Webデザイン特別プログラム教育のための小規模サーチエンジンの試作

Trial Development of Small-scale Search Engine to Educate Undergraduate Students in "Web Design" Special Program

幸谷智紀* 竹口友大**

Tomonori KOUYA* Tomohiro TAKEGUCHI**

Abstract: In April of 2009, we will start to educate "Web Design" as a important part of some "Special Programs" in newly started "Faculty of Comprehensive Informatics." In this program, we will reserve some amount of time to spend on learning scripts languages like PHP and Perl and SQL to manipulate databases by project-based learning method. Lack of educational materials for these purposes is currently making us develop such materials. In this paper, we describe brief summary of our small-scale search engine as one of educational materials for "Web design" special program.

1. 初めに

静岡理工科大学総合情報学部では2009年度から特別プログラムの一つとして「Webデザイン」を据えることを決定している[4]。いわゆるProject-Based Learning(PBL)の手法を用いて実施されるもので，総合情報学部に属する意欲あふれる学部2年生20名程度を対象に，World Wide Web(以下Webと略記)の技術全般を，実践を通じて習得することを目的とするものである。現在の計画では一年間に渡って168コマ(1コマ=90分)を確保し，5〜6名の教員がWebに関する技術の習得と制作物の評価を行うことを予定しており，我々はこの中で「データベースとWeb」，即ちDBMSとWebを組み合わせて利用する3層Webシステムを構築する時間を担当することになっている。計画当初はこの時間内にサーチエンジン全体を構築する実習を行う予定であったが，その後の担当教員グループでの話し合いの結果，あくまでその基礎部分を教えるに留めたほうがよいということになった。従ってサーチエンジンの構築そのものは，意欲と能力のある学生向けのオプション教材，および卒業研究のための教材という位置づけである。

教育としてサーチエンジンを構築する試みはテネシー大学で既に行われコースウェアが出版されている[3]が，これは線型計算と情報技術との橋渡しを目指したものであるのに対し，我々の目指すのは3層Webプログラミング技術の応用としてのサーチエンジンシステムである。従って，基本的には既存のスクリプト言語の習得と，そこからSQLを用いてDBMSを操作するためのプログラミング実習が主となるが，それらを予備知識のない学生に習得させるための教材は現在適切なものが存在しない。そのため，我々は学内LANの環境でも設置可能な小規模サーチエンジンシステムを作り，それを段階を追って技術を習得しながら作り上げるための教材[2]もあわせて執筆した。2007年度の幸谷研究室における情報セミナーIIではこれを用いて3層Webプログラミングの技術習得を行っている。最終的にはこのシステムを様々な用途に応用していくことを予定しており，従って，用途に合わせた実践的なサーチエンジンシステムとその技術を学ぶための教材開発は今後も継続していかねばならない。そのためには，プロトタイプ版でも現状の商用サーチエンジンの基本機能を持つものを作り上げ，その性能評価を行ってシステムの限界を知っておく必要がある。

本稿ではまずこのサーチエンジンシステムを構築するに至った経緯を説明する。次に現状の商用サーチエンジンシステムの例としてGoogleの機能を解説する。そしてこれを土台にして構築した我々の小規模サーチエンジンシステムの解説を行い，その性能評価を行った結果を示す。最後に，まとめと今後の課題を提示して，次年度以降の開発計画への指針としたい。

2. Webデザイン特別プログラムと3層Webプログラミング教育

Webデザイン特別プログラムは総合情報学部に属する5名の教員（菅沼・宮岡・金久保・手島・幸谷）に非常勤講師を加えた6名で担当し，主として人間情報デザイン学科のWebデザインコースに進む2年生が受講することを想定している。

このプログラムの目的は「Webデザイン」を総合的に学ぶことである。近年のWebは静的なHTMLファイルによるWebページだけでは語れず，動画ファイルやFlashなどを用いて視覚効果を高めたり，バックエンドにデータベースを用いる3層Webシステムを用いることが当たり前になっている。当然，TCP/IP Protocol Suiteを土台としたWebサーバシステムの運用についてもそれなりの知識が必要となる。このような現代のWebを成立させるためのIT技術を総合的に学び，加えて狭義のデザイン能力とそこから受ける心理的効果についても学び，身に付けることが本プログラムの目的である。

このように様々な異分野の技術や知識を学ぶため，一年間に渡り，毎週6コマを使い，Project Based Learningの手法を用いて，理論と実践を交互に組み合わせながらFig.1に示すような順序で総合的なWebデザイン能力を磨いていく。

我々はこのうち「データベースとWeb」36コマ分を担当する。この部分のシラバス案の詳細をFig.2に示す。ご覧の通り，この中では3層Webプログラミング技術を中心に学んでいくことになる。

ユーザが直接操作するWebブラウザ，Webサーバ，バックエンドのDataBase Management System(DBMS)の3層から成るWebシステムを構築するためのプログラミングを，本稿では「3層Webプログラミング」と呼ぶことにする。この歴史は古く，CGIはWeb草創期には利用されていたので，それと同じ時期には考案されていたのではないかと推察される。既に1996年にはまとまった邦書[5]が出版されているが，10年以上経った現在でも，フリーなDBMSが主流になってきたと

