パイプラインに基づくインデックス更新過程の改良

宇田川 稔† 佐藤 永欣† 上原 稔† 酒井 義文† 森 秀樹†
東洋大学工学部情報工学科†

1. はじめに

組織内では公開されている Web 文書より、はるかに多い非公開の Web 文書があると考えられる。このような埋もれる情報を発掘するには組織内情報検索システムが必要である。また、このようなシステムでは、新鮮な情報を検索できることが必須の機能である。従来のサーチエンジンは１つのロボットにより文書を収集して検索を行う集中型アーキテクチャを採用している。このようなアーキテクチャでは、現在の情報の増加や多種多様の情報形態から、新鮮な情報を検索するには限界がある。そこで、我々は分散型サーチエンジン「協調サーチエンジン」(Cooperative Search Engine, CSE)[1]を開発した。CSE では、個々の Web サーバにインストールされた局所的なサーチエンジンをメタサーチエンジンが統合することで、文書収集を不要とする。この結果、更新間隔の大幅な短縮に成功した。しかし、すべての Web サーバに局所的サーチエンジンを組み込むことができるわけではない。その場合、CSE の長所を十分に活かすことができない。そこで、本文では従来方式の集中的文書収集を効率的に行う方式について考察する。

従来から文書収集の問題は広く認知されていたため多くの研究がなされている。１つは通信遅延を考慮した分散ロボットによる分散収集[2]、もう１つはネットワークトラフィックを限界まで用いるための超並列収集[3]などがある。しかし、これらは文書収集のみに限定した方式である。我々の経験では、サーチエンジンでは、文書収集よりインデックス生成がボトルネックとなる可能性があることがわかってきた[4]。そのため、文書収集とインデックス生成の両プロセスを最適化する方法論が必要となる。この問題に対して、我々はスループットを最大化することで対処する。

本文では、スループット計算のためにプロセスモデルを定義し、それに基づきインデックス更新処理を最適化した。

2. プロセスのモデリング

各プロセスは互いにキューで結合され、データ駆動で動作する。生成され、初期化されると、以下の動作を繰り返す。まず、入力キューにトークンが挿入されるのを待つ。キューにトークンが挿入されると実行可能になる。実行状態になると、キューからトークンを取り出し、特定の処理を行い、出力キューへ挿入する。もし、出力キューがいっぱいなら空くのを待つ。プロセスの動作を図１(a)に示す。このとき濃色の矢印は実行中の状態を示し、その長さは実行時間 Tr と等しい。また、無色の矢印は待ち状態を示し、その長さは待ち時間 Tw と等しい。このような周期で実行されるプロセスのスループット（単位時間当たりの平均出力）は、以下のようになる。

$$\mathbf{1/(Tr + Tw)} \quad \mathbf{[token/s]} \text{ (式 1)}$$

したがって、スループットを向上させるには Tw を 0 に近づけることが重要となる。

複数ノードにおけるパイプラインを図１(b)に示す。例では、３ノードにそれぞれ１つずつプロセスが配置されている。このような全体プロセスのスループットは、以下のようになる。

$$\min_{o<i\le n} \mathbf{1/(Tri + Twi)} \quad \mathbf{[token/s]} \text{ (式 2)}$$

このとき最小のスループットを持つプロセス p がボトルネックとなる。スループットを向上させるには個々の Tw_i を 0 に近づけることと Tr_p を減らすことが重要である。

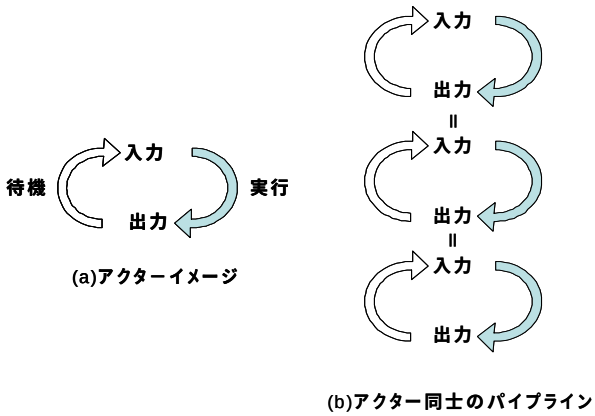

図 1. プロセスとパイプラインの種類

3. インデックス更新パイプライン処理

インデックス更新をパイプライン化した構成を図 2 に示す。プログラムは、オブジェクト指向言語である Ruby を用いて構築し、各プロセスをスレッドにより細分化することによって、ボトルネックの把握を容易にした。

Improving pipeline based index updating process
† Minoru Udagawa, Nobuyosi Sato, Minoru Uehara, Yoshifumi Sakai, Hideki Mori
Department of Information and Computer Science, Toyo University

