小規模 PC cluster を用いた並列分散サーチエンジンシステムについて

On Distributed and Parallelized Web Search Engine System using Small-sized PC clusters

遠山　瞬*　幸谷智紀*

Shun TOHYAMA* and Tomonori KOUYA*

Abstract: We have been implementing a small-sized MeCab Search Engine system based on free RDBMS like MySQL and on PC clusters since 2003. This paper will explain the purpose of our search engine system and show the rich store of our empirical knowledge through the 6-years development, and then will maitain to be able to speed up constructing noun tables by using parallelization of the process on PC cluster and retrieving datum to answer users' queries by introducing cache tables for retrieving.

1. 初めに

我々は 2003 年から卒業研究及び産学連携研究の一環として，膨大な Web 上のテキストデータを自動的に収集し，日本語形態素解析ソフトウェアを用いて名詞のみ取り出して検索する小規模なシステムを作成してきた。昨年度まで得られていた知見によれば，日本語形態素解析部分の速度が非常に遅く，名詞テーブル作成に要する時間を短縮する必要があることが判明していたため，日本語形態素解析部分を PC クラスタ上で素朴に並列分散化し，テーブル作成時間の短縮に成功した。また，検索結果をあらかじめ保持しておくためのキャッシュ機構も導入し，検索時間を大幅に短縮することも可能になった。これで，本システムは日本語形態素解析による単語レベルのデータマイニングを可能にする高性能な並列分散情報処理機構として，一応の完成形をなしたと言える。

本稿では，今まで得られた知見を活かしてどのようなシステムを構築してきたか，そのあらましを述べた後，本年度で達成された名刺テーブルの並列分散生成処理と，キャッシュ機構の成果を，ベンチマークテスト結果で明らかにする。

2. サーチエンジンシステムの概要と開発の意義

我々が 2003 年以来作成してきた小規模な Web 検索システムを Fig.1 に示す。

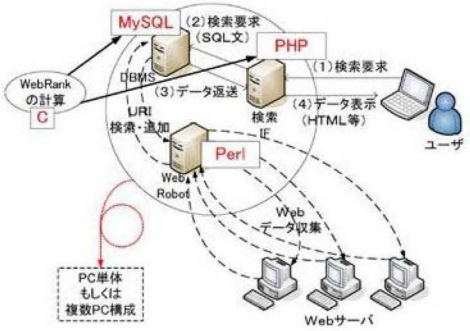

Fig. 1: 小規模サーチエンジンのシステム

Web Robot が自動収集した Web コンテンツデータを Web データベースサーバに蓄え，ユーザの検索要求に対しては，基本的にはここに蓄えたデータ (Main テーブル) を検索して結果を返すようになっている。Web Robot と Main テーブルを合わせてここではサーチエンジンの **Web データ収集部**と呼ぶことにする。

しかし，莫大な Web データを検索要求に合わせてその都度全部総ざらいするようなことをしていては，非力な I/O 処理能力しかない PC では，結果を得るまでに数十秒，長い時には数分以上の時間がかかることになる。これは迅速な処理を期待されるサーチエンジンシステムに置いては致命的な欠陥となる。

そこで我々は，通常の検索要求では「名詞」が主体となることを利用し，あらかじめ必要な名詞のみを切り出し (Fig.2)，その名詞を含む Web データをまとめて保存しておく**名詞テーブル**を作成し，ユーザの検査要求に対してはまずこの名詞テーブルを取り出すという，**検索部**の改良を 2008 年度卒業研究の一環として行うことにし，一応の完成を見た [15]。

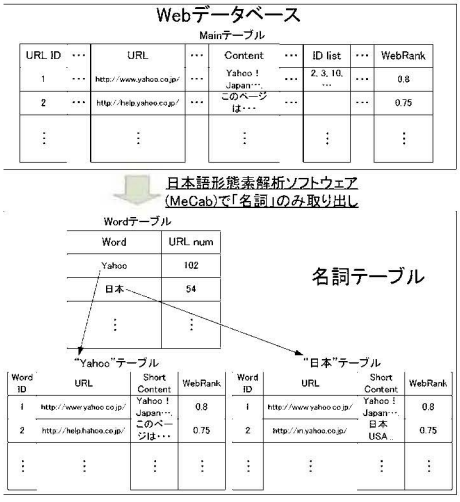

Fig. 2: 日本語形態素解析部分

この名詞の切り出し，及び名詞テーブルの作成には，日本語形態素解析ソフトウェアである MeCab[3] を使用している。しかし後述するように実用的には更なる検索速度の向上が求められることが課題として残された。