2008年3月4日受理

*理工学部情報システム学科

**(資) わいにじ

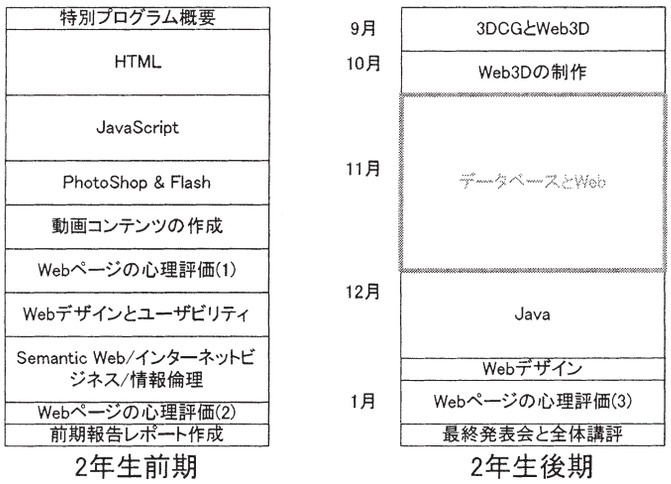

Fig. 1: Web デザイン特別プログラムのシラバス案

いう以外の技術の変化はない。それだけ「枯れた」技術と言えよう。この枯れた技術の現代的応用としてはサーチエンジンというものが面白いのではないかと我々は考えている。

しかし，現状では学生のプログラミング技術の習得状況は甚だ怪しく，プログラミング以外の関連事項も含め，一通りのことを PBL の中で教え込む必要があると思われる。以下，3 層 Web プログラミング教育に必要な重点事項を述べる。

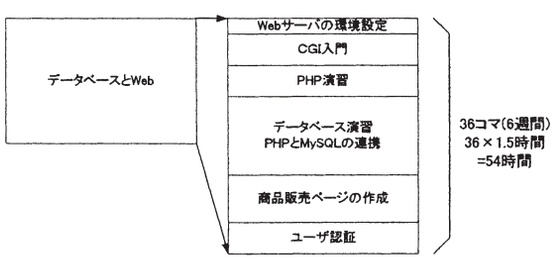

Fig. 2:「データベースと Web」のシラバス案

2.1 Web サーバの運用技術

インターネットにさらされる環境で運用することを前提とした，Web サーバの運用技術の習得は必須と考えている。最近では ISP が用意するお仕着せの環境で CGI を動かすことが当たり前のことになっているが，じっくり時間をかけて学ぶことのできる機会に，システムの根本に触れておくことは無駄ではない。また，仮想マシン環境が一般化してきている昨今の現状を考えると，管理者 (root) としてサーバに触れる機会は逆に増えてくるとも考えられる。

そこで，昨年整備した VMware 環境[6] を利用し，OS(CentOS) のセットアップから Web サーバの環境設定まで全てを学生個人で行ってもらい，その環境の上で全ての実習を行うようにしようと考えている。Web サーバは OS に標準で搭載されている Apache を使用し，httpd daemon の起動と停止，httpd.conf の設定変更，アクセスログの読み方とログ解析ツールの利用法等を習得させる。UNIX の CUI に不慣れな学生が殆どだが，我々の経験上，GNOME のような GUI も併用すれば，実習に支障をきたすことはないと思われる。

2.2 Perl/PHP の知識

3 層 Web プログラミングは，スクリプト言語だけではなく，Java や C/C++といったコンパイラ言語でも可能である。最近ではむしろ後者，特に Java による実装が増えているように思われる。Multi-core CPU が増えつつある昨今では，Multi-thread 化の容易な言語で実装した方がパフォーマンスの点で有利になることが多いだろう。

しかし，Perl による CGI の実装は長い間標準として使用されてきており，現在でも Movable Type[8] のような主要な Web アプリケーションでも Perl を利用している例は少なくない。また，後述するように，我々の実装した Web ロボットは，長年の実績を積んできた Perl Module(libwww，Encode 等) を多数利用し，行数を抑えている。将来的には Java 等のコンパイラ言語への移行も考える必要があるが，現状，我々が持っているネットワーク資源を考えると，Perl で十分という面がある。

そこで我々は，Perl による CGI やデータベースプログラミングを教えると共に，並行して PHP についても触れる予定でいる。言語に関わらず，データベースの操作は SQL を発行してデータを操作するだけ，という基本を知る意味でも複数の言語に触れる機会があった方が良い。また，Perl に似ていながら可読性の良いスクリプトが書け，HTML に埋め込む形で容易に改変が可能な言語が存在することも知っておいた方がいい。

