無作為に一定数の URL を各スレッドが取得し，その URL に対して処理を行うのでは全体の処理時間が，最も処理時間のかかるスレッドの処理に依存することが考えられる．

スレッド間の処理時間の差を少なくするため，各スレッドは URL を 1 つずつ受け取り，その URL に対して処理を行う．スレッド間の処理時間の差は，受け取った URL のデータ転送時間の差となる．よって最も処理時間のかかるスレッドの処理に依存する時間は小さくなる．このように，URL を 1 つずつスレッドが取得する手法を，本論文では，ばらまき法という名前を用いて表現する．

3.2　取得パラメータ

画像データの URL を取得する処理部分で，以下の情報を取得する．

- HTML 文書サイズ： 検索結果の HTML 文書のサイズ．
- HTML 文書取得時間： HTML 文書を取得するのにか

かる時間.

- 画像データ URL 個数：取得した画像データの URL の全個数.
- 画像データサイズ：収集する画像データのサイズ.

画像データサイズとして，HTTP レスポンスメッセージに含まれる，Content-Length エンティティヘッダーフィールドを用いる [4]. Content-Length エンティティヘッダーフィールドを Java の URLConnection クラスのメソッドである，getContentLength [5] を用いて取得する．しかしながら，全ての Content-Length フィールドを取得していたのでは，パラメータの取得に時間がかかる．よって，画像データの URL を取得する HTML 文書のなかから 1 つの画像データの URL を取り出し，その URL の画像データサイズをその HTML 文書から取得される全ての画像データの画像データサイズとする．

3.3 転送コスト見積もり手法

本稿では，2 種類の転送コストを見積もる手法を考える．

[**手法** A] HTML 文書サイズと HTML 文書取得時間から 1 ミリ秒あたりの取得バイト数を計算し，その逆数を転送コストにする．

$$cost = \frac{1}{\frac{HTML文書サイズ}{HTML文書取得時間}}$$

[**手法** B] 手法 A の転送コストに取得する画像データサイズを掛け，転送コストとする．

$$cost = \frac{1}{\frac{HTML文書サイズ}{HTML文書取得時間}} \times 画像データサイズ$$

前章で述べた，画像データの URL 取得部分で得られた情報を利用し転送コストを見積もる．そして取得した画像データの URL に，求めた転送コストを付与する．

各スレッドの処理時間を均等にするために，スレッドが受け取る URL が示す画像データの転送コストを均等にすることで，時間のかかるスレッドの処理時間に依存することを避ける．ばらまき法と比較した場合，転送コスト見積もり手法はデータの受け渡しの回数が少なく，処理時間がかからないと考えられる．

3.4 スレッドへの URL の振り分け

この転送コストを用いて画像データ取得部分の処理の効率化を考える．全ての URL に付与された転送コストの総和を求め，総和を用いるスレッド数で割った値を閾値とし，この情報を各スレッドに渡す．各スレッドは，画像データの URL を取得し，その URL に付与されている転送コストを調べ，URL を取得する度に取得した URL の転送コストの和をとる．その和が，閾値の値を越えるまで，画像データの URL を取得し続ける．

スレッドは，閾値の値を利用することにより処理を行うわけであるが，用いる全てのスレッドが処理を行わず，処理する URL が少数のスレッドに偏ることが考えられる．この状況を避けるため，画像データの URL の全個数をスレッド数で割り，その数を越えない数の URL を，スレッドに処理させる．ゆえに各スレッドは，閾値の範囲内かもしくは制限された個数以下の URL を受け取り，受け取った URL に対して処理を行う．

以上の処理を画像収集システムに適用し，手法 A・手法 B を用いて URL を振り分けた場合と，ばらまき法を用いて URL を振り分けた場合の比較実験を 4 章で行う．

3.5 キューの変更

2 章で述べたように，各スレッドはキューから画像データの URL を取得し，その URL に対して処理を行う．本システムでは画像データ取得部分の効率化を考えるため，全ての画像データの URL を取得した後に，キューに対して処理を行っている．