以下，我々のサーチエンジンシステムの開発の歩みとそこ

2010 年 2 月 13 日 受理

*情報システム学科

で得られた知見を，Web データ収集部と検索部に分け，かいつまんで述べることにする。その後，これらのサーチエンジンシステム開発の意義を再確認しておきたい。

2.1 Web データ収集部

Web データ収集部は，如何にして大量の Web データを収集できるかが重要である。Web データの収集のアルゴリズムはごく単純なもので，最初に与えた URL にアクセスし，そこから未知のリンク (URL) を収集し，そこを辿って更に未知の URL を収集する … というだけのものである。しかし Web の世界は様々な不完全なデータに満ちており，安定的に正しい未知 URL(アクセス可能かつ既知の URL とは内容が異なるもの) を集める Web Robot のアルゴリズムを見い出すには試行錯誤が必要である。この部分は竹口友大 [2]((株) わいにじ) が担当し，データベースと PC リソースが許す限り安定して動く Perl スクリプトとして実装したものを使用している。

これを土台に，更に大量の Web データを集めることが出来ないか実験してみた [2]。B フレッツの回線を使っているため，限られた回線帯域の下でどこまで収集可能な URL 数が増やせるかが勝負となる。まず，1 台の PC のマシンリソースの限界を確認するため，SSD(Solid State Drive) とキャッシュメモリを搭載した RAID カードを組み合わせて I/O 性能を高め，Quad-core の CPU を使って Web Robot をマルチスレッド化して並列 Web 接続が可能なようにした。結果として，一週間で約 19 万 URL まで性能を高めることができ，これが一台の PC における最高記録となっている。

我々は PC cluster を用いて更に性能向上が図れないかを確認するため，2009 年度の卒業研究において，Web Robot の並列動作を行ってみたが [1][16]，Main テーブルへのデータ処理が足を引っ張り，性能向上はせいぜい 2 並列程度までということが判明した。結果として，現状のネットワークと PC リソースを使う限り，PC cluster による性能向上のためにはデータベース処理を並列分散化する他ないという結論が得られたことになる。

2.2 検索部

前述したように，Main テーブルを直接検索するだけでは検索速度の向上が図れないため，我々は名詞テーブルを導入し，検索部の改良を行った。その結果，Main テーブルにある生の Web データを直接検索対象とした場合に比べて約半分の検索時間になることを確認した [15]。しかし名詞テーブルを作成する時間が膨大になるため，2008 年度までの段階ではこの部分の高速化が課題として残った。本年度はこれを PC cluster 上で並列分散処理化し，後述するように大幅な時間短縮を達成した。

更に検索速度を向上させるため，検索要求に合わせた複合名詞テーブルをキャッシュとして作成することにし，複雑な検索要求にも瞬時に検索結果を表示させることが出来るようにした。これの検証結果についても後述する。

2.3 サーチエンジンシステム開発の意義

以上，Web データ収集部と検索部の改良のあらましを述べてきたが，ここで得られた知見は目新しいものではない。むしろ，常識の範囲内のものと言える。学術的に新規なものは皆無と言っていいだろう。それでもなお，サーチエンジンシステムを 6 年間にわたって追求してきたことによって得られたものは多い。

まず，Web とデータベースの連携が必須になってきたこの時代においては格好の「教材」であるということが言える。Web データ収集部では指定された URL にアクセスし，そこから HTML をパーサにかけて未知 URL とタグを取り除いたコンテンツテキストを取り出す必要があるが，この処理は当然 URL や HTML というものをきちんと理解し，適切な Perl モジュールを組み合わせて組み上げなければならない。検索部においては，蓄えた Web データを適切な SQL 文で指定し，検索して HTML として整形しなければならない。これは 3 層 Web プログラミングそのものである。このように，Web 時代の基礎知識とデータベースの知識がなければ組み上げることが出来ないサーチエンジンシステムは，OJT 的に優れた教材であると言える。