2.3 データベース (SQL) の基礎知識

コンピュータを専門とする大学学部において，データベースを講義しないところは皆無であろう。本学でも開講されているが，受講した学生の様子を見ると，ほとんど身についていないというのが現状である。これは担当教員の責任というよりは，座学として RDB 理論を教えるという教育体制自体に無理があると思われる。実際，履修した学生に聞いてみると，RDB の操作は集合とその演算として考えられるという基本は理解しているようだが，それを SQL 文として書き下して活用することには繋がっていないようである。

従って，じっくり実習期間が確保できる PBL こそ，データベース技術の習得にふさわしい場と言える。前述したように，

データベースと Web の連携は枯れた技術であるが，現在のようにフリーで使用できる DBMS が存在し，外部ストレージが Tera バイト単位で扱えるようになった昨今でこそ，その真価を個人で体験できるようになったと言える。

特に今回実装するサーチエンジンでは，扱うデータ量は膨大になる。また，昨日までアクセスできていた Web サイトが今日は存在しない，ということも日常茶飯事である。収集したデータの更新・削除を高速におこなうためには，信頼性の高いトランザクション処理が可能な DBMS が欠かせない。

従って，我々の教育では DBMS を扱うためのごく基本的な SQL 文の使い方だけを扱うようにし，データの更新・削除・検索を行う 1 行 SQL 文が書き下せる水準に達すればよしとする。

3. Google の検索機能

以上述べてきたように，我々の開発するサーチエンジンは 3 層 Web プログラミング技法を学ぶための教材として適切な規模のものである必要がある。また，ネットワークやコンピュータリソースの限られた環境で動作させるため，集められる URI 数は自ずと限られてくる。従って，サーチエンジンとしては行数も URI 数も必然的に小規模なものとなる。

とはいえ，小規模ながらも Web デザイン特別プログラム教育の一環として使用される以上，現代的なサーチエンジンの基本機能は備えていなければならない。そこで，世界的に最も成功している Google の機能の主要な機能を備えたものを目指すことにする。

Google の成功は，特に PageRank と呼ばれる被リンク数に基づく URI のランク付け機能によってもたらされたと言われている [10]。しかし，最近では PageRank に加えて，検索キーワードや URI 間の関連性を柔軟に取り入れた機構も導入されているように見受けられる。ここでは PageRank による URI のランク付けの機構と，表面的に観察できるそれ以外の機構について述べる。

3.1 PageRank による重みづけ

PageRank に関しては既に Web 上 [10] や優れた書籍 [12] による解説が溢れているので，ごく簡単な解説に留める。基本的には Fig.3 に示すようにして，被リンク関係に基づいた確率遷移行列 H を導出することになる。

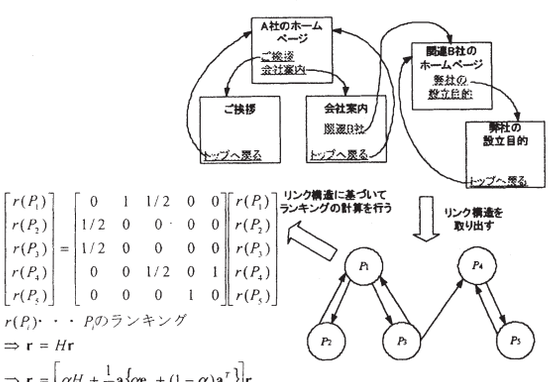

Fig. 3: PageRank 計算のための確率遷移行列の導出

この際，Tree 構造の末端，ループ構造もあり得るので，適度なランダム移動成分を加味して次のような処理を行って Google 行列 G を導出する。

1. H の列成分が全てゼロの列を $1/n$ で置き換えた S を生成する。もし i 列目が全てゼロであれば:

$$S = H + \frac{1}{n}\mathbf{e}\mathbf{e}_i \tag{1}$$

となる。ここで $\mathbf{e}_i$ は単位ベクトルであり，$\mathbf{e} = \sum_{i=1}^{n} \mathbf{e}_i$ である。

2. $0 < \alpha < 1$ という定数 α を決めておき，Google 行列 G を

$$G = \alpha S + \frac{1-\alpha}{n}\mathbf{e}\mathbf{e}^T \tag{2}$$

$$= \alpha H + \frac{1}{n}\mathbf{e}\left(\alpha \mathbf{e}_i^T + (1-\alpha)\mathbf{e}^T\right) \tag{3}$$

とする。

Google 関係者による示唆によれば，$\alpha = 0.85$ が Google 推奨値ということである [12]。数学的にはこの G の最大固有値 $\lambda_1(G) = 1$ に対応する固有ベクトル $\mathbf{r}$ が PageRank 値となる。これを求める数値計算アルゴリズムとしては，行列ベクトル積計算とノルムによる正規化計算のみから成るべき乗法 [13] を用いるのが適切である。最も計算量を必要とする行列ベクトル積では，H がランダム疎行列であることを考慮すると劇的に計算性能が上がる。従って，(3) 式に基づいて行列ベクトル積の計算を行うことが望ましい。