画像データの URL 取得部分では，画像データの転送コストを各 URL に付与し，キュー (図 1での QueueB オブジェクト) に画像データの URL を格納する．全ての画像データの URL が格納されたなら，格納された画像データの URL を転送コストにより，昇順にソートを行う．このことにより，各スレッドは転送コストが大きい (つまり画像データを取得するのに時間がかかる)URL から順に，URL を取得することとなる．同一サイトに含まれている画像データの転送コストは，一意の値となる．画像データの転送コストを昇順にソートを行うことにより，同一サイトから取得した画像データの URL は，連続してキューの中に格納されることとなる．

ばらまき法を用いた場合も，全ての画像データの URL を取得した後に，キューに格納された画像データの URL に対して処理を行う．この場合は，キューに格納された画像データの URL を，画像データに対応する URL 名でソートを行う．同一サイトから取得した画像データの URL は，連続してキューの中に格納されることとなる．

4. 実　　験

3 章で述べた 2 種類の画像データ転送コストの見積もり手法を用いたときと，ばらまき法を用いたときとの処理時間を比較することにより，画像データの転送コストの見積もりの有効性を検討する．

4.1 実 験 環 境

本システムは Java で実装し，実験には表 1の計算機を用いた．

2 章で述べたように，本システムはテキスト検索エンジンを利用して画像データを収集する．実験では，検索エンジンとして Google を用い，数種類のキーワードを与え，各キーワード毎に一定数の検索結果を取得した．本稿では，マルチスレッドを用いて処理を行っている．処理を行うスレッド数を決定するために以下の予備実験を行った．比較的ネットワークの安定している，学内の画像データの URL を 1000 個事前に用意し，スレッド数を変化させ，そのときの処理時間によって適切なスレッド数を決定する．

図 2では横軸にスレッド数，縦軸にそのときの処理時間を表している．このとき，スレッド数が最小で，処理時間が極小値に近い値をスレッド数として選択する．図 2では，スレッド数が 46 で処理時間が極小となり，それ以降処理時間は安定する．スレッド数が最小の値を選択するため，スレッド数を 40 と決定し以降の実験を行う．

表 1　実験に用いた計算機の構成

Site	Sun Ultra30
Type	Ultra SparcII
Clock	248MHz
Memory	128MB
OS	Solaris8
Java 処理系	J2SDK1.4.0

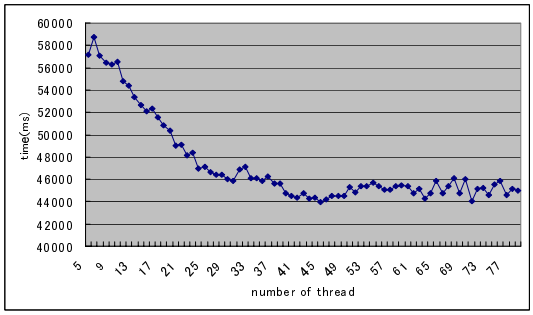

図 2　スレッド数を変化させたときの処理時間

3 章で述べた，2 種類のコストの見積もり手法を用いて，スレッドに URL を振り分けた場合と，ばらまき法を用いた場合の 3 方法の比較実験を行った．本稿では画像データの収集の処理時間の効率化を目的としているので，3 方法の画像データ取得部分でかかる処理時間を比較することによって，処理の効率化を考察する．

検索エンジンから返される検索結果の数を一定にすることより，収集する画像データの枚数はキーワードにより異る．それぞれのキーワードで収集される画像データの性質 (画像枚数，画像サイズの分布など) について示す．