次に，データベースの処理能力がどの程度のものであり，どのようにしてサーチエンジンシステム全体のボトルネックになるか，ということが体験的に理解できるということが挙げられる。今回我々が作成したシステムはデータベースの処理能力，特に Main テーブルや名詞テーブルを保存してある PC 上の外部記憶装置 (SSD やハードディスク) の性能が全体の処理能力を決める。我々のシステムにはべき乗法を用いた簡易なランキングシステム (WebRank と称している) を組み込むことが出来るが [9][11]，WebRank 決定のプロセスの処理時間も，殆どはデータベースからの URL とリンク情報取り出しに要するファイル入出力時間が占めていることが判明している。このように，膨大なデータを扱うシステムではデータベースが必須であり，その処理時間をどの程度減らすことが出来るかということに腐心する必要がある。このような現実的な問題と向き合うことで，理論だけでは分からない，直接目に見えない「情報処理」を体感することができるようになるのである。

最後に，Google や Yahoo! Japan のような巨大なサーチエンジンシステムに比べて，ごく小規模な我々のシステムも，かなりの経験と複雑なプログラムの集積が必要である，ということが体感として理解できるという事実を挙げておく。実際，本年度は検索部の高速化を行ったが，それは昨年度までのプログラムの集積があって初めて可能になったものである。性能はともかく，とにかく動くプログラムが用意されているのと，スクラッチから全てを構築するのでは開発とベンチマークに必要な時間が全く異なる。本年度の改良が可能になったのも，昨年度までの蓄積があったればこそであった。

現在の情報処理システムは，数十年に渡って蓄積されてきた複雑なプログラムの集積によるものが多い。スクラッチからものを作り上げる体験が重要であることは当然だが，過去の蓄積の上に，次世代にバトンを渡すための「積み上げ的プログラム」を行うことも，先人の経験をプログラムを読み解くことで知ることが出来るという意義も含めて大切な教育的体験となるであろう。

3. 名詞テーブル作成の高速化

前述してきたように，我々の小規模サーチエンジンシステムの検索部においては名詞テーブル作成作業の短縮化が一つの課題となっていた。そこで，この作業を素朴に PC cluster 上で並列分散処理化することにした。

今回は CentOS 5.3 を導入した Pentium IV 2.8GHz (P4) PC を 11 台用意し，NFS/NIS サーバを介して/home および/usr/local 以下を NFS 共有したクラスタの上で実験を行っている。この上で，日本語形態素解析部分のみを Fig.3 に示すように PC クラスタ上で割り当てられる名詞数が平等になるように並列分散処理を行った。名詞テーブル作成作業をなるべく平等に割

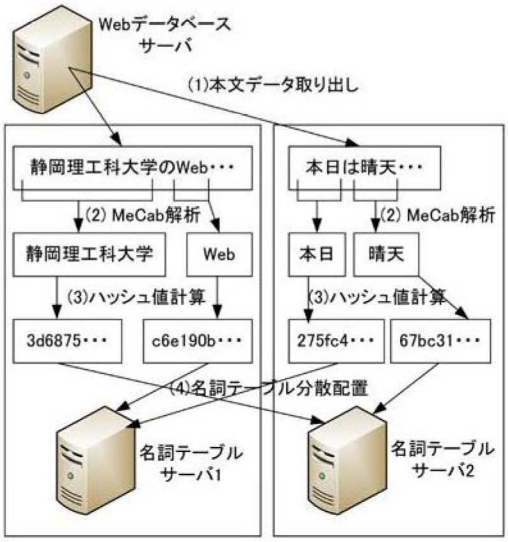

Fig. 3: 日本語形態素解析の流れ

り振るため，MD5 による名詞のエンコードを行い，MySQL によるデータベースは各マシンのローカルなハードディスク上に作成するようにしてある。そのため，Web 収集データを格納したデータベースサーバマシンへの負荷集中がそれほど無ければ，MeCab プロセス及び MySQL データベース処理に要する時間をリニアに減らすことが期待できる。

以上の構想に基づいて名詞テーブル作成処理の並列分散化を実現した。ここでは実際のところどれほどの効果があったのかを検証してみる。Web データは URL1000 件のページデータを使い，名詞テーブル作成に 1,2,5,10 台のマシンをそれぞれ割り当てて処理した時の処理時間の比較を行った。その結果を Fig.4 に示す。