3.2 URI 間の関連性と検索キーワードによる小規模な順序変更

基本的に，Google では PageRank に基づいて検索結果の表示順を決めていることになっているが，検索キーワードや URI によっては局所的にそれが破られることもある。

例えば検索キーワードとして「まんねん貧乏」と「得能史子」を与えた時の検索結果を Fig.4 に示す。

Amazon の URI と，「うだうだ Weblog」の URI の表示順が，検索キーワードによって入れ替えられていることが分かる。厳密に PageRank だけで表示順を決めているのであれば，局所的とはいえ，このような順序変更は起こらない筈である。また，インデントを付けることで関連が強いと判断された URI の引っ張り上げも行われており，これは意図的に PageRank を無視して実行されている。

しかし，このような観測を行っても表示順の詳細は不明なままである。我々のサーチエンジンにおいても，PageRank 以外の要素をどのように使用してユーザの望む順に表示するかはまだ詰め切れていない。従って，現時点ではこの機能は実装していない。

4. サーチエンジンシステムの概要

以上のような Google の機能に基づき，最終的に作成したサーチエンジンシステムは Fig.5 のような構成になった。

教育用のシステムであるため，最小限の 3 つのコンポーネント構成で実現できるようにしてある。各コンポーネントは次のような機能を持つスクリプトもしくはソフトウェアを使用している。

DBMS フリーで使用できる MySQL[11] を採用する。収集した全てのデータはここに格納される。

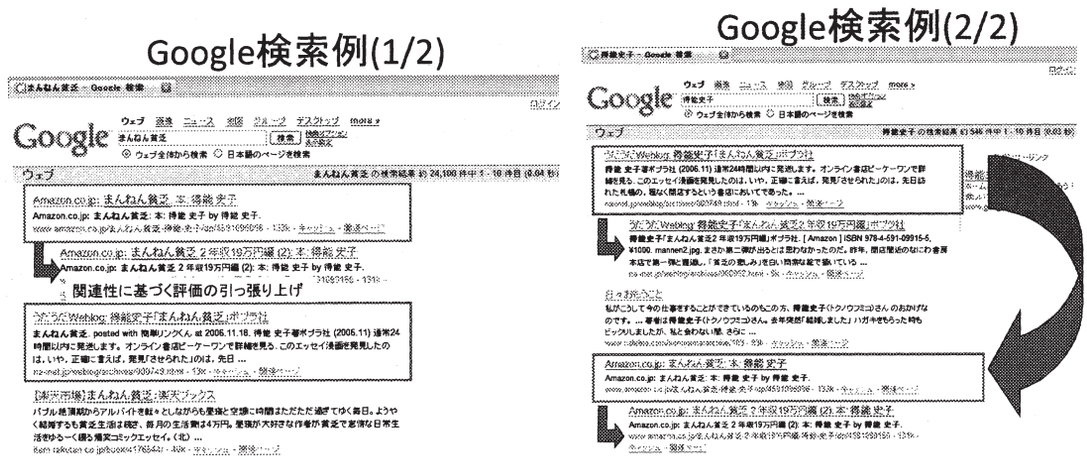

Fig. 4: Google 検索例— 検索キーワードによる順序変更 (左:「まんねん貧乏」, 右:「得能史子」)

検索用 IF 現状ではまだ仕様が固まっていないが，最終的には PHP スクリプト (+ JavaScript?) による実装を予定している。検索結果は後述するランキングの高い順に表示されるようになる。

Web ロボット Perl による Web データ収集プログラムで，現在のところ 400 行程度の規模である。robots.txt による収集制御に対応している。

図では DBMS，検索用 IF，Web ロボットを別マシンで動作させているが，実習ではこれらを 1 台の仮想マシン上で動作させることになる。

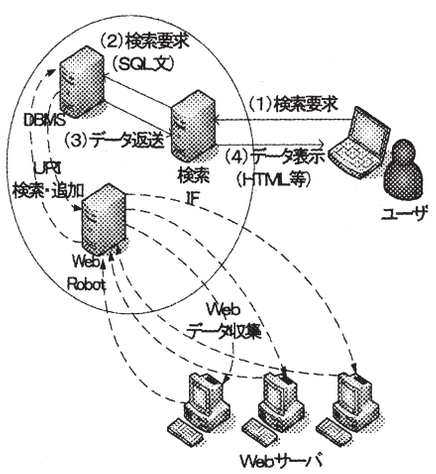

Fig. 5: サーチエンジンのシステム

これらのうち，特に重要な機能である Web ロボットと，ランキング計算部分について以下に述べる。

4.1 Web ロボット

Web ロボットの基本動作は

1. 既知の URI にアクセスし，テキストデータのみを取得
2. 取得したテキストデータ (HTML など) から新たな URI を取得 ⇒ 1 へ

というきわめて単純なものであるが，一歩間違えると悪質なアタックツールにもなりかねない危険な側面を持つ。そのため，次の機能を持つことが，安全な Web ロボットの運用には不可欠である。