4.2 実験結果

検索結果の URL 数を 50,100,200 と変化させたとき， starwars,lion,soccer,display,car,cat,porsche,music,travel,dog の 10 種類のキーワードを用いて比較実験を行った． 3 つの手法を用いるプロトタイプシステムをそれぞれ実装し，それぞれの処理時間を測定することにより， 3 方法の比較を行う．このときのグラフを図 3から図 6に示す．これらの図では，ばらまき法を用いて URL を振り分けるときの処理時間の変化を non-improve，3 章で述べた，転送コストの見積もり手法 A を用いたときの処理時間の変化を improveA，手法 B を用いたときの処理時間の変化を improveB で示した．縦軸は処理時間を表し，横軸は選択したキーワードを表している．検索結果の URL 数を一定としたとき，同じキーワードに対して， 3 方法を用いたプロトタイプシステムの処理時間を，それぞれ計測する．この処理を 5 回繰り返し，その処理時間の平均を取ることにより，処理時間の比較を行った．

検索結果の URL 数を 50 としたときの 3 方法の処理時間の比較を図 3に示す． 3 方法を比較したとき，多くの場合 improveA,improveB の方が non-improve より速くなる．しかしながらキーワードによっては処理時間にあまり差がないものが存在する．例えば， starwars,lion,soccer,porsche を選択したときの 3 方法の処理時間は，ほぼ同じ処理時間となる．

次に検索結果の URL 数を 100 としたときを図 4で表す．検索結果の URL 数を 50 としたときと同様に，多くの場合 non-improve と比較して improveA,improveB の方が速くなる．また，キーワードが soccer,travel では処理時間がほぼ同じ結果を示している．

検索結果の URL 数を 200 としたときを図 5, 図 6で表す．図 5は図 3, 図 4と同様に全てのキーワードの処理時間の変化を表しており，その中からキーワード lion を除いたときの各キーワードの処理時間の変化を図 6で表している．これらの図をみるとキーワード lion のときに， non-improve が improveA,improveB より速くなったことがわかる．

全体的に non-improve と比較して， improveA,improveB の方が速い．しかしながら，選択するキーワードによっては improveA,improveB と non-improve があまり変わらない場合が存在する．また，検索結果の URL 数が 200 で，キーワードとして lion を選択したときの処理時間は， non-improve の方が improveA,improveB と比較して速くなることが分かる．

検索結果の URL 数が 200 で，検索するキーワードが lion を選択するとき，ばらまき法を用いた場合の方が転送コストの見積もりを用いた場合より速くなる．このときの，スレッドの動作と取得する画像データのデータ転送時間を調べる．具体的には，各スレッドが取得した URL を処理する順番，その URL の処理にかかる時間を調べることにより，転送コストの見積もりを用いた場合の方が，ばらまき法より遅くなる理由について考察する．

画像データの転送コストの見積もりを用いた場合，各スレッドは見積もりを利用して，一般的には，複数個の画像データの URL を取得することになる．スレッドが取得した画像データのなかには，その URL が示す画像データを取得する際に，プロトコルエラーが発生する場合が存在する (Java ではこのとき Protocol 例外が発生する)．このような URL にアクセスした場合，スレッドの処理に多くの時間がかかる．このような URL が複数個存在し，それを 1 つのスレッドが処理する場合，このスレッドは他のスレッドと比較して処理時間がかかることになる．よって，全体の処理時間が遅くなってしまうと考えられる．これと比較して，ばらまき法を用いた場合，各スレッドは画像データの URL を 1 つずつ取得している． Protocol 例外が発生する URL が複数個存在する場合，複数のスレッドが 1 つずつ取得することとなり，上の場合より速くなることが考えられる．キーワードに lion を選択した場合，取得する画像データの URL のなかには，同一ホストから取得される，プロトコルエラーが発生する URL が複数個存在した．転送コストの見積もりを行った場合， 1 つのスレッドがこのような URL を複数個取得するため，ばらまき法より処理時間がかかることが分かった．

今回の実験では， improveA と improveB の処理時間を比較した場合，その処理時間はあまり変わらなかった．しかしながら，手法 B での計算で扱う画像データサイズは，画像データを取得するサイトの中から， 1 つの画像データのサイズを取得し，そのサイトの全ての画像データのサイズとして擬似的に見

