スケーラビリティを考慮した分散型インデックスパイプライン手法

宇田川 稔† 佐藤 永欣† 上原 稔† 酒井 義文‡

東洋大学工学部情報工学科†，東北大学大学院農学研究科‡

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。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。しかし全ての 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。そこで従来方式の HTTP アクセスを用いた文書収集を効率的に行う方式について考察する必要がある。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。しかしサーチエンジンでは文書収集よりインデックス作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[2]。そのため最新情報を検索できるサーチエンジンを構築するためには、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。この問題に対して文書収集とインデックス作成をインデックス更新処理として定義し、ボトルネックを解消しながらスループットを最大化することで対処してきた[3]。本稿では、現在まで構築してきたインデックス更新のパイプライン処理を分散化する手法について説明する。

2. クラスタマシンを用いた分散手法

2.1 分散ポイントの特定

インデックス更新のパイプライン処理の中で分散を行うポイントとして、2 ヶ所考えられた。最初のポイント(以下 Distributed1)は、1 番に URI が送られるポイントである。ここのポイントではリンク解析され、抽出された URI が最初に流れるポイントである。このポイントで新しい Web サーバの URI が検出された場合、他のマシンに URI を渡すようにする。このことにより各所に散らばったマシンからその URI の Web サーバに最も通信遅延が小さいマシンに処理を割り振ることも可能で最適化された文書収集が可能になる。

次のポイント(以下 Distributed2)は、各要素(サイズ、タイトルなど)をデータベースに格納する前である。要因はインデックス作成の工程のボトルネックである。この工程は、大きなボトルネックでありマシンパワーが大いに依存する。このことによりマシンパワーが比較的空いているマシンに割り振ることが必要になる。サイズやタイトルなどの格納の前にポイントを置いたのは、文書 ID の整合性を考慮したからである。Distributed2 のポイントで分散する場合、文書に対するコスト計算が必要になる。このコストを考慮することで分散処理を最適化する。コストを次のように定義する。インデックス更新のパイプライン処理は、全ての処理が文書に依存する。しかし文書といっても、文書 ID、URI、サイズ、通信遅延など各アクターにより依存するものが違う。よってそれぞれの要素を含む文書配列 X を以下のように定義する。j は文書 id とする。

$$X_j = (id, uri, path, date, body, size, title, summary, author \Lambda) \quad j = 1,2,\Lambda\, m$$

Distributed2 では、インデックス作成の工程の各文書に対するスループットにだけ考慮する。Distributed2 を通る文書 Xj の配列は、文書サイズを保持する。この文書サイズとその文書の実験より得られたアクターの処理時間用いれば、この文書がインデックスを作成するのにどれだけコストがかかるか予測することができる。実験結果より、平均データサイズは 6.925[kB],インデック作成工程の平均処理時間は 2.6[s]であるので、

$$Cost(X_j) = (Size_j * 2.6) / 6925 [s/B]$$

として、コストを計算する。このコストを用い、各マシン内でボトルネックにならない上限値 $Cost_{upper}$ を設定し、分散処理を図る。

2.2 動作

実際に各マシンを協調することを考える。情報を各マシンが共有するために、TupleSpace(以下 TS)を用いる。TS とは、分散システムのための分散共有メモリである。TS を用いた分散型インデックス更新過動作を Distributed1 と Distributed2 に分け示す。(図 1 参照)

(Distributed1)

I. 最初のページを持つマシンは、処理を開始する。それと同時に他のマシンも起動する。
II. リンク抽出により、新しい web サーバを抽出した場合、その URI を Tuple として、TS に投入する。
III. 待ち状態にある 1 台のマシンは、TS から URI を取り出し処理を開始する。それと同時に自分がどの web サーバ処理しているかという情報を TS に置きすべてのマシンに伝える。そしてリンク抽出により新しい URI を抽出した場合、II に戻る。また、他のマシンで処理している Web サーバを抽出した場合、その URI をそのマシンの Distributed1 に渡す。

(Distributed2)

I. 文書 Xj のコストを計算する。各マシンで設定された上限値 $Cost_{upper}$ を、インデックス作成工程を待っている文書のコストの合計が超えた場合、その後の文書 Xj を Tuple として TS に投入する。上限値 $Cost_{upper}$ は、余裕を持ち設定する。あまりここで TS に Tuple を投入すると、ボトルネックになり、多くの Tuple が TS に貯まってしまうことが予想される。
II. コストが上限を満たしておらず、各自のマシンに文書が貯まっていない場合、TS から、Tuple を取り出し、

The Distributed Index Pipeline in Consideration of the Scalability

†Minoru Udagawa, Nobuyosi Sato,Minoru Uehara, Department of Information and Computer Science, Toyo University

‡Yosifumi Sakai, Graduate School of Agricultural Science,Tohoku University

インデックス作成工程を行う。

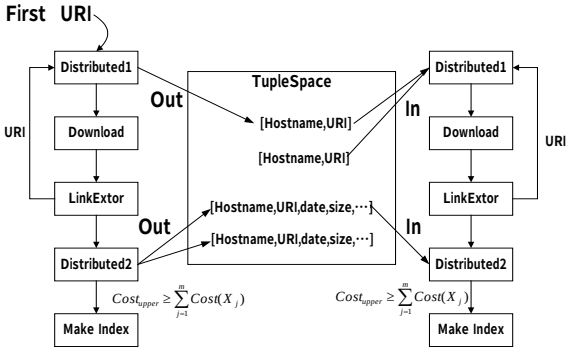

図 1.　分散型インデックス更新手法の構成

2.3　評価

9 台のクラスタマシン(CPU:Celeron1.2GHz, Mem:256MB,NIC:Intel Pro/1000 MT)を用い、分散型インデックス更新パイプライン処理について実験を行った。対象は学内に存在する 52 サーバで、TS は Cluster Server 上に配置し、各マシンの文書収集の並列度は前実験[3]より 16 と設定した。