- 同一 Web サーバにアクセスする際には，必ず十数秒以上の間隔をあける。
- robots.txt[9] によるアクセス制御に従う。

更に，新たな URI 取得のためには meta タグやアンカーを適切に読み取る機能が欠かせない。このためには HTML Parser が必要である。また，検索のためには Unicode に対応したエンコード機能を持ち，格納するテキストデータのエンコード方式を統一しなければならない。これから増えると思われるオープンドキュメント形式のオフィス文書ファイルも取り込むためには圧縮ファイルへの対応も求められるだろう。

現在のところ，これら全ての機能を満足する機能を備えた言語環境はそう多くない。そのため，パフォーマンスの向上に関しては難はあるものの，今のところは Perl とそのモジュール群を利用して Web ロボットを実装してある。この結果，スクリプトサイズも 500 行に満たない程度に抑えられている。

それでも学生は慣れない言語とモジュール群を活用しなければならないため，実際の PBL を行うに際しては配慮が必要となる。まず Perl 言語の解説を行い，次に DBI モジュールの使い方を通じて MySQL の操作手法に慣れ，その後に Encode, LWP, RobotUA, HTML Parser モジュールの機能を，短いスクリプトを作り動かすという経験を通じて理解させるようにする。慣れない言語と高度なネットワークアプリケーションの機能に親しむには，このように段階を追って少しずつ Web ロボット全体の機能が把握できるようにする配慮が必要であろう。

4.2 ランキング計算処理 — BNCsparse ライブラリの開発

前述したように，ランキングは Google 行列 G をべき乗法にかけることで得られる。我々のサーチエンジンではこのランキングを WebRank と呼ぶことにした。

このWebRankの計算を行うには，ランダム疎行列ライブラリを用いることが計算時間とメモリの節約のためには必須である。既にそのような疎行列ライブラリは様々なものが存在するので，既存のものを使うことも充分可能であるが，我々はスクラッチから作り上げたBNCsparseライブラリを用いることにした。

新たに作った理由は次の通りである。

1. べき乗法で使用する計算は次のような行列ベクトル積である。

$$\begin{aligned}\mathbf{r}_{k+1} &:= G\mathbf{r}_k \\ &= \alpha H\mathbf{r}_k + \frac{1}{n}\mathbf{e}\left(\alpha\mathbf{e}_i^T + (1-\alpha)\mathbf{e}^T\right)\mathbf{r}_k\end{aligned}$$

これと基本ベクトル演算だけをサポートした最小限の機能のもので十分なため，最小のコード量(500行程度)で実装できる。

2. 後述するが，ベンチマークテストの結果，WebRank計算よりもその前処理部分に時間がかかっており，あまり高速性は必要ないという現状がある。能力が足りないということが判明してから，より高速な既存のライブラリを用いても遅くない。
3. WebRankによる順位付けが目的なので，各URIにはWebRank値に基づいた整数値を与えておけば良い。こうすることで，浮動小数点数を用いるよりもデータベースのアクセスを高速化することができる。
4. チューニングの必要があまりないので，ANSI Cの範囲で実装すればよい。こうすることでLinuxとWindowsの両方で動作するものができた。

BNCsparseライブラリにおけるランダムスパース行列は，下記のような構造体で実装されている。これはCRS(Compressed Row Storage)方式を基本とし，allゼロの列構造も保持するものになっている。

```
typedef struct {
  // Elements of matrix
  double *element;
  // Dimensions of Row and Column
  long int row_dim, col_dim;
  // Indeces of Non-zero elements
  long int **nzero_index;
  // Numbers of non-zero elements in i-th row
  long int *nzero_col_dim;
  // Numbers of non-zero elements in i-th column
  long int *nzero_row_dim;
  // Total number of non-zero elements
  long int nzero_total_num;
} drsmatrix;
```

例えば，次のようなランダム疎行列Hを格納するとする。

$$H = \begin{bmatrix} 0 & 1 & 1/2 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 0 \\ 1/2 & 1 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

この時，構造体には以下のような形でデータが格納されることになる。

```
element = [1 1/2 1/2 1/2 1 1]
nzero_total_num = 7
row_dim = 5,col_dim = 5
nzero_index[0] = [1 2]
nzero_index[1] = [0]
nzero_index[2] = [0]
nzero_index[3] = [2 3]
nzero_index[4] = [3]
nzero_col_dim[0] = 2
nzero_col_dim[1] = 1
nzero_col_dim[2] = 1
nzero_col_dim[3] = 2
nzero_col_dim[4] = 1
nzero_row_dim[0] = 2
nzero_row_dim[1] = 1
nzero_row_dim[2] = 2
nzero_row_dim[3] = 1
nzero_row_dim[4] = 1
```

BNCsparseライブラリを用いてWebRankを計算する手順は次のようになる。

1. テーブルからテキストファイルにリンク構造を書き出し
2. ベキ乗法に基づくWebRank計算（固有ベクトルが欲しい)
3. WebRank値をテーブルに書き戻し