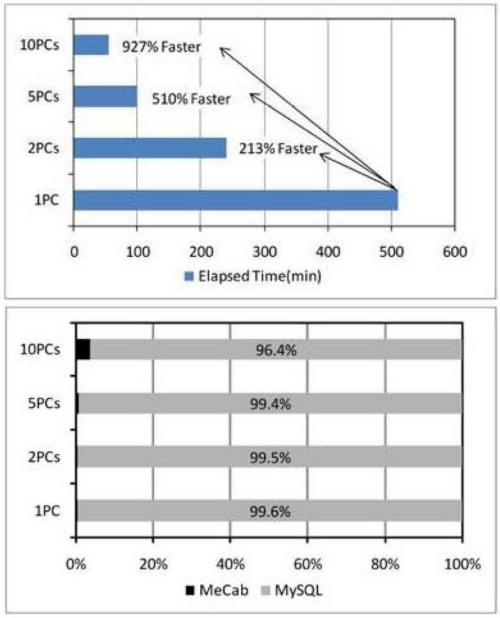

Fig. 4: 並列処理時間 (上) と処理時間に占める MeCab/MySQL の割合 (下)

Fig.4 上のグラフより，台数が増えれば増えるほど処理時間は短くなっており，ほぼリニアに台数効果が現れていることが分かる。なお Web データを格納してあるデータベースサーバマシンへの負荷を考慮し，名詞テーブル作成スクリプトには数十秒程度のウェイトをかけていることを考えると，名詞テーブル作成にかかる時間の大きさがよく分かる。

Fig.4 下のグラフでは，名詞テーブル作成全体における，MeCab の平均処理時間と MySQL の平均処理時間の割合の比較を行っている。当初は MeCab が処理の足を引っ張り速度を低下させていると予想していたが，実際には殆ど影響がないことが分かる。これは偏に MySQL データベースの処理における I/O のボトルネックよるものである。また，並列処理の台数が増えるほどに MeCab の時間の割合が増加しているが，これは Web データベースサーバへの同時アクセスによる負荷の増大によるものと考えられる。しかし 10 台程度の並列分散化では処理時間全体にかかる時間に占める比率は小さい。

4. キャッシュテーブルの導入とその効用

ユーザによる検索要求が名詞一単語だけであれば，該当する名詞テーブルの内容を整形して表示すれば良いが，「静岡県袋井市」のような複合的な要求であれば「静岡県」テーブルと「袋井市」テーブルを，AND 検索・OR 検索それぞれに対応してドッキングした結果を表示する必要も出てくる。このような検索要求に対しては，単語ごとの名詞テーブルが個別に存在するだけでは処理時間が多くなる傾向がある。

そこで，あらかじめ予測される複合検索要求に対して，検索結果をキャッシュしておく機構も導入し，Google で提供されているような高速表示が可能かどうかを実験することにした。そのベンチマークテストの結果を Fig.5 に示す。

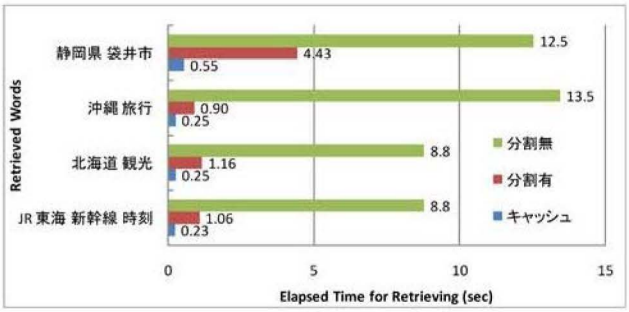

Fig. 5: キャッシュテーブル導入の効果

次の 4 つの複合検索要求

1. 「静岡県」+「袋井市」
2. 「沖縄」+「旅行」
3. 「北海道」+「観光」
4. 「JR 東海」+「新幹線」+「時刻」

に対してどの程度の時間短縮が可能かどうかをベンチマークテストしてみた結果である。一目瞭然だが，名詞テーブル導入無しの場合は，例えば 1 の場合，Main テーブル検索 (分割無) に 12.5 秒かかるものが，名詞テーブルのドッキング処理 (分割有) だけだと 4.43 秒で済み，さらにあらかじめこのドッキング処理結果をキャッシュしておけば (キャッシュ)，キャッシュテーブルの取り出しの 0.55 秒で済むようになっている。