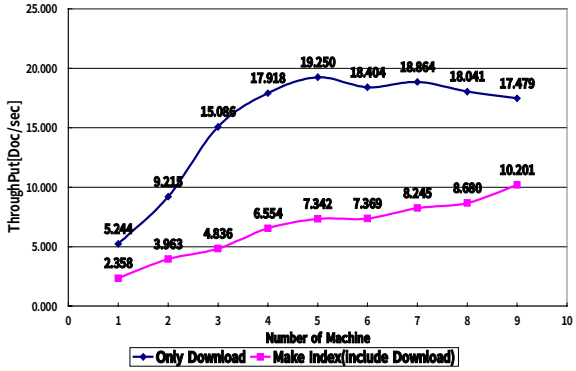

図 2.　分散型インデックス更新手法のスループット

まず文書収集のみの場合は、スループットの向上が見られるが、マシン数が 5 台くらいで頭打ちになっている。これは、各 Web サーバの文書量の違いから多い文書から各マシンの中でアイドル時間が生じてしまうことが原因であると考えられる。この問題に対し、1 台のマシンに対し web サーバ 1 つではなく、ディレクトリ単位に分割し複数台で処理されることが考えられるが、最初にその web サーバが文書の多いサーバか分からないから、分割することがロスを生じてしまうことも考えられる。この問題については現在考察中である。それに比べ、インデックス作成までの方は、マシン数を増やすことでスループットの向上が見られる。これは、アイドル時間が生じた場合、他のマシンで溢れた文書を処理することでインデックスの生産性を高めていると言える。

3.　スケーラビリティを考慮した分散手法の設計

2 章では限られたマシン数で行われたが、マシンを増やすことを考える。このマシンを増やすとき重要なのがどこにマシンを配置するかである。最も効率がいいのは、Web サーバから通信遅延が小さいところに配置することである。各マシンと Web サーバの距離を判定する方法として、2 つの方法が考えられる。1 つはドメイン部分の比較、2 つ目は Ping での実際の判定である。

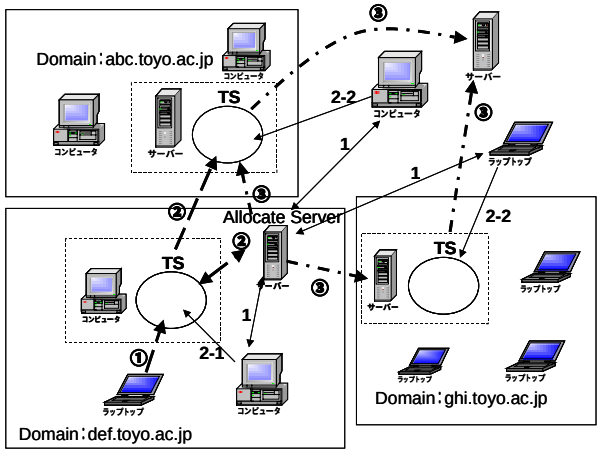

図 3.　スケーラビリティを考慮した分散手法の構成

3.1　動作

新しくこのシステムに参加しようとするノードは、Allocate Server(以下 AS)にアクセスする(図 3_1)。AS はドメインを比較し、同じドメイン内に TS がある場合、そのノードに Tuple スペースの場所を教える(図 3_2-1)。ない場合、幾つかの TS の候補をそのノードに送り、通信遅延を計測させる。そして、最も通信遅延が小さい TS を配置させる(図 3_2-2)。処理の方法は、常に各ノードは割り当てられた TupleSpace の処理を行う。違うドメインの Web サーバ(Tuple となる）を検出した場合、割り当てられた TS に送る(図 3_①)。TS はドメインを判定し、違うドメインである場合は、 Allocate Server に問合せ、最も近い TS を教え、その Tuple を移す(図 3_②)。全く似たドメインがない場合、AS は、全ての TS に通信遅延を測らせる。そして最も近い TS に Tuple を送る(図 3_③)。各ノードの動作は前述の TupleSpace へのアクセス法と同じである。これらの処理により、各ノードと Web サーバ間の通信遅延を最小に抑えられ、スループットの向上が期待できると考えられる。

4.まとめ

今回は、分散型インデックス更新過程の設計をし、クラスタマシンを使い評価を行った。そして、スケーラビリティを考慮した分散型インデックス更新手法について説明した。今後の課題として、今回説明したシステムを実装し、4 キャンパスある学内での評価を行う。

参考文献

[1] Nobuyosi Sato, Mioru 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] 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"，(DICOMO'2002)シンポジウム論文集，pp.129-132,(2002)

[3] 宇田川稔，佐藤永欣，上原稔，酒井義文，森秀樹 " 組織内情報検索におけるインデックス更新過程のパイプライン処理"，(DICOMO 2003)シンポジウム論文集，pp.809-812（2003）