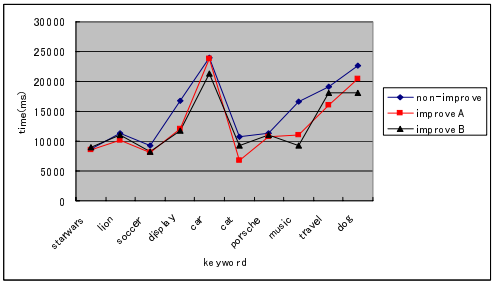

図 3　キーワードを変化させたときの処理時間 検索結果の URL 数 50

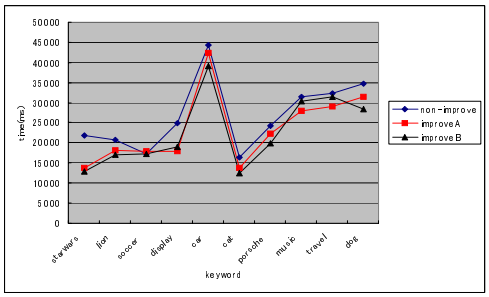

図 4　キーワードを変化させたときの処理時間 検索結果の URL 数 100

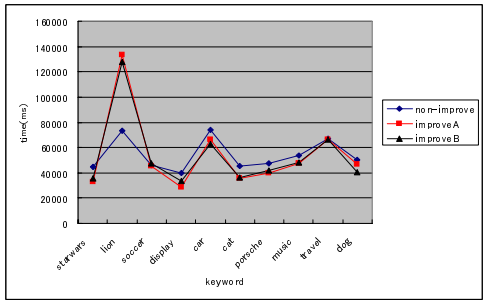

図 5　キーワードを変化させたときの処理時間 検索結果の URL 数 200

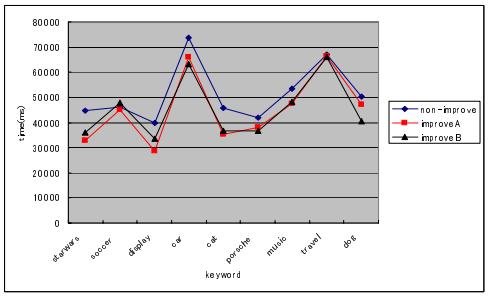

図 6　キーワードを変化させたときの処理時間 検索結果の URL 数 200

積もっている．このサイズに対してもう少し詳しい値を取得することが可能であるなら，手法 A と手法 B の結果が変わることが考えられる．

キーワードと検索結果の URL 数との，全ての組み合わせに対する，取得した画像データの総数と，その総量 (全画像データの容量の和) を表 2から表 4に示す．これらの表と図 3から図 6より，画像データ取得の処理時間と画像データの総数，総量の関係について考察する．図 3から図 6を見ると検索結果の URL 数が 200 でキーワードが lion のとき以外，キーワードに

表 2　取得した画像データの情報 検索結果の URL 数 50 枚

	starwars	lion	soccer	display	car
総数	182	287	259	288	419
総量 (kbyte)	2714	2009	902	1609	4333
	cat	porsche	music	travel	dog
総数	173	305	314	543	545
総量 (kbyte)	1004	2703	1357	1603	2475

表 3　取得した画像データの情報 検索結果の URL 数 100 枚

	starwars	lion	soccer	display	car
総数	346	407	496	423	785
総量 (kbyte)	3943	2810	1923	1972	6350
	cat	porsche	music	travel	dog
総数	427	748	926	913	846
総量 (kbyte)	2541	4358	3030	3205	3536