こうしてある頻度で収集されたURIデータベースにWebRankを書き込んでおくことで，検索結果はこの大きい方から先に表示できるようになる(Fig.6)。

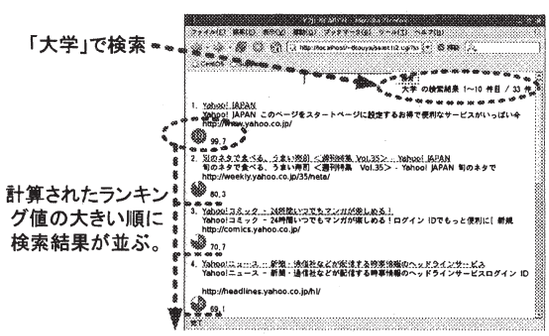

Fig. 6: 検索結果の表示(「大学」で検索)

5. ベンチマーク結果

以上のような機能を備えたサーチエンジンシステムを動作させた結果をここでは示す。Fig.5ではデータベース，検索IF，WebRobotが別のマシン上で動作しているが，ここでの実験結果は特に断らない限り，一台のPCで行ったものである。

サーチエンジンのベンチマークは環境によって大きく異なる。まず外部回線容量がある程度大きくないとそこがボトルネックになってしまう。本学のように，堅固なFirewallで守られた組織内LANから実験を行う際にはFirewallや内部Proxyの性能も関係してくる。そしてWebRobot内の処理を並列化することで能力向上を目指そうとすると，今度は2以上のCPU数あるいはCore数が必要となる。更に全てのデータはデータベースに蓄えられるので，データ量が増えるに従いI/O性能

はより高いものが求められる。どの要素もサーチエンジン性能に直結するものであるから，本来ならばそれらを全てチェックしなければならないが，現状では用意できる環境には制約があるので，その範囲内で出来ることを行ってみたのが以下に述べる結果である。

5.1 本学内 LAN からの URI 収集と WebRank 計算結果

まず静岡理工科大学研究実験棟 LAN から URI 収集を行った結果 (Fig.7) と，それに対して WebRank 計算を行った結果を示す。

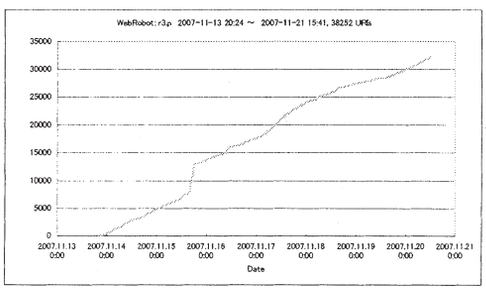

Fig. 7: 学内 LAN 内からの URI 収集数

条件は以下の通りである。

- 静岡理工科大学 LAN からアクセスしたので，Firewall と研究実習棟 Proxy の 2 段の Proxy を介して外部 7Mbps 回線を用いたことになる。
- 使用ハードウェア・ソフトウェア環境は次の通り
  CPU Intel Pentium 4 (1.8GHz)
  RAM 1GB
  HDD 40GB
  OS CentOS 5 x86_32 版
- WebRobot(r3.pl) は前述したように Perl スクリプトとして実装し，1 プロセスのみ起動。完全なシリアル動作である。
- Yahoo! Japan のトップページ (`http://www.yahoo.co.jp/`) からスタートした。後述するように，現状の WebRobot の収集アルゴリズムは深さ優先になっているため，殆どの URI は Yahoo! Japan サイト内のものとなってしまっている。
- 1URI にアクセスするごとに 10 秒の wait を置くようにした。
- robots.txt の指示に従ってアクセス制限をしているので，意図的に WebRobot を拒否しているサイトの情報は収集できない。
- テーブルに記憶する URI は接続確認されたもののみである。アクセスできなかったものは記憶しない。

この結果，一週間で約 3 万 URI が収集できた (Fig.7)。更に収集を行い，最終的には約 8 万 URI 集めるに至った。これらのリンク構造 (行列 H) を Fig.8 に示す。黒点は非ゼロ要素があることを示している。

前述したように，3 万 URI でも 8 万 URI でも殆どが Yahoo! Japan サイト内の URI なので，行列の下三角部分には殆どリンクがない。これは発見した順に URI に固有 ID 番号を振っているためだと思われる。

これら 2 つのリンク構造に対して WebRank 計算した結果を次に示す。WebRank 計算のためのべき乗法の収束条件は次のように設定した。

$$\|\mathbf{r}_k - \mathbf{r}_{k-1}\|_1 \leq 10^{-5}\|\mathbf{r}_{k-1}\|_1 + 10^{-50} \tag{4}$$

高速な CPU を持つマシン (Intel Core2Duo 2.16GHz, gcc 4.0.1 の場合) で WebRank を計算した結果は次の通りである。

1. テキストファイルからの読み込み
   38243URIs: 0.71 sec
   81305URIs: 1.78 sec
2. べキ乗法の計算時間
   38243URIs: 0.09 sec (11 回反復, $\alpha = 0.85$)
   81305URIs: 0.30 sec (16 回反復, $\alpha = 0.85$)