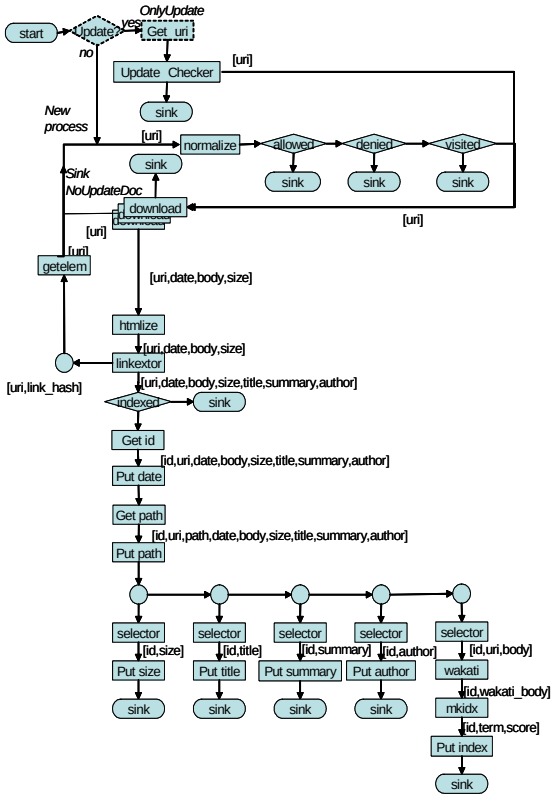

図 2. インデックス更新のパイプライン

4. 評価

ボトルネックを特定するため，実験を行った。実験には，PC/AT 互換機(CPU:Celeron 1.2GHz，Mem:448MB、NIC:Intel/pro 100+)のマシンを用いた。

インデックス更新の各プロセスの 1 回の呼び出し当たりの平均処理時間を図 3 に示す。

対象 Web サーバ数　学内 52 サーバ
対象文書数　4536 文書(43524KB)

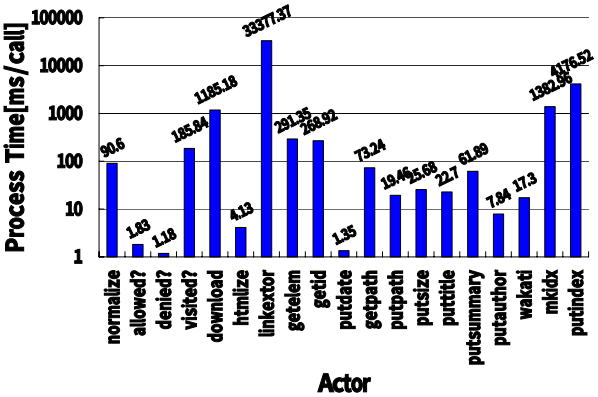

図 3. インデックス更新過程の処理時間

ボトルネックになっているのは、download、linkextor、mkidx、put index である。それぞれ、文書収集、リンク抽出、インデックス作成(put index を含む)に対応するプロセスである。まず文書収集は、全体から見るとボトルネックになっているが、1 文書当たり約 1 秒であり、これ以上の短縮は難しい。よって文書収集は、多重化によって対処した。並列度を変え収集を行った結果を図 4 に示す。

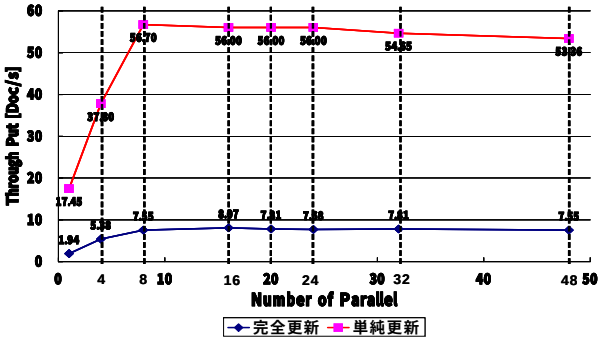

図 4. 文書収集の並列処理スループット

多重化により高速な文書収集が、実現できた。しかし、ただ並列数を増やしただけでは、ネットワークが飽和してしまい、スループットは落ちてしまう。

リンク抽出とインデックス作成は、行単位でタグ、語の解析処理を行っているため、処理時間が増えていると考えられる。この 2 つは、大きな処理時間から文書収集のように 1 プロセッサ内での多重化では、対処できないと考えられる。そこで我々は、このボトルネックに対し、複数のマシンを用いた並列処理によって対処する。

5. まとめ

今回は、インデックス更新のパイプライン化を図ることにより、処理上でのボトルネックを特定することができ、それぞれの対処法を考案した。今後、このボトルネックの解消を中心に、CSE で用いられていたアクセス方法の導入により、高速な処理を図っていく。

参考文献

[1] Nobuyosi Sato, Minoru Uehara, Yoshifumi Sakai, Hideki Mori, "Fresh Information Retrieval in Cooperative Search Engine," In Proceedings of the ACIS 2nd International Conference on Software Engineering, Artificial Intelligence, Networking & Parallel/Distributed Computing(SNPD'01), pp.104-111, (2001.8.20)

[2] 山名早人等 「分散型 WWW ロボットによる WWW 情報収集」 DEWS'98 、http://www.etl.go.jp/~yamana/Publications/ABST/DEWS98/24.htm

[3] 能登信晴等 「スケーラブルな WWW 情報収集ロボットの設計と実装」 DPSWS (2000)

[4] 宇田川稔，佐藤永欣，上原稔，酒井義文，森秀樹 "組織内における最新情報検索のための高速インデックス更新"，マルチメディア・分散・協調とモーバイル(DICOMO'2002)シンポジウム論文集, pp.129-132, 情報処理学会 (2002)