2〜3 の場合も全く同様の結果が得られており，更に Web データが膨大になるとこの差はますます開いていくことになるであろう。

このように，大量のテキスト処理結果をあらかじめキャッシュしておくことで，巨大なサーチエンジンシステムの高速な応答性が維持されていることが理解できる。

5. 結論と今後の課題

以上，6 年間に渡って行われてきた我々のサーチエンジンシステムにおける知見・意義，そして本年度行われた検索部の高速化の結果について述べてきた。現状のネットワーク・マシンリソースである限り，ほぼ本年度までで達成されたプログラム体系・高速化で完成形に達したと言える。ことに，本年度導入した並列分散名詞テーブル作成機構と，キャッシュ機構は，大量の Web データを扱う本システムの要と言えるものであり，これが完成したことで，巨大な既存サーチエンジンシステムと同様の技術がこの中で実現されていることになった。

今後の課題としては，本年度で完成形となった本サーチエンジンシステムをコアにして，次のような機能とカスタマイズ化を行うことが挙げられる。

サーチエンジンシステムの「見える化」

本サーチエンジンシステムはオープンキャンパス等でデモ展示を行ってきたが，中身についての目に見える解説が行えなかったという問題があった。ことに，どれだけ膨大な Web データを Main テーブルに保持し，それが常に更新され増大していくという状態を知らしめることが目的なのに，それをを可視化することができないということは，本システムの一番の「売り」を見せられないということである。デモを行うにしても，せいぜい検索結果を示すのが関の山であったから，我々の苦労やシステムの有効性をなかなか伝えきれないというもどかしさが常に残っていたのである。

そこで，MRTG や Munin のような，マシンリソースの状態を時系列でモニタリングする機構を，我々のサーチエンジンシステムにも導入し，常にどの程度の Web データがあり，どの程度の単語数が登録されているのかを「見える化」するようにしたい。具体的には PHP を使った Web インターフェースとして実現することになるであろう。

膨大なテキストデータを対象としたマイニングシステムへのカスタム化

ネットワークリソースが限られているため，世界中の Web データを収集することは不可能である。Web データ収集は今後も続けていくが，応用としてはもう少し現実的な量の，それでいて PC 単体では時間を要する程度のテキストデータをマイニングするツールとして，カスタム化が容易なものにしたいと考えている。具体的には

Web データ収集部 テキストの形態 (CSV 形式など) に合わせて容易にカスタマイズできるようする

検索部 単語を名詞だけでなく他の品詞にも対応したものにしたり，キャッシュテーブルをユーザの検索要求が増えるごとに記憶しておく機構を導入したりする

ということを可能にし，その上で一つのソフトウェアパッケージとして提供できるようにしたい。これはまだ無謀というべきかもしれないが，ある程度軌道に乗った段階で一つのオープンソースプロジェクトとすることも検討してきたい。

謝辞

本研究は，平成 19 年度から行ってきた卒業研究の集大成としてなされたもので，ことに 2007 年度の山本達文 [13]，2008 年度の仁藤昌彦 [15]，吉田嗣 [14]，黒田博雅 [16] らによる結果があってこそのものである。教員の独断によって設定された研究テーマに一年間お付き合い頂いた彼らの労に対して感謝申し上げる。また本研究に使用した PC 機材は静岡理工科大学学内研究費の補助で購入したものである。関係者各位に感謝する。

参考文献

1) 幸谷智紀, 小規模な分散 Web ロボットの最適化に関する一考察, 第 71 回情報処理学会全国大会講演集, 2009.
2) 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.
3) MeCab, http://mecab.sourceforge.net/
4) 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.
5) 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
6) The Community ENTerprise Operating System, http://www.centos.org/
7) Movable Type, http://www.movabletype.com/
8) http://www.robotstxt.org/
9) Google の秘密 - PageRank 徹底解説, http://www.kusastro.kyoto-u.ac.jp/~baba/wais/pagerank.html
10) MySQL, http://www.mysql.com/
11) A.N.Langville, C.D.Meyer, Google's PageRank and Beyond, Princeton University Press, 2006.
12) 幸谷智紀, ソフトウェアとしての数値計算, http://na-inet.jp/nasoft/
13) 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.
14) 吉田 嗣, ”Web データ収集構造を視覚化するためのプログラムの作成”, 2008 年度静岡理工科大学情報システム学科卒業研究.
15) 仁藤昌彦, ”サーチエンジンの検索速度向上に関する研究”, 2008 年度静岡理工科大学情報システム学科卒業研究.
16) 黒田博雅, ”並列分散型 Web データ収集システムの改良”, 2008 年度静岡理工科大学情報システム学科卒業研究.