現状では，データベースからリンク構造を引っ張り出してファイルに書き出し，それを読み取って J を構築している。従って，テキストファイルの読み込み時間とべキ乗法部分の計算時間の和が WebRank 計算全体の時間を規定していることになる。この結果によって，テキストファイルからの読み時間はべキ乗法の計算時間に比べて約 6 倍〜7 倍かかっていることが分かる。よって，べキ乗法部分をこれ以上高速化しても効果薄であり，それよりは I/O 性能を上げる方が良い。

5.2 URI 収集性能

更なる高速化を目指すため，URI 収集においてボトルネックになりそうな部分の改善を行って，実験を行った。改良したのは以下の 3 点である。

- ネットワークをより高速な外部回線 (ベストエフォート 100Mbps) を持つ所に変更
- WebRobot の処理をマルチプロセス化し，並列動作を行うようにした
- Cache メモリを搭載した RAID カードを用いて SATA HDD を RAID0 構成にし，ローカルストレージの性能向上を図った

このうち，マルチプロセス化による並列動作と，RAID 構成を用いた改善の概要について以下で解説を行う。

5.2.1 マルチプロセス化による並列動作

マルチプロセス化した WebRobot の動作概念図を Fig.9 に示す。

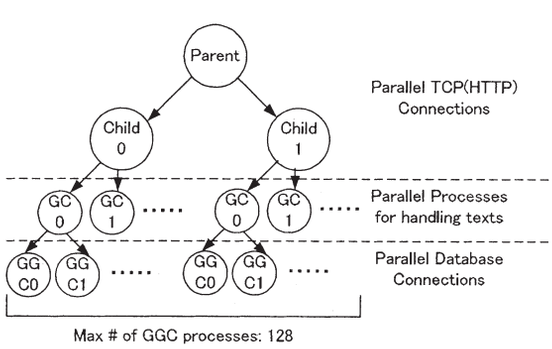

Fig. 9: Web Robot 処理の並列化

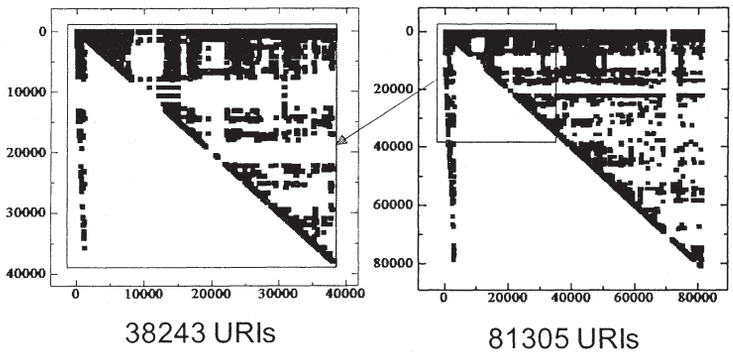

Fig. 8: H の構造

親 (Parent) プロセスは子 (Child) プロセスを fork を使って二つ起動した後はプログラム終了まで何もしない。子 (Child) プロセスは Web からのデータ収集を行うための TCP 接続 (HTTP) を担当し，接続が完了してから孫 (GC, Grandchild) プロセスを生成する。孫プロセスはひ孫 (GGC,Grand Grandchild) プロセスを生成し，テキスト処理を行った後，すぐに終了する。ひ孫プロセスはデータベースとの入出力を担当し，処理を終えた後はすぐに消えるようになっている。

全体としては親からひ孫プロセスを含めた合計数が上限値 (128) に達するまでは孫およびひ孫プロセスが生成され続ける。もし上限に達した場合は，子プロセスが 10 秒間 sleep した後，上限を下回っていることを確認できれば再度孫プロセスを fork する。

このような多段 fork によってプロセスを生成するようにしたのは，子プロセス以下のプロセスをゾンビ化 (親プロセスへのリンクを失った状態) させないためである。末端のひ孫プロセスは init が預かるのでゾンビ化することはない。

一般に，近年はプロセスよりはスレッドが並列動作のために使用されることが多いが，ここではスレッドもプロセスも同じ実体として実装されている Linux を用いているので，枯れた技術である fork によるマルチプロセス構成を採用した。これによって，より大量の URI を収集することが可能になる効用が期待できる。

また，並列動作の性能向上を確実化するために，Quad-core CPU を搭載した次のようなマシン上で動作実験を行った。

CPU Intel Core2Quad 6600
RAM 4GB
HDD 250GB
OS CentOS 5 x86_64 版

5.2.2 RAID カードによるローカルストレージの高速化

これに加えて，I/O 性能の向上を図るため，大容量の Cache メモリを搭載した RAID ボードを上記のマシンに乗せ，RAID0 の機能を用いた。この結果，I/O 性能のボトルネックが解消され，より大量の URI が収集できる可能性が増える。