表 4　取得した画像データの情報 検索結果の URL 数 200 枚

	starwars	lion	soccer	display	car
総数	875	775	842	716	1338
総量 (kbyte)	7928	5188	4200	4627	10520
	cat	porsche	music	travel	dog
総数	864	989	1480	1541	1274
総量 (kbyte)	5078	8990	5343	6316	5263

car を選択したときが最も処理時間がかかることが分かる．表 2から表 4を見ると，キーワードが car のとき検索結果の URL 数に関わらず，画像データの総量が大きいことが分かる．また，キーワードが music， porsche を選択したときの，画像データ取得の処理時間と画像データの総量，総数の比較を行う．このとき，検索結果の URL 数に関わらず， music を選択したときが画像データの総数が多く， porsche を選択したときの方が画像データの総量が多いことが分かる．このときの処理時間は，キーワードに music を選択するほうが時間がかかる．このように，画像データを取得する処理時間は，画像データの総量と総数に関係していることが分かる．しかしながら，現在のところでは画像データを取得する処理時間が，総量，総数のどちらに依存するかを決定することはできない．また総量，総数以外にも処理時間に関係する要因が存在することも考えられる．よって，総量，総数以外の処理時間に関係する要因について調べると共に，画像データの情報と処理時間の関係を考察することが今後の課題である．

5.　まとめと今後の課題

WWW 上から大量に画像データを収集する際の処理時間の効率化のため，画像データの転送コストの見積もり手法を考案した．これは，画像データの含まれる HTML 文書を取得する際にかかる情報を取得し，これを用いて 2 種類の手法で転送コストの見積もりを計算する．この転送コストを用いて，スレッドが取得する URL 数を決定し，画像データを収集する部分の処理時間の効率化を図った．この転送コストの見積もり手法の有効性を検討するために比較実験を行った．

見積もり手法の有効性を検討するための比較実験により，本

稿で考案した，転送コストの見積もりを利用する手法の方が，転送コストの見積もりを利用しない手法よりも処理時間が速くなることが分かった．しかしながら，本稿で考案した 2 種類の転送コスト見積もり手法の処理時間を比較したところ，その処理時間は変わらず，どちらの転送コストの見積もりが適しているかを定めることは出来なかった．また取得する画像データのなかに，取得する際にプロトコルエラーが発生する場合が存在した．このような画像データの URL を 1 つのスレッドが複数個取得した場合，全体の処理時間がそのスレッドの処理時間に依存し，転送コストの見積もりを利用した方が，処理時間がかかる場合が存在した．

今後の課題として，スレッド間のデータの受け渡しについて検討する必要がある．各スレッドが，幾つか画像データの URL を取得し，その画像データの取得する時間が他のスレッドよりかかる場合，取得した画像データの URL を他のスレッドに分け与えることにより，全体の処理時間の効率化を考えることができる．これは上で述べた，見積もりを行った方が処理時間が遅くなる問題に対しても，適用することができる．全体の処理時間の効率化を考えた場合に，複数台の計算機を利用することが考えられる．このとき，スレッド間のデータの受け渡しの仕組みをもとに，計算機間でのデータの受け渡しに応用することも可能である．

文　　献

[1] J. Smith and S. Chang. “An Image and Video Search Engine for the World-Wide Web.” In *Storage & Retrieval for Image and Video Databases (SPIE)*, pp84–95, 1997.

[2] C. Frankel, M. Swain and V. Athitsos. “WebSeer: An Image Search Engine for the World Wide Web.” Technical Report TR-96-14, University of Chicago, http://www.cs.uchicago.edu/research/publications/techreports/TR-96-14, 1996.

[3] 柳井啓司. “キーワードと画像特徴を利用した WWW からの画像収集システム.” 情報処理学会論文誌: データベース, Vol.42, No.SIG10(TOD11), pp79–91, 2001.

[4] Hypertext Transfer Protocol – HTTP/1.1. http://www.ietf.org/rfc/rfc2068.txt

[5] Java 2 Platform, Standard Edition, 1.4.0, API 仕様. http://java.sun.com/j2se/1.4/ja/docs/ja/api/index.html