以上のような環境で URI 収集数の変化 (1 週間分) を Fig.10 に示す。比較のため，RAID ボードなし (No RAID0) の結果も載せてある。

RAID 機能を用いた場合は使用しない場合に比べ，最終的には 1.5 倍の URI 数の収集が可能になっていることが分かる。

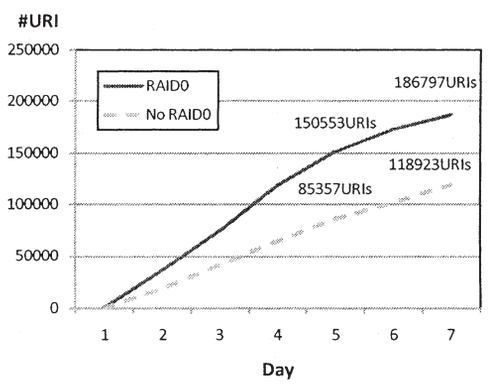

Fig. 10: 100Mbps 回線からの URI 収集数

なお，このグラフからはよく分からないが，RAID0 を使用した結果，ローカル側のルータがハングアップし，URI 数の伸びが若干削がれてしまっている。そのため，より高性能のルータを用いることにより，更なる性能向上が見込めることが予想できる。

6. まとめと今後の課題

以上，設置予定の Web デザイン特別プログラムの紹介，3 層 Web プログラミング教育の必要性とその応用としてのサーチエンジン構築の意義，そして教材として使用する予定のサーチエンジンシステムの性能評価を述べてきた。今後の課題はまず PBL の実践を行うことに尽きるわけだが，その際には以下のことをもう少し詰める必要があると我々は考えている。

教育課題 3 年生向け実験講座での教育実践の結果，PBL としても我々の目指すレベルは相当高く，学生にとっては厳しいものになると思われる。もちろん途中でドロップアウトさせないように精神的にも学習面でも注意深く指導する必要がある。そのためには，充実した内容の印刷物のテキスト [2] が不可欠である。これをを次年度中に是非とも完成させたいと考えている。

研究課題 サーチエンジンシステム自体は，人間とインターネットとの接点となる不可欠のツールとなっており，学術研究課題としてもかなり面白いテーマであると考えている。現在のところ，次のことについて，このPBLで作成したサーチエンジンによる知見を元に研究テーマとして発展させられないか，と思案中である。

- ランキング計算の高速化・安定化と，ユーザ要求に応じたランキングの変動機能
- 日本語形態素解析を利用した検索要求部の改良

応用課題 我々が用意できるネットワークリソースを考えると，とてもGoogleに対抗できるようなサーチエンジンシステムを作り上げることはできない。そこで，このPBLで構築したシステムを限られたネットワークエリア内の組織内ツールとして運用することを思案中である。その際には，グループウェアに取り入れられているような，組織にジャストフィットするようにランキングやWebロボットをカスタマイズするための機構を入れる必要があるだろう。

また，外部回線さえ高速にできれば，更なる大量のURIが収集できることが判明した。多重にマシンを並べ，更なる高速化をローカルルータの限界まで行うことができれば，データマイニングのための大量のテキストデータ収集が行える。

謝辞

本研究は静岡理工科大学特別研究費の支援を受けて行われた。関係者一同に厚く御礼申し上げる。

参考文献

1) 竹口友大・幸谷智紀, ランク機能付きサーチエンジンの開発およびI/Oボトルネック対策, 第70回情報処理学会全国大会講演集, 2008.
2) 幸谷智紀・竹口友大, “サーチエンジンを作ろう”, 未公開テキスト.
3) M.W.Berry, M.Browne, Understanding Search Engine, SIAM, 1999.
4) Webデザイン特別プログラムの紹介, http://ex-cs.sist.ac.jp/~suganuma/dep/PBL/PBL.html
5) 日経データプロ編, WWW-データベース連携システム構築法, 日経BP社, 1996.
6) 幸谷智紀, VMwareを用いたPC clusterとPC教室との共存実験, 平成18年度情報教育研究集会論文集, pp.507 – 501, 2006.
7) The Community ENTerprise Operating System, http://www.centos.org/
8) Movable Type, http://www.movabletype.com/
9) http://www.robotstxt.org/
10) Google の秘密 - PageRank 徹底解説, http://www.kusastro.kyoto-u.ac.jp/~baba/wais/pagerank.html
11) MySQL, http://www.mysql.com/
12) A.N.Langville, C.D.Meyer, Google's PageRank and Beyond, Princeton University Press, 2006.
13) 幸谷智紀, ソフトウェアとしての数値計算, http://na-inet.jp/nasoft/
14) 山本達文, ”PHP+MySQLと茶筌(Chasen)を用いたサーチエンジンの作成”, 2007年度静岡理工科大学情報システム学科卒業研究.
15) 伊藤裕晃, ”サーチエンジンのための検索順序づけ（Rank付け）に関する研究”, 2007年度静岡理工科大学情報システム学科卒業研究.